IBM® Tivoli® Federated Identity Manager
Version 6.2.2.7

Configuration Guide

GC27-2719-02

IBM® Tivoli® Federated Identity Manager
Version 6.2.2.7

Configuration Guide

GC27-2719-02

Note
Before using this information and the product it supports, read the information in "Notices" on page 785.

Edition notice

Note: This edition applies to version 6, release 2, modification 2.7 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 configure IBM Tivoli Federated Identity Manager.

Intended audience

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

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

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

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 xviii.
- A link to the "IBM Terminology website" on page xviii.

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.

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

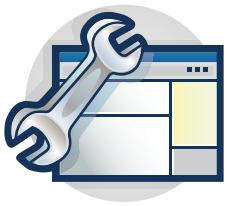
Operating system-dependent variables and paths

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.

Part 1. Federation First Steps tool setup and use



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

You can use the Federation First Steps tool to create a SAML 2.0 federation based on a template and your preferred role.

Start with the topic Chapter 1, “Customizing federation templates,” on page 3.

Chapter 1. Customizing federation templates

The Federation First Steps tool contains federation templates which you can use to create SAML 2.0 federations. The federation templates are response files that contain macros which are expanded during runtime. You can edit the federation templates to customize some properties.

About this task

Before you use the Federation First Steps tool to create a federation, you can customize the federations templates. The following sections guide you through the process.

- Customizing the federation template
- Using the customized federation template

Customizing a federation template

About this task

You can customize a federation template in two ways:

- Modifying the federation template in the `fedfirststeps` directory
- Modifying the federation template in a different directory

Modifying the federation template in the fedfirststeps directory

Procedure

1. Go to `<FIM install folder>/firststeps/fedfirststeps/templates` and locate the template that you must customize.
2. Use a text editor to edit the template that you must customize.
3. Edit the variables that you must change. You can specify your preferred value for the company name, mapping rule, and so on.
4. Click **Save**.

Modifying the federation template in a different directory

Procedure

1. Go to `<FIM install folder>/firststeps/fedfirststeps/templates` and locate the template that you must customize.
2. If you must put the templates folder in another location, copy the templates folder from `<FIM install folder>/firststeps/fedfirststeps/templates`. Move it in another directory, and rename it.
3. Use a text editor application to customize the template.
4. Edit the variables that you must change. You can specify your preferred value for the company name, mapping rule, and so on.
5. Click **Save**.

Using the customized federation template

About this task

There are two ways to use the customized federation template in the Federation First Steps tool: by modifying the fedfirststep.ini file, or by using the command-line interface. If, for example, your custom templates are in `c:\custom_tfim_fed_templates\`, follow these procedures:

Procedure

- Modify the fedfirststep.ini file, then run the Federation First Steps tool.
  1. Go to `FIM_Install/tools/fedfirststeps/fedfirststeps.ini` and open the fedfirststep.ini file in a text editor.
  2. Add **`-custom-template-dir`**.
  3. Click **Save**.
  4. Launch the Federation First Steps tool.
- Use the command-line interface
  1. Open the command-line interface.
  2. In the command-line interface, enter:
     - **`fedfirststeps.exe -custom-template-dir`**
     - `c:\custom_tfim_fed_templates`

Chapter 2. Federation First Steps tool

Use the Federation First Steps tool to create a generic SAML federation, configure risk-based access, or add service providers as a partner.

The Federation First Steps tool has the following limitations:

- The Federation First Steps tool User Interface does not support wrapping of the check box label. The text does not display in the next line and gets truncated because of a known Standard Widget Toolkit issue.
- The use of Federation First Steps tool in cluster environments is not supported.

Launching the Federation First Steps tool

Launch the Federation First Steps tool to create federations.

Before you begin

- For Linux and Solaris users, ensure that you are running GIMP Toolkit (GTK) version 2.12.0 or later.
- For AIX® users, ensure that you have the latest version of `motif` installed.

Procedure

Choose any of the following options to launch the Federation First Steps tool.

- In the installation wizard, the **Launch the First steps console** option check box is selected by default. Click **Finish**.

  Note: This option is only available on a new installation of Tivoli Federated Identity Manager.
- In the `fedfirststeps` directory, launch `fedfirststeps.exe` or `fedfirststeps`.
- Run the following commands in the command-line interface:
  - Microsoft Windows

    ```
    $FIM_INSTALL_DIR\firststeps\fedfirststeps\fedfirststeps.exe
    ```
  - UNIX

    ```
    $FIM_INSTALL_DIR/firststeps/fedfirststeps/fedfirststeps
    ```

    Note: If you are using a 64-bit operating system and the Federation First Steps tool does not launch, ensure that the contents of the `$FIM_INSTALL_DIR/firststeps/fedfirststeps.ini` file is

    ```
    -vm
    $FIM_INSTALL_DIR/_uninst/_jvm/bin
    -nl
    en_US
    ```

Identity provider side configuration

Configure the identity provider federation settings using the Federation First Steps tool.

Creating a generic SAML 2.0 federation with a new or existing domain

The Federation First Steps wizard creates a generic federation so that Tivoli Federated Identity Manager users can access the applications of the service provider.

Before you begin

You must know the following information to complete the wizard:

- The signing key option. See Storage and management of keys and certificates.
- The domain name. See Domain configuration.
- The point of contact server. See Managing point of contact servers.

Procedure

1. Launch the Federation First Steps tool.
2. Select **SAML 2.0 Wizard**.
3. Click **Start**. The tool scans your existing configuration settings.
4. Provide the information that is requested by the wizard.
5. Click **Finish** to start the task processing. Click **Back** if you want to change anything.
6. Optional: Activate the local domain.
   a. Log on to the Integrated Solutions Console.
   b. Click **Tivoli Federated Identity Manager** > **Domains**. A prompt for the detected local domain is displayed.
   c. Click **OK** to activate the local domain.
7. Optional: Check the details of the federation that you created.
   a. Log on to the Integrated Solutions Console.
   b. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations**.
   c. The Federations panel shows a list of configured federations. Select the new federation that you created.
   d. Click **Properties**.
   e. Select the properties to modify. Federation properties are described in the online help.
   f. Click **OK** to close the Federation Properties panel.

Configuring risk-based access with the Federation First Steps tool

Use the IBM Tivoli Federated Identity Manager Federation First Steps tool to configure and enable risk-based access. Risk-based access is a component for IBM Tivoli Federated Identity Manager. Risk-based access provides access decision and enforcement that is based on a dynamic risk assessment or confidence level of a transaction.

Before you begin

Before you start the Federation First Steps tool, complete the following steps:

1. If you have a WebSphere Application Server clustered environment, you must create and configure a JNDI context named `jdbc/rba` in WebSphere Application Server and create the database schema for risk-based access. See Manually configuring the database.
2. Install risk-based access. See Installing risk-based access.
3. Configure the WebSEAL point of contact server for IBM Tivoli Federated Identity Manager. See Configuring WebSEAL point of contact server for SAML federation.

You must know the following information to complete the wizard:

- URL of the Point of Contact Server for IBM Tivoli Federated Identity Manager
- URI of the IBM Tivoli Access Manager secure resource that you want to protect with risk-based access
- WebSEAL instance name

Procedure

1. Launch the Federation First Steps tool.
2. Select **Risk-based Access Configuration Wizard**.
3. Click **Start**. The tool scans your existing configuration settings.
4. Provide the information that is requested by the wizard.
5. Optional: On the General Configuration Settings page, select **Configure Tivoli Access Manager**, if you want to set up IBM Tivoli Access Manager environment so that it delegates authorization decisions to risk-based access for your secure resources.

   Note: If you select this option, you must ensure that IBM Tivoli Access Manager is installed and configured locally on the same system as IBM Tivoli Federated Identity Manager.
6. Specify the URL of the Point of Contact Server for IBM Tivoli Federated Identity Manager, which is used for collecting attributes.

   `http://host_name/webseal-junction-name`

   For example:

   `http://mywebsealhost.company.com/FIM`

   After the configuration process is completed, the Risk-based Access Configuration Summary page describes whether the configuration failed or was successful.

   - If the configuration completes successfully, the next steps to complete the setup for risk-based access are displayed on the Risk-based Access Configuration Summary page. Follow the instructions in the summary page to complete the IBM Tivoli Access Manager and external authorization service (EAS) setup for your environment.
   - If the configuration fails, the log and failure messages are displayed on the Risk-based Access Configuration Summary page. Use the details that are provided in the log and failure messages to check where the configuration process failed and the probable causes of failure. Resolve the configuration issues and run the Federation First Steps tool again to configure risk-based access.
7. Click **Finish**.

What to do next

After you complete all of the next steps that are specified on the summary page, verify that risk-based access is configured correctly on your system. See Verifying the configuration.

Adding a service provider with the Federation First Steps tool

The Federation First Steps wizard adds a service provider as a partner so that Tivoli Federated Identity Manager users can access the applications of the service provider. Supported service providers include Salesforce, Google Apps, Microsoft Office 365, and Workday.

Before you begin

You must know the following information to complete the wizard:

- Partner and domain names of the service provider that you want to add as a partner.
- Federation name
- Federated user names
- If you have multiple domain names in Google Apps, the issuer in the SAML request must be set as `google.com/a/example.com` instead of `google.com`. This option must match the single sign-on option in Google Apps.
- Signing key option
- Point of contact server

About this task

The Federation First Steps wizard adds a service provider as a partner in the following scenarios:

- To an existing domain and an existing federation
- To an existing domain and a new federation
- To a new domain and a new federation

Procedure

1. Launch the Federation First Steps tool.
2. Select the service provider-specific plug-in that you want to use.
3. Click **Start**. The tool scans your existing configuration settings.
4. Provide the information that is requested by the wizard.
5. Click **Finish** if you want to start the task processing. Click **Back** if you want to change anything.
6. Optional: Check the details of the federation that you created.
   a. Log on to the Integrated Solutions Console.
   b. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations**.
   c. The Federations panel shows a list of configured federations. Select the new federation that you created.
   d. Click **Properties**.
   e. Select the properties to modify. Federation properties are described in the online help.
   f. Click **OK** to close the Federation Properties panel.

7. Optional: Check the details of the partner that was created.
   a. Log on to the Integrated Solutions Console.
   b. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Partners**.
   c. The Partners panel shows a list of configured partners. Select the new partner that you created.
   d. Click **Properties**.
   e. Select the properties to modify. Partner properties are described in the online help.
   f. Click **OK** to close the Partner Properties panel.
8. Optional: Activate the local domain.
   a. Log on to the Integrated Solutions Console.
   b. Click **Tivoli Federated Identity Manager** > **Domains**. A prompt for the detected local domain is displayed.
   c. Click **OK** to activate the local domain.

Related concepts:

"Overview on the UPN and immutableID strategy for Microsoft Office 365" on page 11
You must choose a strategy for ImmutableID before configuring single sign-on settings for Microsoft Office 365. Microsoft Office 365 users are identified by the User Principal Name or UPN and ImmutableID.

Service provider side configuration

Configure the service provider federation settings using the Federation First Steps tool.

Note: The configuration instructions for these service providers may change. See the service provider-specific documentation for updates.

First Steps plug-in for Google Apps

Use the first steps plug-in to create a federation with Google Apps.

1. Select the appropriate configuration for your identity provider. For more information, see "Identity provider side configuration" on page 5.
2. "Configuring Google Apps single sign-on settings"
3. "Provisioning users to Google Apps" on page 10
4. "Testing the single sign-on to Google Apps" on page 10

Configuring Google Apps single sign-on settings

Configure the Google Apps single-sign on settings to enable user authentication.

Before you begin

The configuration requires you to provide a certificate that is used to sign the SAML message in your Federated Identity Manager. Export your Federated Identity Manager certificate into a Privacy-Enhanced Message (PEM) format. See "Exporting a certificate" in the *IBM Tivoli Federated Identity Manager Configuration Guide*.

Procedure

1. Navigate to the website of your service provider.
   a. Open a web browser.
   b. Enter the URL provided by Google to access your account. For example, `https://www.google.com/a/example.com`.
2. Log in with your credentials.
3. Navigate to the single sign-on configuration page.
   a. Click **Advanced Tools**.
   b. Select **Set up single sign-on (SSO)**.
4. Select the **Enable Single Sign-on** setting.
5. Configure the single sign-on settings by providing the following information:

   Sign-in page URL
   Enter the Federated Identity Manager Login Endpoint URL. For example, `https://idp.example.com/FIM/sps/<federation name>/saml20/login`

   Sign-out page URL
   Enter the URL to redirect users when they log out.

   Change password URL
   Provide the URL to let users change their password in your system.
6. Upload the verification certificate that you exported at the beginning of this task in this field. This certificate must contain the public key from the key-pair that is used for signing SAML messages in your Federated Identity Manager.
7. Save your settings.

What to do next

Test the single sign-on on Google Apps.

Provisioning users to Google Apps

Provision users to Google Apps so they can be authenticated through single sign-on.

Procedure

1. Log in to Google Apps.
2. Click **Organization & users**.
3. Click **Create a new user**.
4. Specify the required information about your user.
5. Follow the on-screen instructions to complete the steps for adding a user.

Results

Users are added in Google Apps.

Testing the single sign-on to Google Apps

Test the single sign-on authentication to Google Apps after completing all the federation and domain configuration steps.

Before you begin

Ensure that you have added users in Google Apps and Federated Identity Manager. For more information on how to add users in Google Apps, see

"Provisioning users to Google Apps" on page 10. For more information about how to add single sign-on users in Federated Identity Manager, see the *IBM Tivoli Federated Identity Manager Configuration Guide*.

About this task

The steps provide instructions on how you can test the single sign-on.

Procedure

Initiate the single sign-on.

- **To initiate single sign-on from the Identity Provider side:**
  1. Navigate to the login-initial endpoint of the Identity Provider.
  2. Indicate `google.com` as the `PartnerId`.
  3. Set the target to any string. For example, `https://idp.example.com/sps/<federationName>/saml20/logininitial?PartnerId=google.com &Target=<anystring>`. You are redirected to the Identity Provider login page.

     Note: If you do not specify a value for Target or set it to an empty string, Google displays an error message stating that the required response parameter **`RelayState`** is missing.
  4. Login with the user ID and password of the user that you are testing.
- **To initiate single sign-on from Google:**
  1. Access the protected resource URL. For example, `https://drive.google.com/a/example.com`. You are redirected to the Identity Provider login page.
  2. Login with the user ID and password of the user that you are testing.

Results

The user can access the protected resource in Google Apps.

First Steps plug-in for Microsoft Office 365

Use the first steps plug-in to create a federation with Microsoft Office 365. The usage of Windows PowerShell for single sign-on to provision users is not supported by IBM®.

1. "Overview on the UPN and immutableID strategy for Microsoft Office 365"
   a. "Populating the Federated Identity Manager alias service" on page 13
   b. "Sending a request to the Tivoli Federated Identity Manager Security Token Service" on page 14
2. Select the appropriate configuration for your identity provider. For more information, see "Identity provider side configuration" on page 5.
3. "Configuring Microsoft Office 365 single sign-on settings" on page 15
4. "Provisioning users to Microsoft Office 365" on page 16
5. "Testing the single sign-on to Microsoft Office 365" on page 16

Overview on the UPN and immutableID strategy for Microsoft Office 365

You must choose a strategy for ImmutableID before configuring single sign-on settings for Microsoft Office 365. Microsoft Office 365 users are identified by the User Principal Name or UPN and ImmutableID.

UPN The UPN is the local account user name that is appended with

`@domainname` for a registered domain you own. The Federation First Steps tool automatically generates the mapping rule that maps the local account user name to the UPN format.

ImmutableID
This identifier is a non-recycled unique identifier for the account. It must be passed as an attribute in the SAML assertion during single sign-on to Microsoft Office 365.

You must determine the ImmutableID value and where it is stored.

Using the Tivoli Access Manager principal UUID as the ImmutableID
This technique applies only if you use Tivoli Access Manager WebSEAL as your point of contact for Federated Identity Manager. It applies during standard user authentication from the Tivoli Access Manager registry.

Tivoli Access Manager assigns a principal UUID to each user account. This UUID is used during user provisioning and during single sign-on with WS-Federation in Federated Identity Manager.

The advantage of using this technique is that during single sign-on at run time, the UUID is already in the Tivoli Access Manager credential. There is no need to write a Federated Identity Manager mapping rule to retrieve it and insert it into the SAML assertion.

The disadvantage of this technique is that there is no way of determining whether a user is provisioned to Microsoft Office 365 before you attempt single sign-on.

User provisioning must be done before a user attempts single sign-on. Otherwise, single sign-on fails. See "Provisioning users to Microsoft Office 365" on page 16 for more details.

The following scenario shows a `ldapsearch` for a Tivoli Access Manager principal UUID. For example, `secUUID`. The Tivoli Access Manager domain is `default`. The user is `jane`. Type the command on one line.

```
/opt/IBM/ldap/V6.1/bin/idsldapsearch -L -h localhost -p 389 -D
cn=root -w <your_ldap_pwd> -b "cn=users,secauthority=default"
-s one "(&(objectclass=secUser)(principalName=jane))"
```

The following code is a sample result:

```
....
secDN: principalName=jane,cn=Users,secAuthority=Default
secUUID: a1a2a3a4-b1b2-c1c2-1111-000000000000
....
```

Using the Federated Identity Manager alias service to manage ImmutableID

This technique requires a configured alias service for Federated Identity Manager. See IBM Tivoli Federated Identity Manager Information Center for more details about alias service.

If Tivoli Access Manager is not your point of contact or if you do not want to use the Tivoli Access Manager principal UUID as the Microsoft Office 365 ImmutableID, store and manage a different UUID.

Federated Identity Manager provides a generic alias service. It is used in SAML 2.0 federations with persistent name identifiers.The supported programmatic interface to the Federated Identity Manager alias service is through the APIs in the **`IDMappingExtUtils`** class. The class can be called from the Javascript and Java™ mapping rules in the Federated Identity Manager Security Token Service.

To populate the alias service, you must create an STS trust chain and call those APIs. See "Populating the Federated Identity Manager alias service."

Using the Base64 encoded value of UPN

With this technique, the ImmutableID that is sent to Microsoft Office 365 is the base64-encoded value of the UPN.

Note: Use this method only in a test environment and not in a production environment.

Populating the Federated Identity Manager alias service:

To use the Federated Identity Manager alias service to manage ImmutableID, populate the Federated Identity Manager alias service first.

Procedure

1. Create an STS chain with the following structure:
   - **Default STSUU** (validate)
   - **Default Map** (map)
   - **Default STSUU** (issue)
2. Set the following addresses:
   - **AppliesTo Address** `http://appliesto/alias`
   - **Issuer Address** `http://issuer/alias`
3. Use the following JavaScript mapping rule for the default map module.

```
// BEGIN Javascript mapping rule
// mapping rule for performing simple store and fetch alias operations
importPackage(Packages.com.tivoli.am.fim.trustserver.sts);
importPackage(Packages.com.tivoli.am.fim.trustserver.sts.utilities);
importPackage(Packages.com.tivoli.am.fim.trustserver.sts.uuser);

// Figure out the "operation" being performed
var operation = stsuu.getAttributeValueByName("operation");

// store operation
if (operation != null && operation == "store") {
 // get username, federation ID, and alias value to store
 var username = stsuu.getPrincipalName();
 var federationid = stsuu.getAttributeValueByName("federationID");
 var alias = stsuu.getAttributeValueByName("alias");

 if (username != null && federationid != null && alias != null) {
  // store it, first removing any existing values
  var existingAliasValues =
    IDMappingExtUtils.lookupAliasesForUserAsStringArray(
      federationid, username);
  if (existingAliasValues != null) {
   for (var i = 0; i < existingAliasValues.length; i++) {
    IDMappingExtUtils.removeAliasForUser(
      federationid, username, existingAliasValues[i]);
   }
  }
  IDMappingExtUtils.addAliasForUser(federationid, username, alias);
 }
}

// fetch operation
if (operation != null && operation == "fetch") {
 // get username, federation ID
 var username = stsuu.getPrincipalName();
```

```
 var federationid = stsuu.getAttributeValueByName("federationID");

 if (username != null && federationid != null) {
  // fetch alias value(s) and put in STSUU
  var existingAliasValues =
    IDMappingExtUtils.lookupAliasesForUserAsStringArray(
      federationid, username);
  var attr = new Attribute("alisValues", "", existingAliasValues);
  stsuu.addAttribute(attr);
 }
}

// delete operation
if (operation != null && operation == "delete") {
 // get username, federation ID
 var username = stsuu.getPrincipalName();
 var federationid = stsuu.getAttributeValueByName("federationID");

 if (username != null && federationid != null) {
  // remove any existing values
  var existingAliasValues =
    IDMappingExtUtils.lookupAliasesForUserAsStringArray(
      federationid, username);
  if (existingAliasValues != null) {
   for (var i = 0; i < existingAliasValues.length; i++) {
    IDMappingExtUtils.removeAliasForUser(
      federationid, username, existingAliasValues[i]);
   }
  }
 }
}
// END Javascript mapping rule
```

4. Provision an alias for a user. See "Sending a request to the Tivoli Federated Identity Manager Security Token Service."

Sending a request to the Tivoli Federated Identity Manager Security Token Service:

Use the `curl` utility to send requests to the Tivoli Federated Identity Manager Security Token Service to provision a user alias.

Procedure

1. Save the following `RequestSecurityToken` 1.2 message in a file. For example, `rst.xml`.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
    <wst:RequestSecurityToken xmlns:wst=
"http://schemas.xmlsoap.org/ws/2005/02/trust">
      <wst:RequestType xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust
">http://schemas.xmlsoap.org/ws/2005/02/trust/Validate</wst:RequestType>
      <wst:Issuer xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
        <wsa:Address xmlns:wsa=
"http://schemas.xmlsoap.org/ws/2004/08/addressing">
http://issuer/alias</wsa:Address>
      </wst:Issuer>
      <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
        <wsa:EndpointReference xmlns:wsa=
"http://schemas.xmlsoap.org/ws/2004/08/addressing">
          <wsa:Address>http://appliesto/alias</wsa:Address>
        </wsa:EndpointReference>
```

```
      </wsp:AppliesTo>
      <wst:Base xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
<stsuuser:STSUniversalUser
        xmlns:stsuuser="urn:ibm:names:ITFIM:1.0:stsuuser">
        <stsuuser:Principal>
                <stsuuser:Attribute name="name"
                        type="urn:ibm:names:ITFIM:5.1:accessmanager">
                        <stsuuser:Value>jane</stsuuser:Value>
                </stsuuser:Attribute>
        </stsuuser:Principal>
        <stsuuser:AttributeList>
                <stsuuser:Attribute name="operation">
                        <stsuuser:Value>store</stsuuser:Value>
                </stsuuser:Attribute>
                <stsuuser:Attribute name="federationID">
                        <stsuuser:Value>urn:federation:
MicrosoftOnline</stsuuser:Value>
                </stsuuser:Attribute>
                <stsuuser:Attribute name="alias">
                        <stsuuser:Value>myalias</stsuuser:Value>
                </stsuuser:Attribute>
        </stsuuser:AttributeList>
</stsuuser:STSUniversalUser>
      </wst:Base>
    </wst:RequestSecurityToken>
  </soapenv:Body>
</soapenv:Envelope>
```

2. Replace your own values for `username` and `alias`, which is the `ImmutableID` attribute for each user you want to provision. In the example, the `username` is `jane` and the `alias` is `myalias`.
3. Use `curl` utility to send the request to the WS-Trust 1.2 endpoint. For example:

```
curl --header "soapaction: blah"
--header "Content-Type: text/xml" \
    --data-binary @rst.xml http://localhost:9080
/TrustServer/SecurityTokenService
```

Configuring Microsoft Office 365 single sign-on settings

Configuring Microsoft Office 365 single sign-on settings involves installing the Windows PowerShell for Single sign-on, running the command-line tool, and confirming the activation of your federated domain.

Procedure

1. Install Windows PowerShell for single sign-on. See Microsoft Online and search for *Windows PowerShell for single sign-on* for more details.
2. Run the Windows PowerShell for single sign-on and create a federated domain.
   a. Log in to MS Online Service by typing the following command:
      ```
      Connect-MsolService
      ```
   b. Create a new federated domain by typing the following command:
      ```
      New-MsolDomain -authentication federated -DomainName example.com
      ```
   c. Get the domain verification DNS by typing the following command:
      ```
      Get-MsolDomainVerificationDns -DomainName example.com -Mode DnsTxtRecord
      ```
   d. Log in to the domain registrar where you own the domain.
   e. Add the `mx entry`.
   f. Confirm the domain and fill in the federation properties.

      In the following example, the federation name is `office365` and the WebSEAL junction is `FIM`.

```
Confirm-MsolDomain \
  -DomainName example.com \
  -FederationBrandName "Example, Inc" \
  -IssuerUri https://identityprovider.example.com/FIM/sps/office365/wsf \
  -LogOffUri https://profile.example.com/FIM/sps/office365/wsf \
  -PassiveLogOnUri https://profile.example.com/FIM/sps/office365/wsf  \
  -PreferredAuthenticationProtocol WsFed \
  -SigningCertificate MIICBz........Q==
```

Note: This operation fails if Microsoft Office 365 cannot resolve the `mx entry`. Allow time for the DNS propagation to be complete.

3. Confirm the activation of your federated domain.
   a. Go to `https://portal.microsoftonline.com`.
   b. Log in with your Admin account.
   c. Navigate to **Admin Overview** > **Management** > **Domains**.
   d. Check whether your domain is in the domains list and whether that the status is **Verified**.

Provisioning users to Microsoft Office 365

Provision users to Microsoft Office 365 so they can be authenticated through single sign-on. Using Windows PowerShell for single sign-on to provision users is not supported by IBM.

Procedure

1. Log in to MS Online Service by typing the following command:

   ```
   Connect-MsolService
   ```

2. Provision users with the **New-MsolUser** command. Enter the command on one line.

   ```
   New-MsolUser -userprincipalname $upn \
                -immutableID $base64 \
       -lastname $sn \
       -firstname $gn \
       -Displayname $displayName \
       -BlockCredential $false \
       -LicenseAssignment $license \
       -usageLocation $usagelocation
   ```

   Note: Use the **Get-MsolAccountSku** command to know what values you can set for **LicenseAssignment**. See Windows PowerShell cmdlets for Microsoft Office 365 for more details.

Testing the single sign-on to Microsoft Office 365

Test the single sign-on authentication to Microsoft Office 365 after you complete the federation and domain configuration steps to verify that it works correctly.

Before you begin

Ensure that you have added users in Microsoft Office 365 and Federated Identity Manager. For more information on how to add users in Microsoft Office 365, see "Provisioning users to Microsoft Office 365." For more information about how to add single sign-on users in Federated Identity Manager, see the *IBM Tivoli Federated Identity Manager Configuration Guide*.

Procedure

1. Make sure that there are no existing browser sessions in Identity Provider and Microsoft Office 365.

2. Go to `https://portal.microsoftonline.com/`.
3. Enter your provisioned user ID under the federated domain in the **User ID** field. For example, `john@example.com`. The screen changes to indicate that you do not need to provide a password.
4. Click **Sign in at <domain name>**. You are redirected to the login page of your Identity Provider.
5. Log in with your credentials.

Results

You are redirected and logged in to Microsoft Office 365.

First Steps plug-in for Salesforce

Use the first steps plug-in to create a federation with Salesforce.

1. Select the appropriate configuration for your identity provider. For more information, see Identity provider side configuration.
2. "Configuring Salesforce single sign-on settings"
3. "Testing the single sign-on to Salesforce" on page 18

Configuring Salesforce single sign-on settings

Configure the Salesforce single-sign on settings to enable user authentication.

Before you begin

The configuration requires you to provide a certificate that is used to sign the SAML message in your Federated Identity Manager. Export your Federated Identity Manager certificate into a Privacy-Enhanced Message (PEM) format. See "Exporting a certificate" in the *IBM Tivoli Federated Identity Manager Configuration Guide*.

Procedure

1. Navigate to the website of your service provider.
   a. Open a web browser.
   b. Enter the URL provided by Salesforce to access your account. For example, `https://www.salesforce.com`.
2. Log in with your credentials.
3. Navigate to the single sign-on configuration page.
   a. Click your account name to reveal the user menu.
   b. Select **Setup**.
4. Configure the single sign-on settings.
   a. Under **Adminstration Setup**, select **Security Controls** > **Single Sign-On Settings**.
   b. Click **Edit**.
   c. Check the **SAML Enabled** option.
5. Provide the following information:

   SAML Enabled
   You must enable this option.

   SAML Version
   The plug-in supports SAML 1.1.

   Issuer This option is the Federated Identity Manager Endpoint URL. For

example, `https://idp.example.com/sps/<federation name>/saml11`. This value must match the `Issuer` in the SAML assertion. The `Issuer` value comes from the Provider ID value that is defined in the identity provider SAML 1.1 federation settings.

Identity Provider Certificate
This option is the Federated Identity Manager Certificate that you previously exported at the beginning of this task. It is the public certificate of the key for signing the SAML messages. It must be an uploaded PEM-encoded certificate file which contains the public key that matches the signing certificate at the Federated Identity Manager identity provider.

SAML user ID Type
Select the first option if you want to use the identities on your identity provider website. If not, then select **Assertion contains the Federation ID from the User object**. You must enter a Federation ID for each user in the **Manage Users Menu**.

SAML user ID Location
The User ID Location is in the `NameIdentifier` element of the Subject statement. Select the second option if you want to use advanced user mapping scenarios. You can also select the second option if you want the value to be read from a nominated attribute in the `AttributeStatement` of the SAML assertion.

6. Save your settings.

Testing the single sign-on to Salesforce

Test the single sign-on authentication to Salesforce after all the federation and domain configuration steps are completed.

Before you begin

Ensure that you have added users in Salesforce and Federated Identity Manager. For more information on how to add users in Salesforce, see the Salesforce documentation or consult your Salesforce system administrator. For more information about how to add single sign-on users in Federated Identity Manager, see the *IBM Tivoli Federated Identity Manager Configuration Guide*.

Procedure

1. Close any Identity Provider and Salesforce browser sessions.
2. Navigate to the login initial endpoint of the Identity Provider. You are redirected to the Identity Provider login page. For example, `https://idp.example.com/FIM/sps/<fed name>/saml11/login?TARGET=https://saml.salesforce.com`
3. Enter your credentials.

Results

You are redirected and logged in to Salesforce.

First Steps plug-in for Workday

Use the first steps plug-in to create a federation with Workday.

1. Select the appropriate configuration for your identity provider. For more information, see Identity provider side configuration.
2. "Configuring Workday single sign-on settings" on page 19

3. “Testing the single sign-on to Workday”

Configuring Workday single sign-on settings

Configure the Workday security configuration to enable single sign-on.

Before you begin

The configuration requires you to provide a certificate to sign the SAML message in your Federated Identity Manager. Export your Federated Identity Manager certificate into a Privacy-Enhanced Message (PEM) format. See "Exporting a certificate" in the *IBM Tivoli Federated Identity Manager Configuration Guide*.

Procedure

1. Navigate to the website of your service provider.
   a. Open a web browser.
   b. Enter the URL provided by Workday to access your account. For example, `https://www.myworkday.com/<your company>/login.flex`.
2. Log in with your Admin account.
3. Navigate to the single sign-on configuration page.
   a. Click **Workbench** > **Account Administration** > **Edit Tenant Setup - Security**
4. Configure the single sign-on settings by providing the following information:
   a. Under **SAML Setup**, select the **Enable SAML Authentication** option.
   b. Specify the following information:

      Identity Provider ID
      Enter the Federated Identity Manager Login Endpoint URL. For example, `https://idp.example.com/FIM/sps/<federation name>/saml20/login`

      x509 Public Key
      Upload the certificate that you exported at the beginning of this task in this field. This certificate must contain the public key from the key-pair that is used for signing SAML messages in your Federated Identity Manager.
5. Save your settings.

What to do next

Test the single sign-on on Workday.

Testing the single sign-on to Workday

Test the single sign-on authentication to Workday after you complete all the federation and domain configuration steps.

Before you begin

Ensure that you have added users in Workday and Federated Identity Manager. For more information on how to add users in Workday, see the Workday documentation or consult your Workday system administrator. For more information about how to add single sign-on users in Federated Identity Manager, see *IBM Tivoli Federated Identity Manager Configuration Guide*.

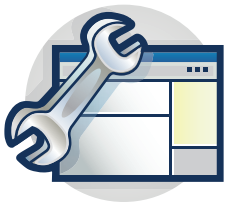
Procedure

1. Initiate the single sign-on by navigating to the identity provider login endpoint. For example, `https://idp.example.com/FIM/sps/<fed name>/saml20/logininitial?PartnerId=http%3A%2F%2Fwww.workday.com`. You are redirected to the Identity Provider login page.
2. Log in with your credentials.

Results

The user can access the protected resource in Workday.

Part 2. Configuration of a domain



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

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

Start with the topic:

- Chapter 3, “Domain configuration,” on page 23

Chapter 3. Domain configuration

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

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

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

When Tivoli Federated Identity Manager is installed, no domains exist. Use the management console to create a domain. When you installed Tivoli Federated Identity Manager, the management service was deployed to a WebSphere server (single server mode) or WebSphere Deployment Manager (WebSphere cluster mode).

Connect with the management service and choose a WebSphere server or cluster to which you must deploy the Tivoli Federated Identity Manager runtime component. When the runtime is deployed and configured, you are ready to configure additional features such as federated single sign-on or Web services security management.

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

The Tivoli Federated Identity Manager management service uses the application deployment services of the WebSphere Deployment Manager to deploy and configure the runtime application to distributed cluster members.

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

Domain management service endpoints properties

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

```
idp.example.com
```

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

WebSphere global security properties

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

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

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

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

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

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

On Windows:

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

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

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

On Windows:

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

SSL Trusted Keystore password
The password that is required to access the SSL trusted keystore file.

The default password for the WebSphere key is:

```
WebAS
```

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

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

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

WebSphere server or cluster name

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

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

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

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

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

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

Tivoli Access Manager environment properties

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

The wizard presents the following prompt:

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

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

Administrator Username
The Tivoli Access Manager administrator. The default ID is sec_master. If you chose another administrator ID when you installed Tivoli Access Manager enter the administrator ID in the **Administrator Username** field.

Administrator Password
The password for the Tivoli Access Manager administrator.

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

```
idp.example.com
```

Port The port number used to communicate with the policy server.

This number matches the port number that you specified when you configured Tivoli Access Manager. The default value is `7135`.

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

```
idp.example.com
```

Port The port number used to communicate with the authorization server.

This number matches the port number that you specified when you configured Tivoli Access Manager. The default value is `7136`.

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

Worksheet for domain configuration

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

The properties on this worksheet are described in Chapter 3, "Domain configuration," on page 23.

Table 1. Domain configuration properties

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

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

Table 2. Tivoli Access Manager environment properties

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

Creating and deploying a new domain

You must create a domain and deploy a runtime application for each instance of the Tivoli Federated Identity Manager.

Before you begin

Note: IBM deprecated the Tivoli Federated Identity Manager Security Token Service (STS) Client in this release.

If you use WebSphere 6.X, you can still use the Tivoli Federated Identity Manager Security STS client while Tivoli Federated Identity Manager supports WebSphere 6.X. When Tivoli Federated Identity Manager discontinues its support for WebSphere 6.X, use WebSphere Application Server version 7 Update 11 and later. See WS-Trust client API and WS-Trust Clients for details.

A wizard prompts you to supply the necessary configuration properties. You can use the properties on the worksheet that you prepared. For more information about the worksheet, see Chapter 3, "Domain configuration," on page 23

About this task

This task is a prerequisite to configure additional Tivoli Federated Identity Manager features such as federated single sign-on or Web Services Security Management. It is also a prerequisite for deployments that use the Tivoli Federated Identity Manager security token service for token exchange.

An example of a token exchange scenario is deployment of Tivoli Federated Identity Manager Kerberos constrained delegation with WebSEAL junctions.

Procedure

1. Verify that the WebSphere Application Server application is running.
2. Copy all the WebSphere key files from the Deployment Manager to all the nodes in the cluster under the following circumstances:
   - When you deploy a domain into a WebSphere Application Server cluster,
   - When the WebSphere global security is enabled

   Place the keys on each node in the same directory as on the Deployment Manager. WebSphere 6.1 does this process automatically. However, ensure that when the administration console is remote from the DMgr (Management Service), the server certificate presented by the DMgr is trusted by the console. One way to do this verification is to copy the truststore from the DMgr to the console profile.
3. Log on to the WebSphere console.
4. Click **Tivoli Federated Identity Manager → Getting Started**.

   The Getting Started portlet opens.
5. Click **Manage Domains.** The Domains portlet opens.
6. Click **Create**. The Domain wizard opens the Welcome panel.
7. Click **Next**. The Management Service Endpoint panel opens.
8. Enter values for the specified properties.
9. Click **Next**. The WebSphere Security panel opens.
10. 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**.
11. Click **Test Connection**. When successful, you can see an information message:

```
FBTCON317I Tivoli Federated Identity Manager connected successfully.
```

12. Click **Next**. The WebSphere Target Mapping panel opens.
13. Select or enter the name of your server or cluster.
14. When finished, click **Next**.
    - When the WebSphere environment consists of a single server, the panel shows a Server name menu with a default name.
    - When the WebSphere environment consists of a cluster, the panel shows the Cluster Name menu. This menu lists the names of clusters defined in the cell. Select the name of the cluster to use.

    The Select Domain panel opens. A default name is provided.
15. Accept default name or enter a name for the new domain. The Tivoli Access Manager Environment Settings panel opens.
16. Select or clear **This Environment Uses Tivoli Access Manager** as appropriate.
17. Click **Next**. When you select this option, provide values for the rest of the properties. The Summary panel opens
18. Verify that the domain information is correct.
19. Click **Finish**.

    The domain is created and the domain wizard exits. The Create Domain Complete panel opens.
20. Select both of the check boxes on the Create Domain Complete panel.
21. Click **OK**.

    You must complete both of the tasks as part of the initial creation and deployment of the Tivoli Federated Identity Manager management service and runtime:
    - **Make this domain the active management domain**
    - **Open Runtime Node Management upon completion**
22. When you are deploying Tivoli Federated Identity Manager into a WebSphere cluster, ensure that the WebSphere Node Agent is running on all the nodes in the cluster.

    Use the WebSphere administrative to verify the status of the node agents.

    The Current Domain portlet and the Runtime Node Management portlet open.
23. In the Runtime Node Management portlet, click **Deploy Runtime**. A message shows:

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

    The following link shows:

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

    The Deployment operation might take several minutes. During this time, you can click the link to check for completion. When the deployment is complete, then click the link to return to the message:

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

    The Runtime Node Management portlet is redrawn. An entry for the runtime is added to the **Runtime Nodes** table for each node in the domain. The **Configure** button is also activated.
24. In the Runtime Node table, select the check box for your node.
25. Click **Configure**.

    The runtime application is configured into the environment.

26. In a WebSphere cluster environment, configure each node in the cluster by repeating the previous step.
27. When all nodes are configured, click the **Load configuration changes to the Tivoli Federated Identity Runtime** button.

    The button is located in the Current Domain portlet.
28. Continue with the instructions that apply to your deployment:
    - In a WebSphere *cluster* environment, continue with "Mapping the runtime to a Web server."
    - In a WebSphere *non-clustered* (stand-alone server) environment, the domain creation, and deployment is now complete. Continue with the appropriate instructions for your scenario.

What to do next

Restart the WebSphere Application Server under the following circumstances:

- If you specified inaccurate information in the WebSphere Security panel in the Domain wizard
- While creating a Tivoli Federated Identity Manager domain, or a connection to a domain

If you attempted to correct the information and you still cannot connect to the Tivoli Federated Identity Manager console, restart the WebSphere Application Server.

Use **Test Connection** in the panel to verify the connection between the Tivoli Federated Identity Manager console and the Management Service.

Mapping the runtime to a Web server

Learn how to map the Tivoli Federated Identity Manager runtime to the IBM HTTP Web server for cluster environments.

About this task

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

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

Procedure

1. Log on to the WebSphere administrative console:

   `http://your_host_name:9060/admin`
2. Select **Enterprise Applications -> ITFIM Runtime**. The Configuration tab shows.
3. In the Web Module Properties section, select the **Virtual hosts** link. A section titled Apply Multiple Mappings shows a table with a row entry for each Web module.

4. Select the check box for each web module. Ensure that all check boxes are selected.
5. Accept the default entry of `default_host` in the Virtual host field for each Web module. A message box prompts you to save your changes.
6. Click the **Save** link. The Save panel opens.
7. Click **Save**.
8. Return to **Enterprise Applications -> ITFIM Runtime**. The Configuration tab opens.
9. In the Modules section, select the **Manage Modules** link. The **Enterprise Application -> ITFIM Runtime -> Manage Modules** page opens. At the top, you can see the title Manage Modules.
10. Select the check box for *each* of the web modules. For Tivoli Federated Identity Manager the list of modules can include, but is not limited to:
    - ITFIM-Runtime
    - ITFIM Security Token Service
    - ITFIM Information Service
    - TokenService
    - TrustServerWST13

    Note: IBM deprecated the Tivoli Federated Identity Manager Security Token Service (STS) Client in this release.

    If you use WebSphere 6.X, you can still use the Tivoli Federated Identity Manager Security STS client while Tivoli Federated Identity Manager supports WebSphere 6.X. When Tivoli Federated Identity Manager discontinues its support for WebSphere 6.X, use WebSphere Application Server version 7 Update 11 and later. See WS-Trust client API and WS-Trust Clients for details.
11. A scrolling window lists Clusters and Servers. Select both of the following entries:
    - The entry for your cluster. For example, `cluster=fimCluster`.
    - The entry for the web server. For example, `server=webserver1`
12. While both items are highlighted, click **Apply**. In the Module table, the definition for each server and cluster is added to the entry, in the Server column, for each of the web modules that you selected.
13. Click **OK** at the bottom of the page. A message prompts you to save your changes.
14. Click the **Save** link. The **Enterprise Applications → Save** panel opens.
15. Click **Save**.
16. To finish configuring the Tivoli Federated Identity Manager runtime into a WebSphere cluster, continue with "Enabling replication in a WebSphere cluster."

Enabling replication in a WebSphere cluster

Learn how to enable cache replication in a cluster environment to enhance the Tivoli Federated Identity Manager runtime application.

About this task

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

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

When the Tivoli Federated Identity Manager runtime is deployed, some WebSphere configuration steps are automatically performed:

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

Additional configuration is required.

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

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

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

Procedure

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

     Select the name of the cluster into which you have deployed the runtime application
   - Replication type: **Both push and pull**
   - Push frequency: **0**
4. Click **OK**. When prompted to save your changes, select the **Save** link. When the next page opens, click **Save**.
5. In the WebSphere administrative console, select **Servers -> Application Servers -> *your_server_name***.

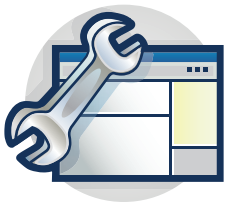
   Note: The properties in this section might already be defined.
6. In the **Container Settings** section, select **Session management**. The Configuration tab shows.
7. In the Additional Properties section, select **Distributed environment settings**. The General Properties panel is redrawn.

8. Examine the **Distributed environment settings** section.
   a. Select **Memory-to-memory replication**.
   b. Click the **Memory-to-memory replication** hyperlink.

   The General Properties panel opens.
9. Specify your replication settings in the General Properties panel:
   a. Set the Replication domain to the name of the cluster into which you have deployed the Tivoli Federated Identity Manager runtime application.
   b. Set Replication mode to **Both client and server**.
10. When prompted to save your changes, select the **Save** link. When the next page opens, click **Save**.
11. From the Server Cluster panel, select the check box for your cluster, and click **Ripplestart**.

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

Part 3. Configuration of a single sign-on federation



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

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

Complete the configuration tasks in the following order:

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

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

   a. Chapter 5, "Identity provider and service provider roles," on page 37
   b. Chapter 6, "Using keys and certificates to secure communications," on page 39
   c. Chapter 7, "Configuring LTPA and its keys," on page 47
   d. Chapter 8, "Setting up message security," on page 49
   e. Chapter 9, "Setting up transport security," on page 71
   f. Chapter 10, "Selecting a point of contact server," on page 83
   g. Chapter 11, "Configuring WebSphere as point of contact server," on page 87
   h. Chapter 12, "Configuring a Web server plug-in," on page 117
   i. Chapter 13, "Setting up the alias service database," on page 131
   j. Chapter 14, "Planning the mapping of user identities," on page 143
2. Complete the instructions for your federation type:
   - Chapter 15, "SAML federations overview," on page 165
   - Chapter 21, "Planning an Information Card federation," on page 271
   - Chapter 24, "OpenID planning overview," on page 313
   - Chapter 30, "Planning a Liberty federation," on page 449
   - Chapter 32, "Planning a WS-Federation single sign-on federation," on page 479

Chapter 4. Overview of configuration tasks for federated single sign-on

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

A federation is a group of two or more trusted IBM Business Partners who want to initiate or receive the transfer of user identities within the federation. The integrity of the identity is based on existing trust relationships among members of the federation, often codified by a legal agreement. A user of a company that participated in a federation through federated single sign-on can securely access the resources of their federated IBM Business Partner. Resource access is typically done with a web browser.

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

- Open standards for single sign-on
- Integration with the single sign-on capabilities of IBM WebSphere Application Server 6.1, eliminating the need for authentication by individual applications
- Support for an unlimited number of federations and the ability to tailor unique configurations for each federation.

  For example, you can assume either an identity provider role or a service provider role in any federation with only one installation of Tivoli Federated Identity Manager.
- Integration support for web applications running on any of the following types of servers:
  - WebSphere Application Server version 6.1 and later
  - Microsoft Internet Information Server (IIS)
  - IBM HTTP Server (IHS)
  - Apache version 2.0 or 2.2 HTTP server
- Simplified Web-based administration

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

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

Note: You must create a domain before deploying a single sign-on federation. If you have not yet deployed a domain, complete the instructions in Chapter 3, “Domain configuration,” on page 23.

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

1. Chapter 5, “Identity provider and service provider roles,” on page 37
2. Chapter 6, “Using keys and certificates to secure communications,” on page 39
3. Chapter 7, “Configuring LTPA and its keys,” on page 47

4. Chapter 8, “Setting up message security,” on page 49
5. Chapter 9, “Setting up transport security,” on page 71
6. Chapter 10, “Selecting a point of contact server,” on page 83
7. Chapter 11, “Configuring WebSphere as point of contact server,” on page 87
8. Chapter 12, “Configuring a Web server plug-in,” on page 117
9. Chapter 13, “Setting up the alias service database,” on page 131
10. Chapter 14, “Planning the mapping of user identities,” on page 143

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

Chapter 5. Identity provider and service provider roles

Each partner in a federation has a role. The role is either **Identity Provider** or **Service Provider**. An identity provider is a federation partner that vouches for the identity of a user. A service provider is a federation partner that provides services to the user.

- **Identity provider**

  The Identity Provider authenticates the user and provides an *authentication token* (that is, information that verifies the authenticity of the user) to the service provider.

  The identity provider does either of the following authentication:

  – Direct user authentication. For example, validating a user name and password.

  – Indirect user authentication. For example, validating an assertion about the user identity as presented by a separate identity provider.

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

- **Service Provider**

  Typically, service providers do not authenticate users but instead request authentication decisions from an identity provider. Service providers rely on identity providers to assert the identity of a user and certain attributes about the user that are managed by the identity provider.

  Service providers might also maintain a local account for the user, along with attributes that are unique to their service.

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

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

  – **Relying party**

    The Information Card protocol specification uses the term Relying Party to describe the service provider role. Select the Service Provider role for your Relying Party when you configure the Information Card federation in the Tivoli Federated Identity Manager wizard.

  – **Consumer**

    The OpenID protocol specification uses the term Consumer to describe the service provider role. Select the Service Provider role for your Consumer when you configure the OpenID in the Tivoli Federated Identity Manager wizard.

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

Chapter 6. Using keys and certificates to secure communications

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

For example, the SAML standards state that the partners must use a Public Key Infrastructure (PKI) and implement Secure Sockets Layer (SSL)-over-HTTP or HTTPS, to establish a trust relationship. Doing so ensures the integrity and confidentiality of the messages during transport.

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

Message-level security

To secure the content of messages and assertions, the SAML standards specify the use of a public key cryptography. By using this method, the federation partners exchange public/private key pairs, and use the keys to sign, encrypt, validate, and decrypt messages and the assertions within the messages. The message signing, encryption, validation, and decrypting process are required in the SAML standard or as appropriate to their environment.

When you configure a federation in Tivoli Federated Identity Manager, the federation configuration wizard prompts you with either signing, validation, or encryption *requirements* or signing, validation, or encryption *options* based on your SAML protocol and profile or binding selections.

For example, if the choices you make when configuring your federation indicate that a signature is required, the wizard prompts you for a signing key. If your selections result in an option to sign or not, the wizard prompts you to make a selection.

Before you use the federation configuration wizard, you must have created the appropriate keys. The information in Chapter 8, “Setting up message security,” on page 49 helps you plan which keys you need in your environment, and provides instructions for creating or obtaining them.

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

Signing

XML messages and SAML assertions are signed by one partner to protect the integrity of the message. The signature enables the receiving party to check if the message had been changed or modified during transmission.

Signing is done with a private key. The partner who receives the signed XML message or SAML assertion needs the X.509 certificate (public key) that corresponds to the private key of the message signer. By default, the X.509 certificate (public key) is included with the signature as a base64-encoded X.509 certificate. However, you have the option of specifying which certificate data you want to include with your signatures.

Validation

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

Encryption and decryption

In SAML 2.0, messages can be encrypted in addition to being signed. The use of the public/private key pair during encryption and decryption differs from its use during signing and validation. The *public key of the intended recipient* is used for encryption. For one partner to encrypt a message, that partner must have the public key of the partner to whom the message is being sent.

The partner who receives the encrypted message must use its *private* key to decrypt the message. In Tivoli Federated Identity Manager, when SAML 2.0 is used, both partners must obtain their own public/private key pairs to be used for encryption. They must then exchange their public keys so that each partner can encrypt messages to the other.

Transport-level security

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

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

Note: SSL is a complex topic. This overview serves only as an introduction to familiarize you with the basic concepts and terminology used in this book.

Server authentication

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

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

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

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

To enable server authentication, you must create a public/private key pair and obtain a certificate. Your server uses certificate to authenticate itself to the client. The certificate is also known as a *server certificate*, or a *personal certificate*.

Although you can create your own server certificate with a software that supports certificate creation in a production environment, you might want to obtain a server certificate from a third-party. The server certificate is also known as a *certificate authority* or *CA* that issues certificates.

Before attempting an SSL connection, the client that the server presents its certificate to must obtain the certificate of the CA that issued the server certificate. The client maintains a list of trusted issuers and adds the CA certificate to that list.

The server certificate contains the following information:

- the server certificate public key
- the certificate serial number
- the certificate validity period
- the distinguished name of the server which includes the host name associated with the server
- the distinguished name of the issuer
- the digital signature of the issuer

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

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

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

Table 3. SSL server authentication certificate requirements

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

Instructions for enabling SSL are described in “Enabling SSL on the WebSphere Application Server” on page 71.

Client authentication

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

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

Client certificate authentication
A *client certificate* is similar to a server certificate. To obtain a client certificate, the client typically requests it from a CA. Before establishing a federation, the partners typically agree on a CA to use for signing the client certificate. The server must ensure that CA is in its list of trusted issuers.

When client certificate authentication is configured, the server requests authentication from the client. The client responds by sending its client certificate and its digital signature in a randomly generated piece of data to the server. The client certificate typically includes the following items:

- public key of the client
- serial number of the certificate
- validity period of the certificate
- distinguished name of the client
- distinguished name of the issuer
- digital signature of the issuer

The server verifies the client information. For example, the client exports its certificate and provides it to the server partner. Then, the server uses the public key of the client in the client certificate to do the following tasks:

- validate the digital signature of the client
- checks its list of trusted issuers or certificate authorities
- to see if the client certificate issuer is trusted
- compare the digital signature of the issuer in the client certificate to the digital signature of the issuer in the CA certificate

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

Table 4. SSL client authentication certificate requirements

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

Partners who act as *servers* must follow instructions for configuring a client authentication requirement on their servers, "Configuring client authentication requirements" on page 76.

Partners who act as *clients whose partners require client certificate authentication* must follow instructions for configuring their client certificates, "Configuring your client certificates" on page 80.

Storage and management of keys and certificates

Keys and certificates are stored in keystores and truststores.

Keystores
Private keys and personal certificates are stored in keystores.

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

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

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

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

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

- SSL client certificates and their private keys

  Certificates used for client certificate authentication. The private keys are those that are in the client keystore
- CA certificates for servers that have SSL server authentication configured

  Pertains to the certificates in the truststore of the client
- Signing keys, validation keys, and encryption keys are also managed in the keystores and truststores in Tivoli Federated Identity Manager. For example:

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

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

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

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

WebSphere Application Server

During profile creation, WebSphere Application Server creates:

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

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

Tivoli Federated Identity Manager

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

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

The default password for the keystores is `testonly`.

The default keys and certificates are for test purposes only. You must create new keys and certificates, and you might also want to create new keystores when you configure Tivoli Federated Identity Manager.

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

Creation of keystores, keys, and certificates

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

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

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

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

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

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

Message-level security instructions:
Chapter 8, "Setting up message security," on page 49.

Transport-level security instructions:
Chapter 9, "Setting up transport security," on page 71

Key selection criteria

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

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

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

You can use the custom runtime property `key.selection.criteria` to configure the order of certificates or keys. Use these values for the custom property:

only.alias
Alias only: The selected key only, without Auto rollover. If the key is invalid, the software indicates failure. Configure the property to use the value.

shortest.lifetime
Shortest lifetime: For signing, a valid key with the shortest available lifetime. For validation, key lifetime availability runs from shortest to longest.

longest.lifetime
Longest Lifetime: For signing, a valid key with the longest available lifetime. For validation, key lifetime availability runs from longest to shortest.

Chapter 7. Configuring LTPA and its keys

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

About this task

The default LTPA configuration is as follows:

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

Key sets
The default key sets are NodeLTPAKeyPair and NodeLTPASecret.

Key generation
By default, LTPA keys are automatically generated the first time you start the server after installation. LTPA 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. A separate target application server can be a separate WebSphere Application Server, or a server that is supported by the Tivoli Federated Identity Manager Web server plug-in.

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.

For more information about exporting keys from your WebSphere Application Server and importing them to your application server, see "Exporting LTPA key from the point of contact server" on page 115, and either "Importing the LTPA key to the WebSphere Application Server" on page 122, or "Copying the LTPA key to the Web server" on page 125 .

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

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

To review or modify these settings, complete the steps in this procedure:

Procedure

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

   Secure administration, applications, and infrastructure panel opens.
3. On the left, 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 as appropriate, and then click **Apply**.
5. Return to the previous Configuration tab.
6. 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.
7. Click **Apply** when you are done.
8. Save the changes to the master configuration file.

What to do next

Continue with the configuration of your environment. For example, continue with Chapter 8, “Setting up message security,” on page 49.

Chapter 8. Setting up message security

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

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

Note: Instructions for configuring SSL-related certificates, such as server certificates, client certificates, and client authentication requirements are described in Chapter 9, "Setting up transport security," on page 71. The topics in this chapter cover only message-level security, except for the topics related to preparing your keystores.

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

1. Prepare your keystores. See "Preparing the keystores."
2. Discuss message security requirements with your partner and make a list of the keystores and certificates that each of you need. Consider using the checklists in "Planning message-level security" on page 52.
3. Obtain the necessary certificates for your environment. See "Obtaining your keys and certificates" on page 55.
4. Add your certificates into your keystores. See "Adding your certificates to your keystore" on page 58.
5. Obtain any certificates you need from your partner. See "Obtaining a certificate from your partner" on page 60.
6. Provide your partner with any of your certificates that might be needed by that partner. See "Providing certificates to your partner" on page 63.
7. If any of the certificates you use are PKCS#12 files, you must update your Java cryptography policy. See "Updating the cryptography policy" on page 65.
8. If you are setting up a production environment and do not use the default keystores and certificates, remove them so that they are not used unintentionally. See "Removing default keystores" on page 65.

Preparing the keystores

Prepare keystores in the Tivoli Federated Identity Manager key service, regardless of the role that you assume in a federation or the SAML standard that you use. The keystores store keys and certificates that are used to secure the content and transport of messages.

About this task

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

Signing/Encryption keystore
: The keystore is where you store your private keys. Private keys are what you use for signing and decryption. Private keys are also used for your client certificate, if you are a client in an SSL connection with your partner, and that partner requires that you authenticate with a certificate.

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

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

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

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

Changing a keystore password

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

About this task

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

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

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Key Service**.
   The Keystores panel opens.
3. Select a keystore from the Keystore table. The **Change Password** option 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**.

What to do next

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

Creating a keystore

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

About this task

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

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

For details about the keytool utility, see the WebSphere Application Server 8.0 information center http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

What to do next

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

Importing a keystore

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

About this task

Procedure

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

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

   `/tmp/mykeys.jks`

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

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

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

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

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

7. Repeat these steps for each keystore you must create for your certificates and the certificates of your partner.
8. Click **Finish**.

What to do next

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

Planning message-level security

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

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

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

In general, you can sign messages, sign assertions, and validate the signatures of your partner. In a SAML 2.0 federation, each partner must also encrypt the data that they send to each other. Then, each partner must decrypt the data so it can be used in the federation.

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

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

Your keys

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

Table 5. Your keys

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

Keys you need from your partner

Use the *public key* from the public/private key pair of your partner to perform the actions listed. The Notes column indicates if a key is required, or if it cannot be used because of a specific provider role or the SAML specification used by the federation.

In most cases, you can obtain these keys from your partner by way of a metadata file. However, if you are using a SAML 1.x federation, you must obtain these keys manually. Consider sharing this table with your partner to ensure that your partner knows what keys it must provide to you.

Table 6. Keys you need from your partner

Purpose of the key	Alias of public key	Truststore in which to store key	Notes
Validation key for message signatures			Corresponds to the signing key of your partner. Required if your partner signs messages.

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

Purpose of the key	Alias of public key	Truststore in which to store key	Notes
Validation key for assertion signatures			Corresponds to the signing key of your partner. Required if your partner signs assertions. Not available for identity providers using SAML 1.x
Encryption key			Corresponds to the decryption key of your partner. Required in SAML 2.0 Not available in a SAML 1.x federation

Keys you must provide to your partner

Provide your public key from your public/private key pair to your partner so that your partner can perform the actions listed. The Notes column indicates if a key is required or if it cannot be used because of a specific provider role or the SAML specification used by the federation.

In most cases, you must provide these keys by exporting your federation properties into a metadata file that your partner must import into its configuration. However, if you are using a SAML 1.x federation, you must export these keys from your federation and provide them to your partner manually.

Table 7. Keys you must provide to your partner

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

Table 7. Keys you must provide to your partner (continued)

Purpose of the key	Alias of public/private key pair	Keystore in which key is stored	Notes
Encryption key			Corresponds to your decryption key. Required in SAML 2.0 Not available in a SAML 1.x federation

Obtaining your keys and certificates

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

About this task

In general, you must obtain the following private keys:

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

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

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

- In a test environment, you can use the default testkey or create a self-signed certificate. See "Using the default key as your signing and decryption key" or "Creating self-signed certificates" on page 56.
- In a production environment, you would want to request your keys from a certificate authority. See "Requesting CA-signed certificates" on page 56.

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

- PEM

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

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

  Before using PKCS#12 certificates, you must update the cryptography policy. See "Updating the cryptography policy" on page 65.

Using the default key as your signing and decryption key

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

About this task

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

Creating self-signed certificates

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

About this task

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

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

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Key Service**.
   The Keystores panel opens.
3. Select a keystore from the Keystore table. The **View Keys** option is activated.
4. Click **View Keys**. The Password panel opens.
5. Type your keystore password and click **OK**.
6. Click **Create Self-Signed Certificate**. The Create Self-Signed Certificate panel opens.
7. Provide the corresponding entry for each field.
8. Click **OK**. A public/private key pair is added to the keystore.
9. Click **Load configuration changes to Tivoli Federated Identity Manager runtime**.

What to do next

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

Requesting CA-signed certificates

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

Before you begin

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

About this task

A certificate sign request (CSR) is an electronic file that can be sent through using email, 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.

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

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Key Service**.
   The Keystores panel opens.
3. Select a keystore from the Keystore table. The **View Keys** option 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. Enter the appropriate value for each field.
9. Click **OK**. The Generated Certificate Signature Request window opens.
10. Copy and paste the request text into a text file or click **Export Certificate Signature Request** 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 or you might want to use one certificate for all activities. Some activity examples are: signing, decryption, and client authentication.

When you have created all of your certificate sign requests, follow the instructions of your CA for transmitting the request file. Then, continue with the steps for receiving a CA certificate from the CA in "Receiving a signed certificate from a CA" on page 59.

Adding your certificates to your keystore

Before you establish your federation, you must add the keys to sign and decrypt your keystore.

About this task

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

Created a self-signed certificate
If you use the utility in Tivoli Federated Identity Manager to create a self-signed certificate, the certificate is automatically imported into your keystore. If you created a self-signed certificate, but used a utility other than the one provided with Tivoli Federated Identity Manager, import your certificate into the keystore as described in "Importing a certificate."

Requested a signed certificate
If you generated a certificate sign request and sent that request to a CA, you will receive the certificate into your keystore as described in "Receiving a signed certificate from a CA" on page 59.

Importing a certificate

Import a certificate if you received it using a utility or if you manually obtained it from a CA.

About this task

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

- You have used a utility other than the one provided with Tivoli Federated Identity Manager to create a self-signed certificate
- You manually obtained a certificate from a CA

You must also import a certificate that you have received from your partner. For more information about importing partner certificates, see "Importing a certificate from your partner" on page 61.

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, or
- A key from a PKCS#12 file

Note: If you must use a PKCS#12 file, follow the instructions in "Updating the cryptography policy" on page 65.

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

Imported keys are enabled by default.

Procedure

1. Click **Tivoli Federated Identity Manager** > **Key Service**.
   The Keystores panel opens and is activated.
2. Select a keystore from the Keystore table to store your public/private key pair. The **View Keys** option shows and 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 does 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 **Import**. 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. Then click **Next**.
11. Specify a label for the key and, if prompted, select the key to import.
12. Then click **Next**. A summary panel opens.
13. Click **Finish** to exit the wizard.
14. Repeat these steps to import all the keys and certificates you must use in the federation.

What to do next

Next, add the keys of your partner into your truststore. See "Obtaining a certificate from your partner" on page 60.

Receiving a signed certificate from a CA

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

Before you begin

Ensure that you have completed the steps in "Requesting CA-signed certificates" on page 56, and have saved the certificate from the CA to a location that is accessible to the key service.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Key Service**.
   The Keystores panel opens.
3. Select the keystore where the CSR was generated in the Keystore table. The **View Keys** option 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.
9. Then click **OK**. The temporary self-signed certificate in the keystore is replaced with the received signed certificate.
10. Click **Load configuration changes to Tivoli Federated Identity Manager runtime**.

What to do next

Next, add the keys of your partner into your truststore. See "Obtaining a certificate from your partner."

Obtaining a certificate from your partner

Depending on the requirements of your environment, you must obtain certificates from your partner.

Before you begin

Use the worksheet, "Planning message-level security" on page 52, to determine which certificates you might need from your partner. In general, the public keys to obtain from your partner include:

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

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

About this task

Typically in a SAML 2. 0 federation, you receive the validation and encryption keys of your partner in a metadata file from your partner. This process is further

explained in “Importing certificates from your partner's metadata file.” In addition to the keys, other information from your partner, such as company name, is included in the metadata file.

When you create your partner in the federation, you are prompted to save the keys of your partner into the appropriate keystore. Partner keys must be saved to your truststore.

You can manually receive the keys (such as through an email, FTP, or other media) and then import them into your truststore using the instructions in “Importing a certificate from your partner” under the following circumstances:

- If you have already received the metadata of your partner, and only must receive a new certificate from your partner, or
- If you are using a SAML 1.x federation

Importing certificates from your partner's metadata file

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

About this task

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

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

See “Planning message-level security” on page 52.

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

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

Importing a certificate from your partner

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

Before you begin

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

About this task

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

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

You might also need to import a certificate that you have received from your partner. For more information on importing partner certificates, see "Importing a certificate from your partner" on page 61.

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

Use this task to import either:

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

Note: If you will use a PKCS#12 file, be sure to also follow the instructions in "Updating the cryptography policy" on page 65.

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

Imported keys are enabled by default.

Procedure

1. Click **Tivoli Federated Identity Manager** > **Key Service**.
   The Keystores panel 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

The **Upload Key File** panel opens.

9. Click **Next**.
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 use in the federation.

What to do next

Provide your keys to your partner. See "Providing certificates to your partner."

Providing certificates to your partner

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

Before you begin

See "Planning message-level security" on page 52 to determine which certificates you might need to provide to your partner. In general, the public keys you will need to provide include:

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

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

About this task

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

In a SAML 1.0 federation, you also have the option of providing this information to your partner manually. See "Exporting a certificate" on page 64. You could also use the manual method if you have already provided your metadata to your partner and you need to provide an updated certificate by itself.

Exporting certificates to a metadata file

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

About this task

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

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

See "Planning message-level security" on page 52.

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

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

Exporting a certificate

Export a certificate if you cannot provide a metadata file containing your keys to your partner.

Procedure

1. Click **Tivoli Federated Identity Manager** > **Key Service**.
   The Keystores panel opens.
2. Select the appropriate keystore from the Keystore table. You are prompted for your keystore password.
3. Type the password.
4. Click **OK**. The **View Keys** button is active.
5. Click **View Keys**. The Keys panel opens. Keys in the selected keystore are listed.
6. Select the keys you want to export.
7. Click the **Export** button. The Export Key panel opens.
8. 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
9. Ensure that the **Include Private Key** check box is *not* selected. Only you should have your private key.
10. Click **Download Key**.
11. 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.
12. Click **Cancel** to exit.

Updating the cryptography policy

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

About this task

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

To obtain the unlimited jurisdiction policy files:

Procedure

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

   Note: After you click this link, you are redirected to the policy file in the SDK that is compatible with your version of Java. However, the version number of the SDK might not be the same as the version number of the Java version you are using. For example, for Java 1.5 you might be directed to the SDK 1.4.
4. You are prompted to log on using your IBM user ID and password. If you do not have an IBM user ID and password, you need to register. Follow the registration link on the logon page.
5. Log on.
6. When prompted, select the .zip file for the version of Java you are using.
7. Click **Continue** to begin the download.
8. Unpack the .zip file. The JAR files are:
   - local_policy.jar
   - US_export_policy.jar
9. Place the files in the following directory:
   your_Java_runtime_installation_dir`/jre/lib/security`

   For example, your Java runtime might have been installed as part of the embedded version of WebSphere Application Server. In this case, the directory might be
   `/opt/IBM/FIM/`*ewas*`/java/jre/lib/security`

Removing default keystores

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

Procedure

1. Click **Tivoli Federated Identity Manager** > **Key Service**.
   The Keystores panel opens.
2. Select **DefaultKeyStore**.
3. Click **Delete**. A message asks you to confirm that you want to delete the specified keystore.
4. Click **OK** to delete the keystore.
5. Select **DefaultTrustedKeyStore**.
6. Click **Delete**. A message asks you to confirm that you want to delete the specified keystore.
7. Click **OK** to delete the keystore.

Enabling certificate revocation checking

You can use the IbmPKIX trust manager to determine the validity of server certificates. If you enable this function, the trust manager checks the certificate presented by the SSL server when the SOAP client establishes an SSL connection. Then, the trust manager checks the certificates that are used for XML messages signing, validation, encryption, and decryption. The operation attempt fails if the trust manager finds that the certificate has been revoked.

About this task

The following procedures are required to enable certificate revocation checking:

- "Enabling WebSphere for certificate revocation checking."
- "Enabling the IbmPKIX trust manager for SSL connection" on page 68.
- "Enabling the IbmPKIX trust manager for XML messages signing, validation, encryption, and decryption" on page 68

Enabling WebSphere for certificate revocation checking

You must enable some settings in WebSphere Application Server before you can configure the Tivoli Federated Identity Manager to do certificate revocation checking.

About this task

Enable the settings depending on the WebSphere Application Server type that you use. Choose the appropriate procedure for your installation:

Embedded WebSphere Application Server
"Enabling CRC on embedded WebSphere Application Server,"

Existing WebSphere Application Server
"Enabling CRC on existing WebSphere Application Server" on page 67.

Enabling CRC on embedded WebSphere Application Server

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

Before you begin

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

About this task

To enable the appropriate settings, complete the following steps:

Procedure

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

   Windows

   ```
   wsadmin.bat
   ```

   AIX, Linux, HP-UX, or Solaris

   ```
   wsadmin.sh
   ```

   Note: For more information about the options that can be specified when you run the wsadmin tool, see the http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp.
3. At the command prompt, run the following commands and replace *server1* with the name of your server:

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

What to do next

Continue with the steps in "Enabling the IbmPKIX trust manager for SSL connection" on page 68.

Enabling CRC on existing WebSphere Application Server

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

Procedure

1. Log on to the console for your WebSphere Application Server.
2. Click **Servers** > **Application Servers**.
3. Select your server.
4. Click **Java and Process Management** > **Process Definition** > **Java Virtual Machine**.
5. Under Generic JVM Arguments, add the following text:

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

What to do next

Continue with the steps in "Enabling the IbmPKIX trust manager for SSL connection."

Enabling the IbmPKIX trust manager for SSL connection

Enable Tivoli Federated Identity Manager to use the IbmPKIX trust manager to perform certificate revocation checking for certificates used for SSL connection.

Procedure

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

What to do next

Use the IbmPKIX trust manager to ensure the validity of the certificates used in XML security operations.

Enabling the IbmPKIX trust manager for XML messages signing, validation, encryption, and decryption

You can configure IBM Tivoli Federated Identity Manager to use the IbmPKIX trust manager. The IbmPKIX trust manager ensures that any certificates used in XML security operations like signing, encryption, signature validation, and decryption are valid according to their CRLs.

Procedure

1. Log on to the Integrated Solutions Console.
2. Select **Tivoli Federated Identity Manager** > **Domain Management** > **Runtime Node Management**.
3. Click **Runtime Custom Properties**. The Runtime Custom Properties panel opens.
4. Click **Create**. A list item is added to the list of properties with the name of **new key** and a value of **new value**.
5. Select the newly created placeholder property.
6. Type `kessjksservice.revocation.enabled` in the **Name** field. Do not insert the space character in this field.

7. Type `true` in the **Value** field.
8. Click **OK** to apply the changes that you have made. and exit from the panel.

Chapter 9. Setting up transport security

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

About this task

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

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

Procedure

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

Note: If you are a service provider in a SAML 1.x federation, you become the *client* in an SSL configuration. Therefore, you do not need to configure *server SSL*. See the steps for configuring client certificates in "Configuring your client certificates" on page 80.

   Enabling SSL on a server includes the following subtasks:
   a. "Creating a certificate request" on page 72.
   b. "Receiving a signed certificate issued by a certificate authority" on page 73.
   c. "Associating a certificate with your SSL configuration" on page 74.
   d. Optionally, you might want to complete the steps in "Deleting the default certificate" on page 75.
   e. "Extracting a certificate to share with your partner" on page 75.
2. "Configuring client authentication requirements" on page 76. Your authentication requirement options are:
   - No authentication
   - Basic authentication, in which a username and password are requested
   - Client certificate authentication
3. If you act as a client in the federation and your partner requires a client certificate, you must also complete the steps in "Configuring your client certificates" on page 80.

Enabling SSL on the WebSphere Application Server

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

Before you begin

Note: If you are a service provider in a SAML 1.x federation, you are always the client in an SSL configuration. Therefore, you do not need to configure SSL on your server. See the steps for configuring client certificates in "Configuring your client certificates" on page 80.

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 contains 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, see the WebSphere Application Server 6.1 information center http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp

About this task

Complete the following tasks in the console. If you need additional details, see 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**.
4. Click the name of the keystore where you must store the certificate, for example, **NodeDefaultKeyStore**.
5. Click **Personal certificate requests** under Additional Properties.
6. Click **New**.
7. 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 has an .arm extension. For example: `c:\servercertreq.arm` (on a Windows server).
8. 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.
9. 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.
10. In the **Organization unit** field, type the organization unit portion of the distinguished name.
11. In the **Locality** field, type the locality portion of the distinguished name.
12. In the **State or Province** field, type the state portion of the distinguished name.
13. In the **Zip Code** field, type the zip code portion of the distinguished name.
14. In the **Country or region** list, select the two-letter country code portion of the distinguished name.
15. Click **Apply**.

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. 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.
17. 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 make your request.
18. Make a backup copy of your keystore file before you receive the certificate that you have requested. Use the path information of your keystore as shown in the console to locate the file. Then copy it to a new location for safe-keeping.

What to do next

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

Receiving a signed certificate issued by a certificate authority

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

Before you begin

The certificate request must have been created and must be in a WebSphere keystore as described in "Creating a certificate request" on page 72. Also, the certificate must have been received from the CA and placed on your computer so that you can receive it into the keystore.

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

About this task

Complete the following tasks in the console. If you need additional details, 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** and **Save**. The keystore contains a new personal certificate that is issued by a CA. The SSL configuration is ready to use the new CA-signed personal certificate.

What to do next

Associate the certificate with your SSL configuration. See "Associating a certificate with your SSL configuration."

Associating a certificate with your SSL configuration

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

About this task

When you install WebSphere Application Server 6.1 and 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 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 72 and "Receiving a signed certificate issued by a certificate authority" on page 73 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 uses the new certificate.

What to do next

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

Deleting the default certificate

After you have received your personal signed certificate, delete the default key so that it is not used inadvertently.

About this task

Attention: Ensure that none of your SSL configurations use the default key before continuing with this procedure. See the instructions in "Associating a certificate with your SSL configuration" on page 74.

Procedure

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

What to do next

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

Extracting a certificate to share with your partner

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

Before you begin

The keystore and the personal certificate must already exist.

Procedure

1. Log on to the console.
2. Click **Security** > **SSL certificate and key management** > **Manage endpoint security configurations**.

3. Select your node on the **Outbound** tree.
4. Click **Manage certificates**.
5. Select the CA signed certificate.
6. Click **Extract** in the upper-right corner.
7. 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`
8. Select the default data type from the list.
9. Click **Apply**.
10. 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 is using Tivoli Federated Identity Manager, the partner must import your certificate into its Tivoli Federated Identity Manager truststore.

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

Configuring client authentication requirements

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

About this task

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

First, you must decide whether you require client authentication.

- If you do not require client authentication, see "Configuring access with no authentication."
- If you require client authentication, you have two options:
  - Basic authentication. See "Configuring basic authentication access" on page 77.
  - Client certificate authentication. See "Configuring access with client certificate authentication" on page 78.

Configuring access with no authentication

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

About this task

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

Note: These instructions apply to stand-alone WebSphere servers. For WebSphere Network Deployment servers in a cluster, see "Configuring IHS for client worksheet" on page 91.

To ensure that this setting is selected:

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 that you use in your environment.
4. Click **Advanced**. The SOAP Endpoint Security Settings panel opens.
5. Ensure that the SOAP Port is correct in your configuration and that **Allow unauthenticated users access to SOAP endpoints** is selected.
6. Click **OK**.
7. Click **Load configuration changes to Tivoli Federated Identity Manager runtime**.

What to do next

If you are configuring a SAML 2.0 federation, continue with the steps for configuring your client certificate, "Configuring your client certificates" on page 80. If you are configuring a SAML 1.x federation, the task is complete.

Configuring basic authentication access

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

Before you begin

Before beginning this task:

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

About this task

To configure basic authentication, complete the following steps.

Procedure

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

   Note: See the user creation instructions for the user registry you have configured for your environment.
2. Your next step depends on whether to allow individual authenticated users or authenticated users who are part of specific groups.
   - If you require basic authentication from individual users, repeat step 1 for each service provider user you need to configure. Then, proceed to step 3 on page 78.

- If you require basic authentication from users in specific groups, create a group for the users and add the user you created in step 1 on page 77 to the group. For example, create a group with a name of `soapgroup` and then add user `soapclient` to the group.

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

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

   Note: These instructions apply to standalone WebSphere servers. For WebSphere Network Deployment servers in a cluster, see "Configuring IHS for client worksheet" on page 91.

   a. Log on to the console.
   b. Click **Tivoli Federated Identity Manager** > **Manage Configuration** > **Point of Contact**.
   c. Select the point of contact server that you are using in your environment.
   d. Click **Advanced**. 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 **Basic Authentication**.
   g. Click **OK**.
   h. Click the **Load configuration changes to Tivoli Federated Identity Manager runtime** button.

What to do next

If you are configuring a SAML 2.0 federation, continue with the steps for configuring your client certificate, "Configuring your client certificates" on page 80.

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

Configuring access with client certificate authentication

If you require client certificate authentication from your partner, you complete the tasks in this topic.

Before you begin

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

Before beginning this task:

- Ensure you have the public key certificate for the client certificate that your partner uses to access your artifact resolution endpoint.

- Ensure you have the common name attribute of the certificate that your partner uses 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 allow access to the endpoint by authenticated users individually or authenticated users who are part of specific groups.

Procedure

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

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

       Note: See the 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 step 10a for each service provider user you need to configure. Then proceed to step 11.
       - If you require client certificate authentication from users in specific groups, create a group for the users and add the user you created in step 10a to the group. For example, create a group with a name of `soapgroup` and then add user `soapclient` to the group.

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

Note: These instructions apply to standalone WebSphere servers. For WebSphere Network Deployment servers in a cluster, see "Configuring IHS for client worksheet" on page 91.

a. Log on to the console.
b. Click **Tivoli Federated Identity Manager** > **Manage Configuration** > **Point of Contact**.
c. Select the point of contact server that you are using in your environment.
d. Click **Advanced**. 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.

What to do next

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

Configuring your client certificates

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

Retrieving the server certificate from your partner

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

Before you begin

Before continuing with this procedure, ensure that you have a truststore prepared for storing the certificate. See "Preparing the keystores" on page 49.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Key Service**.
   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 **Retrieve Certificate from SSL Connection**. The Password panel opens.
5. Type your truststore password .
6. Click **OK**.

7. Complete the fields to specify the host name and port name from which you retrieve the certificate.

   (Optional) Click the **Show Signer Info** to view the certificate before retrieving.
8. Complete the **Alias** field with the name you want to use for the certificate.
9. Click **OK**. The certificate is added to the truststore.

What to do next

If you assume the role of a client in an SSL connection with your partner and your partner requires you to authenticate using a client certificate, continue with "Obtaining your client certificate."

Obtaining your client certificate

If assume the role of a client in an SSL connection with your partner and your partner requires you to authenticate using a client certificate, obtain and configure the certificate. Then, share that certificate with your partner.

Before you begin

Ensure that you have a keystore prepared for storing the certificate. See"Preparing the keystores" on page 49.

Procedure

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

      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 and click **OK**.
   f. Click **Certificate Request**. The Create a certificate request panel opens.
   g. Complete the fields.
   h. Then click **OK**. A public/private key pair is added to the keystore and a file with the encoded BASE64 data is created. The temporary self-signed certificate is replaced with the signed certificate from the CA.

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

      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 and click **OK**.
   f. Click **Receive Certificate from CA**.
   g. Select the location of the certificate that you received from the CA.

h. 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 **Tivoli Federated Identity Manager** > **Key Service**.

      The Keystores panel opens.
   c. Select the appropriate keystore from the Keystore table. You are prompted for your keystore password.
   d. Type the password.
   e. Click **OK**. The **View Keys** button is active.
   f. Click **View Keys**. The Keys panel opens. Keys in the selected keystore are listed.
   g. Select the keys you want to export.
   h. Click the **Export** button. The Export Key panel opens.
   i. 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
   j. Ensure that the **Include Private Key** check box is *not* selected. Only you should have your private key.
   k. Click **Download Key**.
   l. 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.
   m. Click **Cancel** to exit.

What to do next

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

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

Chapter 10. Selecting a point of contact server

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

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

The point of contact server provides endpoints, which are the locations to and from which messages are sent and received. Each endpoint has a URL, so that the endpoints can be accessed by external users as Web sites on the Internet. The point of contact receives access requests and provides the authentication service.

It serves as the first component capable of evaluating the authentication credentials of the user that is requesting access to the protected network. It also manages session lifecycle of the user, from session creation, to session access, to session deletion (such as in response to session logout services).

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

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

WebSphere as point of contact server

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

Identity Provider options

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

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

Service Provider options

When you use IBM WebSphere Application Server as the point of contact server and you are the service provider in a federation, single sign-on is enabled using Lightweight Third-Party Authentication (LTPA).

You can use the following hosting application options in a federation that is configured in Tivoli Federated Identity Manager:

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

Each of these options has specific requirements. For more information about these requirements, see "WebSphere as point of contact for identity providers" on page 93 and "WebSphere point of contact server for a service provider" on page 108.

WebSEAL as point of contact server

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

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

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

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

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

  For example:

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

  For example:

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

  For example:

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

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

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

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

WebSEAL No ACLD as point of contact server

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

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

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

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

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

Generic point of contact server

The generic point of contact is an additional point of contact implementation provided by Tivoli Federated Identity Manager. It is a HTTP-headers based solution that provides administrators with the ability to modify their point of contact environments (for example, Apache) to set and read headers. This allows integration with Tivoli Federated Identity Manager without writing a custom point of contact server.

The generic point of contact works pretty much the same as the WebSEAL point of contact server. The main difference is that headers names are used for the user information.

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

Custom point of contact server

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

For example, a custom point of contact server would be useful in the following scenarios:

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

  You can choose from any of the point of contact server implementations:

  – A point of contact server that uses a local identity callback to retrieve the user for the transaction.
  – A custom point of contact server that uses a SignIn callback to assert the user identity to the environment.
  – A point of contact server that uses both types of callbacks.

Developing a custom point of contact server requires programming experience with developing callback modules and knowledge of Tivoli Federated Identity Manager programming concepts. See the developerWorks® links 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.

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

Chapter 11. Configuring WebSphere as point of contact server

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

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

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

Set up an outbound HTTP proxy in the WebSphere Application Server so that you can use it when Tivoli Federated Identity Manager makes connections to other HTTP servers.

For more information, see the following topics:

- "Using IBM HTTP Server with WebSphere as point of contact"
- "WebSphere as point of contact for identity providers" on page 93
- "WebSphere point of contact server for a service provider" on page 108
- "Setting up an outbound HTTP proxy server" on page 91

Using IBM HTTP Server with WebSphere as point of contact

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

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

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

For instructions, consult the information center for your IBM HTTP Server for WebSphere Application Server: http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp. See the topics that describe how to secure an IBM HTTP Server, including:

- Working with key database
- Securing with SSL communications

Adding an SSL port for a SOAP backchannel

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

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

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

For instructions, consult the information center for your IBM HTTP Server for WebSphere Application Serverhttp://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp. See the topics that describe how to secure an IBM HTTP Server, including:

- Securing with SSL communications

Updating federation configuration for SOAP connection

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

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

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

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

Confirming WebSphere Application Server security properties

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

Before you begin

The settings are:

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

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

Use the WebSphere management console to check the WebSphere settings.

About this task

Application and administration security is enabled

To confirm that application and administration security are enabled:

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

Single sign-on is enabled

To confirm that single sign-on is enabled:

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

Enabling multiple language encoding on WebSphere Application Server

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

About this task

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

Procedure

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

   Windows

   ```
   wsadmin.bat
   ```

   AIX, Linux, 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 centerhttp://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp.
3. At the command prompt, run the following commands to enable UTF-8 encoding:
   a. To show the current JVM properties:

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

b. To set the JVM properties:

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

c. To save the configuration changes:

```
$AdminConfig save
```

4. Restart WebSphere Application Server.

Mapping application roles to users

You can map different security application roles to the users of IBM Tivoli Federated Identity Manager.

Before you begin

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

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

About this task

Use the WebSphere administration console to specify the mappings

Procedure

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

   For each role, select either **Everyone** or **All authenticated**.

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

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

   For instructions, consult the WebSphere information center: http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp. See the topic *Mapping user to roles*.

Results

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

Configuring IHS for client worksheet

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

Complete the instructions in the following section for your authentication method.

Configuring certificate authentication for IHS

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

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

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

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

For instructions, consult the information center for your IBM HTTP Server for WebSphere Application Server: http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp. See the topics that describe how to secure an IBM HTTP Server, including:

- Storing a certificate authority certificate

Configuring basic authentication for IHS

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

For instructions, consult the information center for your IBM HTTP Server for WebSphere Application Server: http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp. See the topics that describe how to secure an IBM HTTP Server, including:

- Authenticating with LDAP on IBM HTTP Server

Setting up an outbound HTTP proxy server

Set up an outbound HTTP proxy in the WebSphere Application Server so that you can use it when Tivoli Federated Identity Manager makes connections to other HTTP servers.

About this task

The configuration must be set on the Tivoli Federated Identity Manager runtime nodes. The JVM configuration settings that enable outbound HTTP connections

through a proxy applies to all outbound connections for all applications, including Tivoli Federated Identity Manager, that are running on WebSphere Application Server.

The following procedure applies to WebSphere Application Server 7.0. The steps are similar to other WebSphere Application Server versions.

Procedure

1. Log on to the **Integrated Solutions Console**.
2. Select **Servers** > **Server Types** > **WebSphere application servers**.
3. Click the appropriate server. For example, **server1**.
4. Under **Server Infrastructure**, click **Java and Process Management** > **Process Definition**.
5. Under **Additional Properties**, click **Java Virtual Machine**.
6. Click the **Configuration** tab.
7. Under **Additional Properties**, click **Custom properties**.
8. Click **New**.
9. Specify the appropriate **Name** and **Value** based on the required configuration. The required configuration is based on whether you are using HTTP or HTTPS. See Table 1 for more details.
10. Click **OK**.
11. Repeat steps 8 to 10 for each required configuration.
12. Click **Save directly to the master configuration**.
13. Restart the **WebSphere Application Server**.

    Note: Repeat steps 2 to 13 for each server. This configuration is for WebSphere Application Server JVM so it must be repeated for each Tivoli Federated Identity Manager runtime node and WebSphere Application Server instance.

Table 8. A list of all the possible name and values for an outbound HTTP proxy server

Name	Sample values	Description
`http.proxyHost`	`http.proxy.ibm.com`	The hostname or IP address of the HTTP proxy
`http.proxyPort`	`3128`	The port of the HTTP proxy
`http.proxyUser`	`admin`	The username that is used to authenticate to the proxy for HTTP connections
`http.proxyPassword`	`password`	The password that is used to authenticate to the proxy for HTTP connections
`https.proxyUser`	`admin`	The username that is used to authenticate to the proxy for HTTPS connections
`https.proxyPassword`	`password`	The password that is used to authenticate to the proxy for HTTPS connections
`https.proxyHost`	`https.proxy.ibm.com`	The hostname or IP address of the HTTPS proxy
`https.proxyPort`	`3128`	The port of the HTTPS proxy

Table 8. A list of all the possible name and values for an outbound HTTP proxy server (continued)

Name	Sample values	Description
`http.nonProxyHosts`	`host1.ibm.com|internal.ibm.com`	A list of hosts, which are separated by `|`, which a proxy must not be used

Note: The `http.nonProxyHosts` property applies for both HTTP and HTTPS connections.

WebSphere as point of contact for identity providers

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

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

- Form-based authentication using any user registry that is supported by WebSphere Application Server
- Windows desktop authentication using the WebSphere Application Server 8.0 SPNEGO TAI support and Microsoft Active Directory as the user registry

Attention: Before proceeding with the tasks described in this chapter, confirm that your settings are correct using "Confirming WebSphere Application Server security properties" on page 88.

Form-based authentication

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

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

An example of this configuration is shown in Figure 1 on page 94.

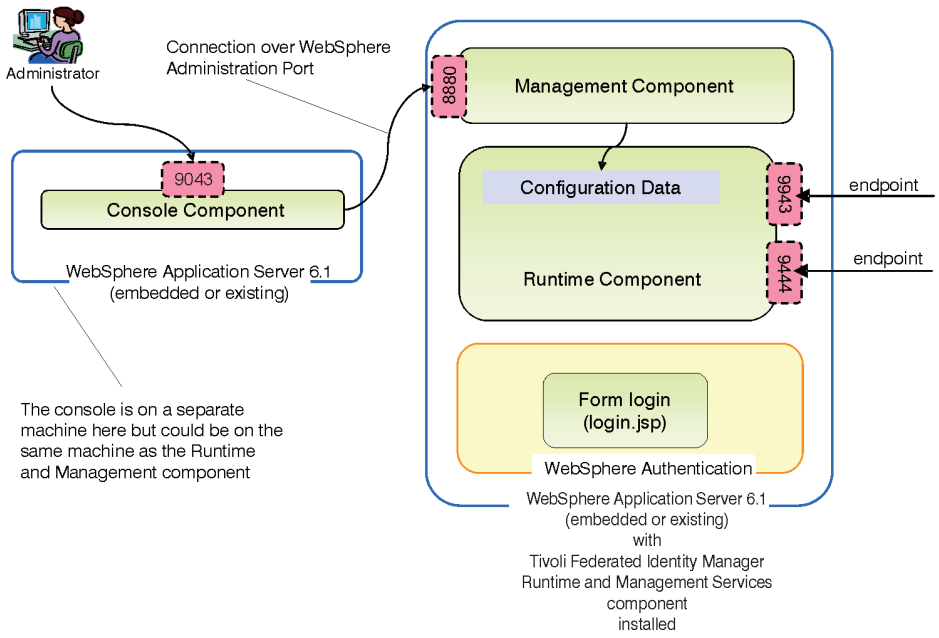


Figure 1. Example of WebSphere Application Server with form-based authentication

Notes on configuration:

- The WebSphere Application Server can be either an existing WebSphere deployment (with the correct level of fix pack applied) or can be the embedded version of WebSphere Application Server Version 8.0 that is distributed with Tivoli Federated Identity Manager.
- A log on form presented by the WebSphere Application Server where Tivoli Federated Identity Manager is installed. The log on form is provided.

Complete the tasks in "Configuring form-based authentication" on page 95.

Windows desktop authentication through SPNEGO TAI with Microsoft Active Directory

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

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

An example of this configuration is shown in Figure 2 on page 95.

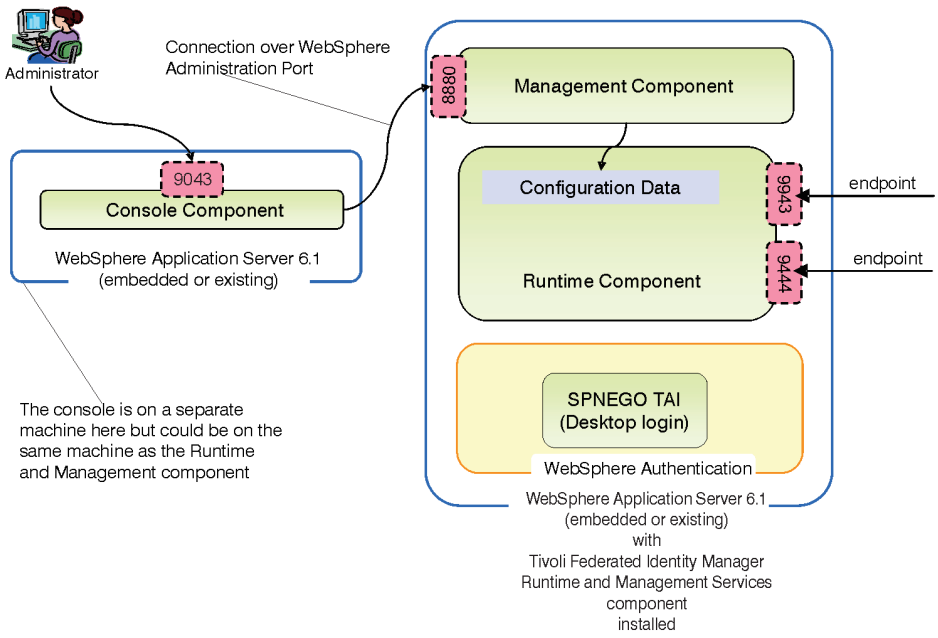


Figure 2. Example of WebSphere Application Server with SPNEGO TAI authentication

Configuration notes:

- The WebSphere Application Server can be either an existing WebSphere deployment (with the correct level of fix pack applied) or can be the embedded version of WebSphere Application Server Version 8.0 that is distributed with Tivoli Federated Identity Manager.
- Microsoft Active Directory must be used as the user registry. Use a version that is supported by Microsoft Windows Server 2003. The user registry must include a user for the WebSphere administrative user and a user for the Kerberos identity. In addition, a keytab file must be built for each user. You need the LDAP connection properties for the Active Directory server prior to configuring Tivoli Federated Identity Manager.

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

Complete the tasks in “Configuring SPNEGO authentication” on page 99.

Configuring form-based authentication

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

About this task

The tasks include:

1. "Selecting and installing the user registry"
2. "Configuring the user registry"
3. "Adding single sign-on users" on page 97
4. "Adding administrative users" on page 97
5. "Configuring user registry for embeddedWebSphere" on page 97
6. "Configuring an SSL connection to the user registry" on page 98
7. "Customizing the login form" on page 98

Selecting and installing the user registry

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

Before you begin

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

If you are using form-based authentication, you can choose a user registry from many options. See the WebSphere Application Server 8.0 information center at http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp. Then, locate information about selecting a user registry by selecting **WebSphere Application Server (Distributed platforms and Windows)** > **Securing applications and their environment** > **Authenticating users** > **Selecting a registry or repository**.

About this task

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

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

- Use the default file-based user repository realm (the federated repository), which was installed with the embedded version of WebSphere Application Server. The administrative user was configured in this registry during installation. Additional tasks needed for adding the single sign-on users are provided later in this chapter.
- Use a different user registry. Review the WebSphere Application Server documentation for information about your user registry options. Then, install and configure the user registry you chose, if you are not using a previously existing user registry. Then, configure WebSphere to use that user registry. See"Configuring user registry for embeddedWebSphere" on page 97.

Configuring the user registry

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

Before you begin

You must already have selected which user registry to use, and have installed it as described in "Selecting and installing the user registry" on page 96.

About this task

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

Adding single sign-on users:

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

Adding administrative users:

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

Before you begin

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

Procedure

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

Configuring user registry for embeddedWebSphere

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

Procedure

1. Log on to the console.
2. Select **Security** > **Secure administration, applications, and infrastructure**. The Configuration tab opens.
3. Click **Security Configuration Wizard** to change the user registry used by the WebSphere runtime. The **Specify extent of protection panel** opens.
4. Verify that the check box **Enable application security** is selected.
5. Click **Next**. The **Secure the application serving environment** panel opens.
6. Select the appropriate option for the user registry to use:
   - **Federated repositories**
   - **Standalone LDAP registry**

- **Local operating system**
- **Standalone custom registry**

7. Click **Next**. The **Configure user repository** panel opens.
8. Specify values for each of the registry configuration settings. See the online help for descriptions of the fields presented.
9. Click **Next** and finish the wizard.
10. Save your configuration changes.
11. Stop the WebSphere Application Server.
12. Restart the WebSphere Application Server. You must use the same administrative name you used to log on and make these changes.
13. From the console, select **Tivoli Federated Identity Manager** > **Manage Configuration** > **Domain properties**.
14. In the WebSphere Security section of the panel, update the following values:

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

    Administrative user password
    Enter the password for LDAP administrator.
15. Save the changes.
16. Stop the WebSphere Application Server.
17. Restart the WebSphere Application Server.

Configuring an SSL connection to the user registry

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

About this task

For instructions, see the WebSphere Application Server 8.0 information center at http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp. Locate information about creating SSL connections by selecting **WebSphere Application Server (Distributed platforms and Windows)** > **Securing applications and their environment** > **Securing communications**.

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

Customizing the login form

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

About this task

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

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

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

Configuring SPNEGO authentication

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

About this task

Complete these configuration tasks to configure the SPNEGO authentication:

Procedure

1. Configuring the Microsoft Active Directory, including:
   a. Creating an Active Directory user for the WebSphere administrative user.
   b. Creating an Active Directory user that contains the Service Principal Name (SPN) of the Tivoli Federated Identity Manager server.
   c. Building a Kerberos keytab file and assigning the SPN for the Active Directory user created in 1b.
   d. Collecting the Active Directory configuration parameters.
2. Configuring the Windows domain and user logins.
3. Configuring WebSphere Application Server, including:
   a. Configuring administration security, with Active Directory used as the type of LDAP user registry.
   b. (Optional) Configuring an SSL connection to Active Directory.
4. Enabling WebSphere SPNEGO and the Trust Association Interceptor (TAI), using the Integrated Solutions Console.

   (Optional) Customize the TAI attributes, as might be required in your environment.
5. Instructing your users to configure Internet Explorer, as follows:
   a. Adding the hostname as a trusted host in the Intranet Zone.
   b. Enabling Integrated Windows Authentication.

Configuring Active Directory for use with SPNEGO

Use Microsoft Active Directory as your user registry when you use WebSphere Application Server with SPNEGO authentication.

Before you begin

You must perform several configuration tasks in Microsoft Active Directory:

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

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

About this task

For detailed information on completing the steps in this procedure, see the Microsoft Active Directory documentation.

Procedure

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

Table 9. Parameters to use with the Microsoft Windows **ktpass** *command*

Parameter	Example value	Description
-out	`was1-krb5.keytab`	A file name in which to store the secret key that is later used for Kerberos authentication validation on the WebSphere server. This file is uploaded to the WebSphere server when you enable SPNEGO. See "Enabling and configuring SPNEGO authentication" on page 104.
-princ	`HTTP/ibm-fim611-1.fimtest. example.com@FIMTEST .EXAMPLE.COM`	The Kerberos service principal name to use for generating the key. The name is case-sensitive and must start with `HTTP/`. The portion following the `HTTP/` must be the fully qualified DNS domain name of the URL that users see on their browsers when accessing the WebSphere server.
-pass	`*`	The password to set for the Kerberos principal. A value of `*` results in prompting for the password. The password must match the user created in step 2.
-mapuser	`was-1`	The Active Directory user to whom the Kerberos service principal is mapped. The value here must match the user name you created in step 2.
-mapOp	`set`	Indicates that the SPN must overwrite any existing value mapped for this Active Directory user.

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

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

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

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

Type the password again to confirm:

Key created.

Output keytab to was1-krb5.keytab:

Keytab version:0x502

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

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

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

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

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

Figure 3. Example of the ktpass command

The keytab file that is created in this step is uploaded to the Tivoli Federated Identity Manager server during the configuration of WebSphere Application Server. See "Configuring WebSphere for use with SPNEGO" on page 102 for details.

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

      Hostname
      The host name of the Active Directory server.

      Port Port number of the active directory server.

      Base DN
      The base search DN for active directory users.

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

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

   b. If an SSL connection is required to Active Directory, WebSphere must be configured with the certificate of the issuing CA of the domain controller. If Windows Certificate Services was installed on the domain controller, this becomes the CA certificate of the Certificate Services on that domain controller.

      To export the CA certificate to a file:

1) Open **Administrative Tools** > **Certification Authority**.
2) Right-click on the top-level CA name.
3) Click **Properties**.
4) Select the **General** tab and then click **View Certificate**.
5) Click the **Details** tab and click **Copy to File**.

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

Configuring the Windows domain and user logins

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

About this task

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

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

By default, the SPNEGO TAI support in Tivoli Federated Identity Manager shows a login form when a user who has not authenticated through the desktop login attempts a single sign-on. By default, the login form is the sample login form that is provided with Tivoli Federated Identity Manager.

You can customize the appearance of this form, as described in "Customizing the login form" on page 98. If you do not want to show this login form, you can modify the TAI attributes as described in "Configuring custom TAI attributes" on page 107.

Configuring WebSphere for use with SPNEGO

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

About this task

The steps include:

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

Procedure

1. (Optional) Load the CA root certificate of the Active Directory server. This step is required only if you use LDAP/SSL to communicate with the Active Directory server.
2. Before continuing with this step, make sure you have completed the steps in "Configuring Active Directory for use with SPNEGO" on page 99, especially step 4b on page 101. If you are not using LDAP/SSL, continue with the next step.
   a. Log on to the console.
   b. Click **SSL certificate and key management**.
   c. On the SSL Certificate and key management panel, click **Key stores and certificates**.
   d. On the Key stores and certificates panel, click **NodeDefaultTrustStore**.
   e. On the NodeDefaultTrustStore panel, click **Signer certificates**.
   f. On the Signer certificates panel, click **Add** to add a new signer.
   g. Complete the details for the signer certificate. Use the following values:

Table 10. Signer certificate details in SPNEGO environment

Field name	Value
Alias	Any alias name for the CA certificate from Active Directory. For example, you might use the name of the Active Directory domain controller.
File name	The path and file name of the certificate. Note that this path and file name is on the server where WebSphere Application Server is installed, not on the server where the browser is running. This means that the file must be copied to the WebSphere server prior to completing this step.
Data type	The file format for the certificate. Use the same format you used in step 4b on page 101.

   After the certificate is successfully loaded, the certificate is added in the signer certificates list.
   h. Click **OK**.
3. Enable WebSphere application security with Active Directory set as the user registry.

   Note: To complete this step, you must be able to contact the Active Directory server using port 389 (that is, without using SSL). During this step, the security configuration wizard performs a connection test which does not support SSL. The setup cannot proceed if this test fails. You can enable LDAP/SSL after the test and setup have been completed.
   a. In the console, select **Security** > **Secure administration, applications, and infrastructure**.
   b. Click the **Security Configuration Wizard** button to start the Security Configuration wizard.
   c. Click **Next**.
   d. In step 1 of the wizard, make sure that the **Enable application security** check box is selected.
   e. Click **Next**.
   f. In step 2 of the wizard, select **Standalone LDAP directory**.

g. Click **Next**.
h. In step 3 of the wizard, enter the following parameters.

Table 11. Parameters for the LDAP directory in SPNEGO environment

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

i. Click **Next**.
j. In step 4 of the wizard, the connection to the Active Directory server is tested.
k. Click **Finish** to complete the wizard.

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

Enabling and configuring SPNEGO authentication

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

About this task

Use the console to complete this procedure. This procedure includes:

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

Procedure

1. Log on to the console.

2. Click **Tivoli Federated Identity Manager** > **Manage Configuration** > **Point of Contact**.
3. Select the point of contact 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. Select the **Enable SPNEGO Authentication** check box.
7. Complete the fields with the information for your authentication configuration. See the online help for complete descriptions of the fields.
8. Import the Kerberos keytab file, which you created using the -out option of the ktpass utility, as follows:
   a. Click the **Import Keytab file** button.
   b. In the **Location of Keytab File** field, type the path for the location of the file.

      (Optional) Use the **Browse** button to locate the file.
   c. Click **Finish**.
9. Click **OK**.

Configuring the Trust Association Interceptor

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

About this task

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

Note: If you plan to make changes to these default values, see the instructions in "Configuring custom TAI attributes" on page 107.

SPNEGO TAI configuration attributes:

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

Content

The file is located in the following default directory:

AIX, Linux or Solaris

```
/opt/IBM/FIM/etc/tai.properties.template
```

Windows

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

In general, you do not need to modify this file. You can configure the TAI by using the console as described in "Configuring the Trust Association Interceptor." However, if you need to make additional changes that require updates to the `tai.properties.template`, use the instructions in "Configuring custom TAI attributes" on page 107.

Note: The version of the tai.properties.template file that is installed as part of Tivoli Federated Identity Manager contains additional attributes that are not provided with WebSphere Application Server 8.0. If your environment requires the use of attributes that are not described here, refer to the WebSphere Application Server 8.0 information center for a list of all the attributes available for the customization of SPNEGO TAI configuration. The information center is located at http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp.

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

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

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

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

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

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

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

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

Figure 4. tai.properties.template file

Macros

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

Table 12. Macros used in the tai.properties.template file

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

Configuring custom TAI attributes:

Configure the custom TAI attributes to suit your deployment requirements.

Before you begin

The TAI is enabled automatically when you enable SPNEGO using the console as described in "Enabling and configuring SPNEGO authentication" on page 104. However, if you need to customize TAI attributes, modify the tai.properties.template file.

About this task

Review the content of the tai.properties.template file in "SPNEGO TAI configuration attributes" on page 105.

Procedure

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

   AIX, Linux or Solaris

   `/opt/IBM/FIM/etc/tai.properties.template`

   Windows

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

Configuring browsers for use with SPNEGO

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

Before you begin

This requires that:

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

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

About this task

In general, browser configuration for SPNEGO involves:

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

Procedure

1. Add the hostname:
   a. Start Windows Internet Explorer
   b. Click **Tools** > **Internet Options**.
   c. Click the **Security** tab.
   d. Click **Local intranet**.
   e. Click the **Sites** button. Make sure the **Include all local (intranet) sites not listed in other zones** is checked.
   f. Click **Advanced**.
   g. Add the Web sites for the WebSphere Application Server as viewed at the browser, using either http or https, as needed.

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

      ```
      http://ibm-fim611-1.fimtest.example.com
      https://ibm-fim611-1.fimtest.example.com
      ```
2. Verify that Integrated Windows Authentication is enabled:
   a. Start Windows Internet Explorer.
   b. Click **Tools** > **Internet Options**.
   c. Click the **Advanced** tab and scroll to the Security section.
   d. Ensure that the **Enable Integrated Windows Authentication (requires restart)** box is selected.
   e. Save the changes and restart the browser, if necessary.

WebSphere point of contact server for a service provider

There are several configuration options when you assume the service provider role in your federation.

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

- IBM WebSphere Application Server 5.1 or 6.0 or later

In most cases, you can host your web applications on an installation of WebSphere Application Server that is separate from the server where Tivoli Federated Identity Manager is installed.

You can use the same server for both the Tivoli Federated Identity Manager and your applications under the following circumstances:

- If your WebSphere Application Server installation is version 6.1
- If your WebSphere Application Server installation meets the Tivoli Federated Identity Manager installation and web application hosting requirements

- Microsoft Internet Information Service 6.0
- IBM HTTP Server 6.1
- Apache HTTP Server 2.0 and 2.2

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

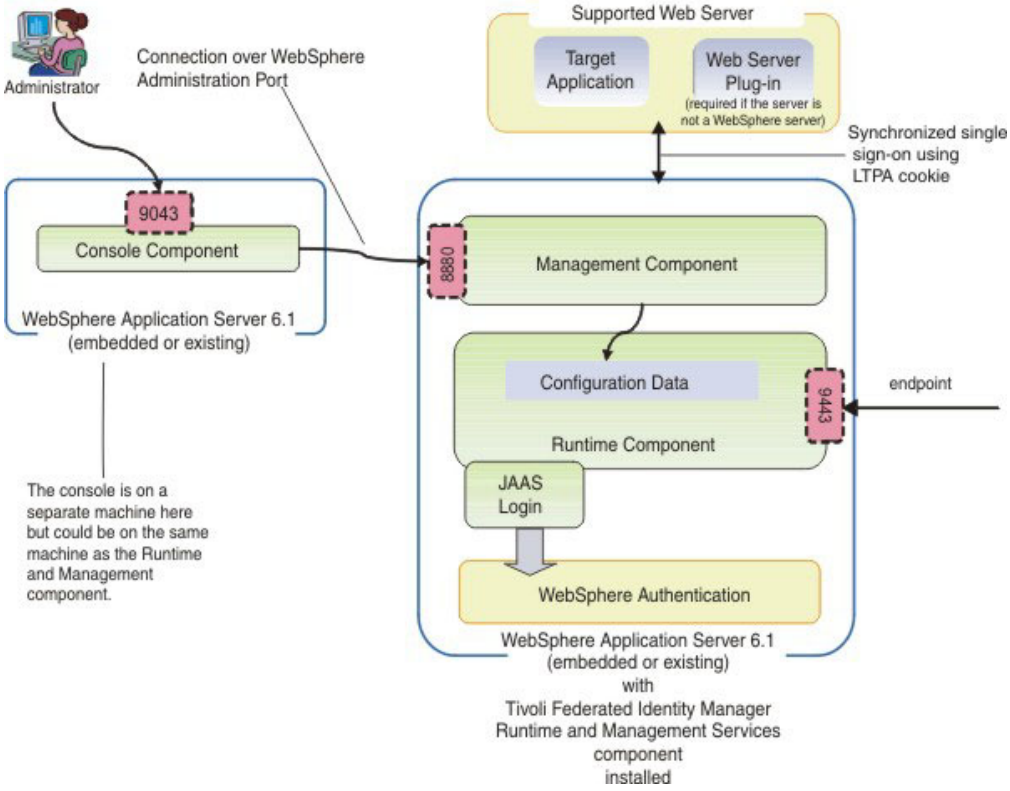


Figure 5. Example of Tivoli Federated Identity Manager and a Web application server

In the configuration depicted, the target application is hosted by a server that is separate from the Tivoli Federated Identity Manager server. The user authenticates to the identity provider, and the credential is transferred from the identity provider to Tivoli Federated Identity Manager. The service provider validates the token in the Tivoli Federated Identity Manager, and returns an LTPA cookie with the user identity. It also returns any attributes carried by the token, or added by the service provider mapping rules.

The user is redirected by some single sign-on protocol to the target application where the LTPA cookie is transferred from the Tivoli Federated Identity Manager node to the Web server node. The LTPA key must be shared between these nodes for the cookie to be recognized.

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

Environment requirements

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

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

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

- Applications must be able to accept user identity by way of an HTTP header or server variable.
- A user registry is required in your environment for both your point of contact server and your application server. The users to whom you are providing single sign-on capabilities must exist in both user registries. Configure a user registry for the separate server that hosts your target application.

  Examples of separate server can be, another WebSphere Application Server, or a supported server with a plug-in such as an IHS, IIS, or Apache server. Select a user registry that can be used for your point of contact server and your Web server to minimize the number of user registries you must maintain in your environment.

Plug-in requirements

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

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

Configuring WebSphere

To configure your WebSphere Application Server point of contact server, continue with the instructions in “Configuring a WebSphere Application Server point of contact server (service provider)” on page 111.

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

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

About this task

Attention: Before proceeding with the tasks described in this section, confirm that your settings are correct using "Confirming WebSphere Application Server security properties" on page 88.

Configuring the LTPA cookie

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

Procedure

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

What to do next

Additional information about setting the domain properly can be found in the topic about implementing single sign-on to minimize Web user authentications of the WebSphere Application Server 8.0 Information Center at http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp.

Defining attributes for the LTPA token

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

About this task

Note: Your target application must be configured to make use of the attributes that are included in the LTPA token. For more information about development topics, see the developerWorks links on the Welcome page of 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.

Procedure

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

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

   A list of configuration properties shows.

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

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

Selecting and installing a user registry

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

Before you begin

Because you are using WebSphere Application Server as your point of contact server, you can choose a user registry from many options. See the WebSphere Application Server 8.0 information center at http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp. Then, locate information about selecting a user registry by selecting **WebSphere Application Server (Distributed platforms and Windows)** > **Securing applications and their environment** > **Authenticating users** > **Selecting a registry or repository**.

- If you are using an existing installation of WebSphere Application Server, you might have a compatible user registry already installed and configured.
- If you are using a new installation of the embedded version of WebSphere Application Server, you have several options:
  - Use the default file-based user repository realm (the federated repository), which was installed with the embedded version of WebSphere Application

Server. The administrative user was configured in this registry during installation. Additional tasks needed for adding the single sign-on users are provided later in this chapter.

– Use a different user registry. Review the WebSphere Application Server documentation for information about your user registry options. Then, install and configure the user registry you chose, if you are not using a previously existing user registry. Configure WebSphere to use that user registry. Refer to "Configuring WebSphere to use the user registry" on page 114.

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

Configuring the user registry

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

Before you begin

Before continuing with this task, you should have already selected which user registry to use and have installed it as described in "Selecting and installing a user registry" on page 112.

About this task

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

Adding single sign-on users:

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

Adding administrative users:

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

Before you begin

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

About this task

To add this user to a different user registry:

Procedure

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

Configuring an SSL connection to the user registry:

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

About this task

For instructions, see the WebSphere Application Server 8.0 information center at http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp. Locate information about creating SSL connections by selecting **WebSphere Application Server (Distributed platforms and Windows)** > **Securing applications and their environment** > **Securing communications**.

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

Configuring WebSphere to use the user registry

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

Before you begin

Before continuing with this task, review the information in "Selecting and installing the user registry" on page 96. Ensure that you have selected and installed the appropriate user registry option for your environment.

About this task

To enable WebSphere to use your user registry:

Procedure

1. Log on to the console.
2. Select **Security** > **Secure administration, applications, and infrastructure**. The Configuration tab opens.
3. Click **Security Configuration Wizard** to change the user registry used by the WebSphere runtime.
4. The **Specify extent of protection panel** is displayed. Verify that the check box **Enable application security** is selected. Click **Next**.
5. The **Secure the application serving environment** panel is displayed. Select the appropriate option for the user registry to use:
   - **Federated repositories**
   - **Standalone LDAP registry**
   - **Local operating system**
   - **Standalone custom registry**
6. Click **Next**. The **Configure user repository** panel opens.
7. Specify values for each of the registry configuration settings. See the online help for descriptions of the fields presented.

8. Click **Next** and finish the wizard. Save your configuration changes.
9. Stop the WebSphere Application Server.
10. Restart the WebSphere Application Server. You must use the same administrative name you used to log on and make these changes.
11. From the console, select **Tivoli Federated Identity Manager** > **Manage Configuration** > **Domain properties**.
12. In the WebSphere Security section of the panel, update the following values:

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

    Administrative user password
    Enter the password for LDAP administrator.
13. Save the changes.
14. Stop the WebSphere Application Server.
15. Restart the WebSphere Application Server.

Exporting 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 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 them with your target application. See the appropriate instructions:

- If you are using a separate WebSphere Application Server, see "Importing the LTPA key to the WebSphere Application Server" on page 122.
- If you are using an Apache, IHS, or IIS server, see "Copying the LTPA key to the Web server" on page 125.

Chapter 12. Configuring a Web server plug-in

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

Web request processing

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

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

1. Retrieves the Web request URL.
2. Retrieves the LTPA token cookie from the request, if there is one.
3. Checks its configuration to see if the plug-in functionality is enabled. If it is not enabled, processing ends. If it is enabled, the following actions occur:
   a. Checks whether the URL in the request matches any of the URLs that are configured in the plug-in configuration file. This capability enables you to apply specific processing to specific applications.
   b. Identifies the list of HTTP headers to strip from the request. The plug-in configuration file identifies the HTTP headers to strip and prevents attacks in which fake headers are added by the client.
   c. The LTPA token cookie is examined and one of the following actions occurs:
      - If the request does not contain a valid LTPA token cookie, the plug-in identifies the list of session cookies, if any, to strip from the request based on the configuration specified in the plug-in configuration file.

        Cookies are stripped only if the LTPA token cookie is missing, expired, or improperly encoded. Session cookies are present only after a federated single sign-on, which is indicated by the presence of an LTPA token cookie.

        A session cookie without a valid LTPA token cookie implies that the session cookie is no longer applicable. Processing ends.
      - If the request contains a valid LTPA token cookie, the cookie is decoded.

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

        If the LTPA token is decoded successfully, processing continues and the plug-in creates a list of HTTP headers to set in the request. It creates a list by using the configuration specified in the plug-in configuration file and the LTPA attribute values in the token. For information about the LTPA attribute to HTTP header mapping process, see "LTPA-attribute-to-HTTP-header mapping" on page 118.

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

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

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

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

LTPA-attribute-to-HTTP-header mapping

To map the LTPA cookie information to an HTTP header, the plug-in relies on a special configuration file, itfimwebpi.xml, which creates and then modifies or strips (removes) the HTTP headers into the final HTTP request that is sent to the target application.

The following figure shows how the content of the configuration file is used to determine the final HTTP request. Note that the figure shows only an example. LTPA attributes and headers are specific to each application that is used in an environment.

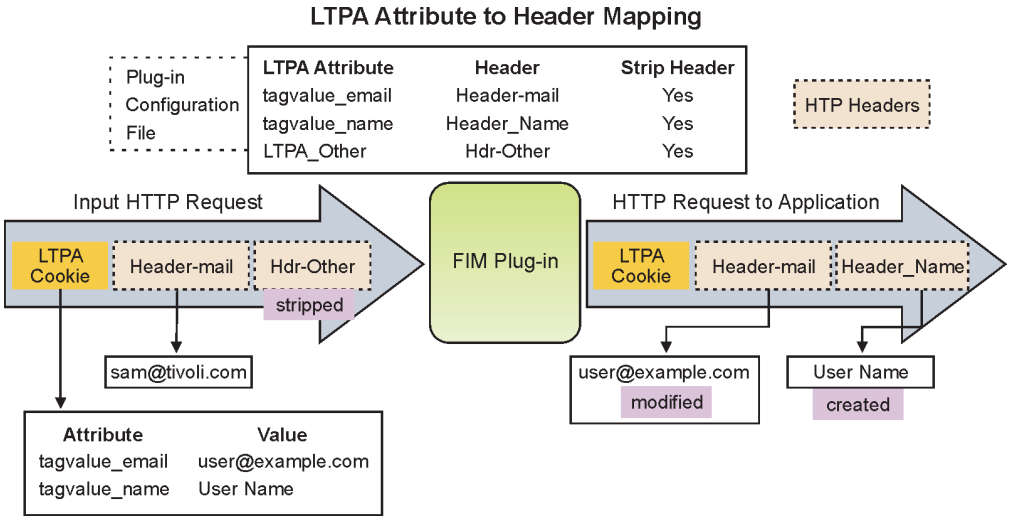


Figure 6. Example of LTPA attribute to HTTP header mapping

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

   For all the headers, if the corresponding LTPA attribute does not exist, any Header with the configured name should be stripped.

For example, in the figure, the LTPA value 'LTPA_Other' is not present, so the input HTTP header 'Hdr-Other' is stripped (removed). The LTPA value 'tagvalue_email' is present, so the existing header 'Header_mail' is modified to contain the value from the LTPA cookie: "user@example.com." The LTPA value 'tagvalue_name' is present, so the header 'Header_Name' is created with the value from the LTPA cookie: "User_Name."

Headers that are not listed in the configuration file remain unchanged. If an LTPA cookie is not present, then all headers with "strip=yes" are removed.

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

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

Configuring service provider components

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

About this task

Complete the following tasks to configure service provider components:

1. Configure the application server that will host your target applications, as described in “Configuring your Web server.”
2. Configure your target application, as described in “Configuring the target application” on page 128.

Configuring your Web server

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

About this task

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

- IBM WebSphere Application Server 5.1 or 6.0 or later

  Note: The servers described here are typically dedicated to hosting a target application. However, you also have the option of hosting your target application on the same instance of WebSphere Application Server where you installed the Tivoli Federated Identity Manager runtime component. Your runtime component must have been installed on either of the following versions of WebSphere:

  – WebSphere Application Server version 6.1
  – The embedded version of WebSphere Application Server 6.1, which came with Tivoli Federated Identity Manager

- Microsoft Internet Information Service 6.0
- IBM HTTP Server 6.1
- Apache HTTP Server 2.0 or 2.2

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

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

Procedure

1. Select and install a user registry for the server, as described in "Selecting and installing a user registry."
2. Configure an SSL connection to the user registry, as described in "Configuring the user registry for the target application."
3. "Configuring an SSL connection to the user registry" on page 121
4. If you host a target application using a WebSphere Application Server that is separate from the server on which Tivoli Federated Identity Manager is installed, complete the steps in "Configuring a separate WebSphere Application Server to host applications" on page 121.
5. If you host a target application using an IIS, IHS, or Apache server, complete the steps in "Configuring an IIS, IHS, or Apache server to host the application" on page 124.

Selecting and installing a user registry

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

Before you begin

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

However, if you use a separate user registry, ensure that it meets the requirements for the server that is hosting your application. See your server documentation for more information.

For example, if you are using WebSphere Application Server to host your application, see the WebSphere Application Server library and locate the information center for the version you are using: http://www.ibm.com/software/webservers/appserv/was/library/. In the appropriate information center, search for the topics about setting up a user registry.

Configuring the user registry for the target application

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

Before you begin

Before continuing with this task, you must already have selected which user registry to use and have installed it as described in "Selecting and installing a user registry" on page 120.

About this task

If you are using the same user registry that you configured for your point of contact server, no further registry configuration is necessary. However, if you are using a separate registry, create users to whom you are providing single sign-on capabilities. These must be the same users that are defined in your point of contact user registry. See the documentation for your user registry for information on adding users.

Configuring an SSL connection to the user registry

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

About this task

If you are using the same user registry for your point of contact server and your target application server, you might have completed this task already. If you are using a separate user registry, see the documentation for that user registry for information about configuring SSL.

What to do next

After you have configured SSL, continue with the appropriate steps for the server on which your target applications are hosted:

- "Configuring a separate WebSphere Application Server to host applications"
- "Configuring an IIS, IHS, or Apache server to host the application" on page 124

Configuring a separate WebSphere Application Server to host applications

In a Tivoli Federated Identity Manager environment, you can host your target applications on the same WebSphere Application Server that is used as your point of contact server or on a separate WebSphere Application Server.

About this task

To configure a separate WebSphere Application Server so that it can host applications that your single sign-on users can access, complete the following tasks:

Procedure

1. Import the LTPA key from your WebSphere Application Server point of contact server as described in "Importing the LTPA key to the WebSphere Application Server" on page 122.
2. Disable the automatic generation of LTPA keys as described in "Disabling the automatic generation of an LTPA key" on page 122.
3. Configure the WebSphere Application Server to use the user registry as described in "Configuring WebSphere to use the user registry" on page 123.

Importing the LTPA key to the WebSphere Application Server

If your target application is hosted on a WebSphere Application Server that is separate from your WebSphere point of contact server, you must import the LTPA key from the point of contact server onto your target application server.

Before you begin

Ensure that you have completed the following steps:

- Make sure the time between the servers is synchronized.
- Copy the LTPA keys from the location where they were exported to a location on your target application server.
- Obtain the password for the LTPA keys. A password was assigned to the keys when they were exported from the WebSphere point of contact server.

Procedure

1. Log on to the console on the *target application server*. Do not log on to your Tivoli Federated Identity Manager console to perform these steps.
2. Select **Security** > **Secure Administration, Applications, and Infrastructure** > **Authentication mechanisms and expiration**.
3. In the **Password** and **Confirm** password fields, enter the password that is used to encrypt the LTPA keys. The password must match the password that was used when the keys were exported.
4. In the **Fully qualified key file name** field, specify the fully qualified path to the location where the LTPA keys are located. You must have write permission to this file.
5. Click **Import keys** to import the keys.
6. Click **OK**.
7. Click **Save** to upload the changes to the master configuration.

What to do next

Disable the automatic generation of LTPA keys on the target application server, as described in "Disabling the automatic generation of an LTPA key."

Disabling the automatic generation of an LTPA key

By default, WebSphere Application Server automatically generates an LTPA key. However, if you are using a WebSphere Application Server other than your point of contact server to host your target application, you must use the LTPA key from your point of contact server on your application server. Therefore, you must disable the automatic key generation so that no conflicts occur.

Before you begin

To complete this task, you must know the name of the key set group and the management scope where the key set group is defined.

Procedure

1. Log on to the console on the *target application server*. Do not log on to your Tivoli Federated Identity Manager console to perform these steps.
2. Click **Security** > **SSL certificate and key management** > **Manage endpoint security configurations.**.
3. Expand the tree to the inbound or outbound management scope that contains the key set group, and then click the scope link.

4. Under Related Items, click **Key Set Groups**
5. Click the key set group that you want to disable.
6. Clear the **Automatically generate keys** check box.
7. Click **OK**
8. Click **Save** to upload the changes to the master configuration.

What to do next

Continue with the steps for "Configuring WebSphere to use the user registry."

Configuring WebSphere to use the user registry

Ensure that the WebSphere Application Server that you are using to host your target application is configured to use the user registry that you selected and installed.

Before you begin

Review the information in "Selecting and installing the user registry" on page 96. Ensure that you have selected and installed the appropriate user registry option for your environment.

About this task

To enable WebSphere to use your user registry, complete these steps:

Procedure

1. Log on to the console *for your target application*. Do not log on to your Tivoli Federated Identity Manager console to perform these steps.
2. Select **Security** > **Secure administration, applications, and infrastructure**. The Configuration tab opens.
3. Click **Security Configuration Wizard** to change the user registry used by the WebSphere runtime. The **Specify extent of protection panel** opens.
4. Verify that the check box **Enable application security** is selected.
5. Click **Next**. The **Secure the application serving environment** panel opens.
6. Select the appropriate option for the user registry to use:
   - **Federated repositories**
   - **Standalone LDAP registry**
   - **Local operating system**
   - **Standalone custom registry**
7. Click **Next**. The **Configure user repository** panel opens.
8. Specify values for each of the registry configuration settings. See the online help for descriptions of the fields presented.
9. Click **Next** and finish the wizard.
10. Save your configuration changes.
11. Stop the WebSphere Application Server.
12. Restart the WebSphere Application Server. You must use the same administrative name you used to log on and make these changes.
13. From the console, select **Tivoli Federated Identity Manager** > **Manage Configuration** > **Domain properties**.
14. In the WebSphere Security section of the panel, update the following values:

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

Administrative user password
Enter the password for LDAP administrator.

15. Save the changes.
16. Stop the WebSphere Application Server.
17. Restart the WebSphere Application Server.

What to do next

When you are done, continue with the appropriate step for your environment:

- If you are hosting applications on an IHS, IIS, or Apache server, continue with "Configuring an IIS, IHS, or Apache server to host the application."
- If you are hosting applications on only your WebSphere server, your server configuration is complete. Continue with the target application configuration in "Configuring the target application" on page 128.

Configuring an IIS, IHS, or Apache server to host the application

If you must host your target applications using a Microsoft Internet Information Services server, an IBM HTTP Server, or an Apache HTTP Server, you must complete specific configuration tasks.

Before you begin

Before continuing with these tasks, you must have installed the plug-in.

Ensure that:

- The plug-in is installed on the server that is hosting the target application.
- The server is in the same domain as the Tivoli Federated Identity Manager server.

Also, ensure that you have completed the steps in "Configuring your Web server" on page 119, including:

- "Selecting and installing a user registry" on page 120
- "Configuring the user registry for the target application" on page 120
- "Configuring an SSL connection to the user registry" on page 121

About this task

Complete the following tasks to prepare your plug-in environment:

Procedure

1. Copy the LTPA key to your server, as described in "Copying the LTPA key to the Web server" on page 125.
2. Create the plug-in configuration file, as described in "Creating the plug-in configuration file" on page 125.
3. Copy the plug-in configuration file to the server, as described in "Copying the plug-in configuration to the server" on page 127.

Copying the LTPA key to the Web server

The LTPA key that is used by WebSphere Application Server on your point of contact server must be shared with the server where the plug-in is installed.

Before you begin

Ensure that you have completed the following tasks:

- Installed the plug-in on the Web server.
- Completed the configuration of your point of contact server, as described in "Configuring a WebSphere Application Server point of contact server (service provider)" on page 111.
- Exported the LTPA keys from your point of contact server, as described in "Exporting LTPA key from the point of contact server" on page 115.
- Verified that the time on the point of contact server and on the server to which you are copying the LTPA key are synchronized.

Procedure

1. Copy the LTPA key, which should be named `ltpa.keys`, from the location to which it was exported.
2. Paste the LTPA key in the webpi directory on the application server. For example:

   On an IHS or Apache server:

   ```
   /opt/IBM/FIM/webpi/etc
   ```

   On an IIS server:

   ```
   C:\Program Files\IBM\FIM\webpi\etc
   ```

What to do next

Continue with "Creating the plug-in configuration file."

Creating the plug-in configuration file

After you have installed the plug-in and prepared your environment to use the plug-in, you must configure it with specific information about the Web applications to be accessed by your single sign-on users.

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 map to which HTTP header or server variable. The HTTP header and server variables are those expected by the target application.
- A list of cookies to remove if the LTPA cookie is missing or is not valid, which usually indicates that the user is not a federated single sign-on user.
- A list of mappings between server variable names and LTPA token attribute names. Server variables are an alternative mechanism for presenting LTPA attributes to the application instead of using HTTP headers.

  Note: The use of server variables is not supported on IIS.

For more information about HTTP header and LTPA attribute mappings and how the plug-in functions in the environment, see Chapter 12, “Configuring a Web server plug-in,” on page 117.

Procedure

1. Log on to the console.
2. Select **Tivoli Federated Identity Manager** > **Manage Configuration** > **Web Server Plugin Configuration**. The Web Plugin Single Sign-on Configuration panel opens.
3. Complete the information that is required for your server in the **Web Server Plug-in Single Sign-on Configuration** and the **Web Server Plug-in Logging Configuration** sections. See the online help for descriptions of the fields.

   Note: Make sure the that the password you specify in the LTPA password field matches the password you created when you exported the ltpa.keys file.
4. When you have completed all of the fields, click **Save**.
5. In the **Web Server Plug-in Applications Configuration**, define an application to the single sign-on configuration by clicking **Create**. 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 applications 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

Continue with “Copying the plug-in configuration to the server” on page 127.

Copying the plug-in configuration to the server

After you have created the plug-in configuration file, you must copy that configuration to your Web server.

Procedure

1. Locate the configuration file that you created using the steps in "Creating the plug-in configuration file" on page 125. The file is named `itfimwebpi.xml` and was created in the directory that you specified when you exported the file.
2. Copy the file and then paste the file in the webpi directory on your Web server:

   On an IHS or Apache server:

   ```
   /opt/IBM/FIM/webpi/etc
   ```

   On an IIS server:

   ```
   C:\Program Files\IBM\FIM\webpi\etc
   ```
3. Restart your Web server for the changes to take effect.

What to do next

The configuration of your server is complete. Continue with the target application configuration in "Configuring the target application" on page 128.

Verifying plug-in configuration on Apache or IBM HTTP Server

After you have configured the plug-in on an Apache HTTP Server or an IBM HTTP Server, you can verify that the configuration was successful.

Before you begin

Before continuing with this task, ensure that you have completed the following tasks:

- "Configuring a WebSphere Application Server point of contact server (service provider)" on page 111
- "Configuring an IIS, IHS, or Apache server to host the application" on page 124

Procedure

1. On the server, locate the httpd.conf file. The location of this file is dependent on your installation. For example:

   ```
   /etc/httpd/conf/httpd.conf
   ```
2. Open the file in a text editor and locate the appropriate line for the plug-in you are using:

   Apache HTTP Server 2.2:

   ```
   LoadModule fimwebpi_module /opt/IBM/FIM/webpi/lib/libitfimwebpi-apache22.so
   ```

   Apache HTTP Server 2.0 or IBM HTTP Server:

   ```
   LoadModule fimwebpi_module /opt/IBM/FIM/webpi/lib/libitfimwebpi-apache20.so
   ```

   Make sure that the webpi module (`libitfimwebpi-apache22.so` or `libitfimwebpi-apache20.so`) has write access to the log file path that is defined in your plug-in configuration file (`itfimwebpi.xml`).

What to do next

Continue with the tasks in "Configuring the target application" on page 128.

Configuring the target application

As the service provider, your role in the federation is to provide a service, such as a Web application, to the user.

About this task

As part of this role, ensure that the application (referred to as the *target application*) that you are providing to the users is configured appropriately for use in a Tivoli Federated Identity Manager environment:

- The application must be able to accept user identity information using an HTTP headers or server variables.
- The Tivoli Federated Identity Manager environment and the application must be in the same DNS domain.
- The application must be hosted by a supported Web server, such as:
  - Microsoft Internet Information Services (IIS) server 6.0, with the Tivoli Federated Identity Manager plug-in installed
  - IBM HTTP Server 6.1, with the Tivoli Federated Identity Manager plug-in installed
  - Apache HTTP Server 2.0 or 2.2, with the Tivoli Federated Identity Manager plug-in installed
  - WebSphere Application Server version 5.1
  - WebSphere Application Server version 6.0 or later

    Note: You can also use the same instance of WebSphere Application Server where you installed the Tivoli Federated Identity Manager runtime component as the server to host your target application. The version of that WebSphere Application Server is either:
    - WebSphere Application Server version 6.1 with fix pack 15
    - The embedded version of WebSphere Application Server, which came with Tivoli Federated Identity Manager

For information about configuring your target application, see the documentation for the server that will host the application. For example, if you are hosting your target application on WebSphere Application Server, see the Information Center for the version of WebSphere Application Server you are using from the library at http://www.ibm.com/software/webservers/appserv/was/library/.

Configuring the login for your application

Before using Tivoli Federated Identity Manager, you probably used a login method that was specific to your application. For example, you might have provided a URL to your users that directed them to a login form or required client authentication. In your Tivoli Federated Identity Manager environment, your identity provider partner is responsible for authenticating users. Depending on the configuration of your federation, you need to either direct your users to a new URL (such as one hosted by your identity provider partner) or by redirecting users from your site to the appropriate login method used by your identity provider partner.

About this task

Discuss login requirements with your identity provider partner. Then, ensure that your environment is configured appropriately to send your users to the appropriate login location.

Instructing users to enable cookies

Users must enable cookies in their browsers when using single sign-on to a service provider who is using WebSphere Application Server as its point of contact server.

About this task

Advise users to follow the instructions for enabling cookies for their browsers.

Chapter 13. Setting up the alias service database

SAML 2.0 supports the use of name identifiers (aliases) for communication of user identities between partners. Aliases are intended to increase the privacy of the user when that user accesses resources at a service provider. When aliases are used, an identifier that both the identity and service provider recognize is sent instead of the actual account name of the user. Aliases are created and recorded during account linkage (federation). After account linkage, the alias is in all messages that are sent between the partners. A different alias is used with each partner. The alias used in one direction, such as from identity provider to service provider, can be different from the alias that is used in the other direction, such as from service provider to identity provider.

About this task

Note: The use of aliases is optional in SAML 2.0.

The default setting for the alias service is to use persistent IDs.

A service in Tivoli Federated Identity Manager, called the *alias service*, generates new aliases, associates aliases with local users, and performs mapping from alias to user and from user to alias.

Most aliases are persistent and must be retained for a long time. Therefore, some type of database must be used to store them. You have two options for the type of database you can use:

- JDBC database, such as the Derby database in WebSphere Application Server
- LDAP database, such as IBM Tivoli Directory Server, which is available separately

The tasks you must perform to set up your alias service database depend on whether you installed the embedded version of WebSphere Application Server, or are using an existing version of WebSphere Application Server with your installation of the Tivoli Federated Identity Manager Runtime and Management Services component.

Embedded version of WebSphere

Your database options are:

- **JDBC database**

  If you installed the embedded version of WebSphere Application Server, a JDBC database, Cloudscape 10, also known as Derby, was configured on WebSphere Application Server to be used for storing alias information. No further tasks for setting up the database are required.

- **LDAP database**

  You have the option of using an LDAP database, such as IBM Tivoli Directory Server, that you have purchased, installed, and configured separately from Tivoli Federated Identity Manager. See the information in "Configuring an LDAP alias service database" on page 134. Then, to use that LDAP database with Tivoli Federated Identity Manager, you must modify the alias service settings, as described in "Modifying alias service settings" on page 134.

Existing version of WebSphere Application Server

Your database options are:

- **JDBC database**

  If you installed Tivoli Federated Identity Manager on an existing version of WebSphere Application Server, and you want to use a JDBC database, you must manually create and configure the database, using a procedure similar to the procedures below for Cloudscape 10 (Derby), as described in "Configuring a JDBC alias service database." (As previously stated, if you installed the embedded version of WebSphere Application Server, these steps were performed automatically and are already completed.)

- **LDAP database**

  You have the option of using an LDAP database, such as IBM Tivoli Directory Server, that you have downloaded, installed, and configured separately from your Tivoli Federated Identity Manager. See the information in "Configuring an LDAP alias service database" on page 134. Then, to use that LDAP database with Tivoli Federated Identity Manager, you must modify the alias service settings, as described in "Modifying alias service settings" on page 134.

Configuring a JDBC alias service database

If you installed Tivoli Federated Identity Manager on an existing version of WebSphere Application Server and you want to use a JDBC database, you must manually create and configure the database. If you installed the embedded version of WebSphere Application Server, these steps were performed automatically and are already completed.

About this task

The following instructions describe how to create and use the JDBC Derby database that is provided with WebSphere Application Server. The Derby database is created by an Apache tool called ij. It is implemented with the Java class org.apache.derby.tools.ij.

Procedure

1. Create the FIMAliases database and import the schema:
   a. Open a command prompt and start the ij tool, which is located in the `/derby/bin/embedded` directory where you installed WebSphere Application Server.

      On AIX, Linux, or Solaris

      Type *$was_home*`/derby/bin/embedded/ij.sh`.

      On Windows

      Type *$was_home*`/derby/bin/embedded/ij.bat`.

   b. From the ij command line, create the database and the schema by typing the following commands:

      ```
      connect 'jdbc:derby:FIMAliases;create=true';
      run '/opt/IBM/FIM/etc/Table.ddl';
      quit;
      ```

      Note: The location of the `Table.ddl` file is in the installation directory of Tivoli Federated Identity Manager. If you used a different installation

directory, use that path with the run command. On Windows, the default installation directory is `C:\Program Files\IBM\FIM`.

2. Verify the database and schema:
   a. Open a command prompt and locate the *$was_home*/derby/FIMAliases directory.
   b. Verify that the SQL file output contains the FIMAliases schema.

      On AIX, Linux, or Solaris
      Type *$was_home*`/derby/bin/embedded/dblook.sh -d jdbc:derby:FIMAliases -o FIMAliase.sql`.

      On Windows
      Type *$was_home*`/derby/bin/embedded/dblook.bat -d jdbc:derby:FIMAliases -o FIMAliases.sql`.

3. Create the Derby embedded JDBC provider and data source:
   a. Open the WebSphere administration console.
   b. Click **Resources** > **JDBC** > **JDBC Providers**.
   c. Set the scope to Cell, for example *Cell=myhostNode01Cell*, from the **Scope** setting list.
   d. Click **New**.
   e. Complete the required fields as follows:

      Database type
      Select **Derby**.

      Provider type
      Select **Derby JDBC Provider**.

      Implementation type
      Select **Connection pool data source**.

      Name Use a name to indicate that this entry is the JDBC provider of the alias service for Tivoli Federated Identity Manager.

      For example, use ITFIM Alias Service JDBC Provider.

   f. Click **Next**.
   g. Click **Finish**.
4. Create a data source for this JDBC provider:
   a. In the WebSphere administration console, click **Resources** > **JDBC** > **JDBC Providers** > **ITFIM Alias Service JDBC Provider** > **Data sources** > **New**.
   b. Complete the required fields as follows:

      Database source name
      Type a name that identifies the data source, such as `ITFIM Alias Service Datasource`.

      JNDI name
      Type `jdbc/IdServiceJdbc`.

      Attention: Use this name exactly as shown so that the mappings between the alias service and the data source occurs automatically.

   c. Click **Next**.
   d. Provide a name for the database, such as `FIMAliases`.
   e. Click **Next**.
   f. Click **Finish**.

5. To verify the connection to the database, select the data source you configured and click **Test connection**.

Modifying alias service settings

You can modify the setting for your name identifier database in the Integrated Solutions Console.

About this task

To modify the setting for your name identifier database:

Procedure

1. Log on to the console. The Alias Service Settings portlet opens.
2. Select **Tivoli Federated Identity Manager** > **Manage Configuration** > **Alias Service Settings**.
3. Select **JDBC Provider and Data Source**

   Use this option if you use a JDBC database to store name identifier information.
4. Click **Apply**.
5. Click **OK**.

Configuring an LDAP alias service database

If you install Tivoli Federated Identity Manager with embedded WebSphere, a JDBC database is the default setting for an alias service database in Tivoli Federated Identity Manager. However, you can use an LDAP database instead.

Before you begin

If you install Tivoli Federated Identity Manager with an existing WebSphere deployment, you might already have an LDAP database in use as a user registry. When using WebSEAL as the point of contact server, you are installing into an environment that includes Tivoli Access Manager. The most common LDAP deployment with Tivoli Access Manager is IBM Tivoli Directory Server.

The Tivoli Federated Identity Manager alias service stores alias information in a user registry. The alias service supports the following user registries:

- IBM Tivoli Directory Server
- Sun ONE

Note:

You can write your own alias service for use with other registries, such as Lotus® Domino® or Microsoft Active Directory.

The alias service requires a location in LDAP to store the necessary information, and the Tivoli Federated Identity Manager runtime and management services feature needs an account on the LDAP server to search for alias information.

If you do not have an LDAP database installed already, you must install an LDAP product to use the alias service.

If you need LDAP, you could use the IBM Tivoli Directory Server product, which can be downloaded from http://www-306.ibm.com/software/tivoli/resource-center/security/code-directory-server.jsp.

About this task

If you use an LDAP database, the following configuration tasks are required:

- "Using tfimcfg to configure LDAP for the alias service"

  Tivoli Federated Identity Manager provides a utility that automates this process, when used with either IBM Tivoli Directory Server or Sun ONE Directory Server.Configuring the LDAP user registry for the alias service
- "Creating an LDAP suffix" on page 138
- "Modifying alias service settings" on page 134

Using tfimcfg to configure LDAP for the alias service

Use the tfimcfg utility to automate the LDAP configuration for the alias service.

About this task

This installation guide instructs you to run **tfimcfg** for the purpose of configuring the alias service.

The tfimcfg uses a data file called ldapconfig.properties to decide which actions to take. You can modify the tfimcfg behavior by using a text editor to modify values in this file. You can specify whether or not specific sets of LDAP properties are defined. For each set that you create, you can specify the values of the individual properties.

In order for tfimcfg to configure LDAP programmatically, the utility must know some LDAP information, such as the LDAP hostname, LDAP port number, and administrator account information. The ldapconfig.properties file contains entries for each of these properties. Default values are provided. You must modify the values to fit your deployment environment.

The following steps list the properties for which you should define a value.

Procedure

1. Obtain a copy of ldapconfig.properties. You can view the file contents by either:
   - Viewing the default file listing in Appendix A, "tfimcfg reference," on page 753
   - Accessing your installation software (CD or installation directory), and viewing the default file:

     AIX, Solaris, or Linux

     ```
     /opt/IBM/FIM/tools/tamcfg/ldapconfig.properties
     ```

     Windows

     ```
     C:\Progra~1\IBM\FIM\tools\tamcfg\ldapconfig.properties
     ```
2. Specify whether tfimcfg adds suffixes to the LDAP server as needed.

   Default:

   ```
   ldap.suffix.add=true
   ```

The tfimcfg utility adds a number of suffixes, based on other settings in the ldapconfig.properties file. If you want to override the creation of any suffixes, set this value to false.

The suffixes that can be created are:

- A suffix for the hierarchy to hold alias service information (user identity aliases)

  Default: cn=itfim
- A suffix for use by the Tivoli Access Manager servers

  Default: secAuthority=Default
- A suffix for a hierarchy for storing user and group information

  Default: dc=com

3. Specify whether tfimcfg creates LDAP containers to store Tivoli Federated Identity Manager server users and groups.

   The Tivoli Federated Identity Manager users and groups are:

   Default:

   ```
   ldap.suffix.user.configuration=true
   ldap.organization.configuration=true
   ```

   - When `ldap.suffix.user.configuration=true`, tfimcfg adds an LDAP suffix `dc=com` and creates an object for it. The utility also sets additional properties as specified in ldapconfig.properties. The list of properties, with their default values, is:

     ```
     ldap.suffix.user.dn=dc=com
     ldap.suffix.user.name=com
     ldap.suffix.user.attributes=dc
     ldap.suffix.user.objectclasses=domain
     ```
   - When `ldap.organization.configuration=true`, tfimcfg sets additional properties. The properties are specified in ldapconfig.properties. The list of properties, with their default values, is:

     ```
     ldap.user.container.dn=cn=users,dc=example,dc=com
     ldap.group.container.dn=cn=groups,dc=example,dc=com
     ldap.organization.dn=dc=example,dc=com
     ldap.organization.name=example
     ldap.organization.attributes=dc
     ldap.organization.objectclasses=domain
     ldap.user.objectclasses=person,organizationalPerson,inetOrgPerson
     ldap.group.objectclasses=groupOfUniqueNames
     ldap.user.shortname.attributes=cn,sn,uid
     ```

   You can use a text editor to modify the values for these LDAP containers.

4. Specify whether tfimcfg creates an LDAP suffix to store single sign-on aliases.

   Default:

   ```
   ldap.suffix.alias.configuration=true
   ```

   When you do not want the utility to specify a new suffix, set this to `false`.

   When this property is set to `true`, tfimcfg uses the value set in the following property:

   ```
   ldap.suffix.alias.dn=cn=itfim
   ```

   You can use a text editor to modify the DN value. This value of this property must begin with `cn=`.

5. Specify whether tfimcfg creates the `secAuthority=Default` suffix for Tivoli Access Manager.

This suffix is used by Tivoli Access Manager to define an LDAP hierarchy for use by the Tivoli Access Manager servers. This suffix is typically created by the Tivoli Access Manager installation scripts. The tfimcfg utility adds the suffix if it does not already exist.

Default:

```
ldap.suffix.tam.configuration=true
```

- When you have already configured Tivoli Access Manager set this value to `false`.
- When Tivoli Access Manager is not using this LDAP server, set this value to `false`.

Note: When the secAuthority=Default suffix already exists, the tfimcfg program ignores the value of the ldap.suffix.tam.configuration property.

6. Specify whether tfimcfg configures LDAP for the Tivoli Federated Identity Manager alias service.

   Default:

   ```
   ldap.fim.configuration=true
   ```

   Default value: `true`.

   When this value is true, tfimcfg sets the following properties, as specified in ldapconfig.properties:

   - The distinguished name, short name, and password that the Tivoli Federated Identity Manager server (runtime and management service) uses to bind to the LDAP server. Defaults:

     ```
     ldap.fim.server.bind.dn=uid=fimserver,cn=users,dc=example,dc=com
     ldap.fim.server.bind.shortname=fimserver
     ldap.fim.server.bind.password=passw0rd
     ```

   - The distinguished name and short name for the group to which the user identity for the Tivoli Federated Identity Manager server (fimserver) belongs. Defaults:

     ```
     ldap.fim.admin.group.dn=cn=fimadmins,cn=groups,dc=example,dc=com
     ldap.fim.admin.group.shortname=fimadmins
     ```

   The tfimcfg utility then adds the user:

   ```
   uid=fimserver,cn=users,dc=example,dc=com
   ```

   to the group:

   ```
   cn=fimadmins,cn=groups,dc=example,dc=com
   ```

7. Specify whether tfimcfg attaches appropriate ACLs (access control lists) to the LDAP server.

   Default:

   ```
   ldap.modify.acls=true
   ```

   When this is set to `false`, you must attach the ACLs manually.

   These ACLs grant read and write access to the Tivoli Federated Identity Manager administrative users created by tfimcfg.

   For example, when `ldap.modify.acls=true`, and tfimcfg is run using the default values for creation of suffixes, ACLs are set for the following suffixes:

   - cn=itfim
   - secAuthority=Default
   - dc=com

   Note: tfimcfg attaches ACLs for IBM LDAP and Sun ONE servers. For other LDAP servers, you must attach the ACLs manually.

8. Specify values for each property that describes your LDAP deployment.

   Default values are provided for most properties. Modify the properties to match your deployment. When LDAP security is enabled, enter the name of the Java keystore that contains the certificate used for SSL, and enter the password to be used by the Tivoli Federated Identity Manager management service.

Table 13. LDAP properties to modify for tfimcfg

Property	Description	Your value
ldap.hostname	The system that hosts the LDAP server. Default value is `localhost`.	
ldap.port	The LDAP port. Default value is `389` for non-SSL communication.	
ldap.admin.dn	LDAP administrator name. Default: `cn=root`	
ldap.admin.password	The password for the LDAP administrator	
ldap.security.enabled	Boolean value that specifies whether LDAP security is enabled. This value is disabled by default.	
ldap.security.trusted.jks.filename	The name of the Java keystore that contains the signer of the LDAP-presented SSL certificate that LDAP presents during trusted communications. There is no default entry.	
ldap.fim.server.bind.password	The password for servers that communicate with the LDAP servers. Change the default to values used in your deployment.	

9. To configure the LDAP server, see Appendix A, "tfimcfg reference," on page 753.

Creating an LDAP suffix

You must create an LDAP suffix, such as `cn=itfim`, to enable the alias service to access the LDAP user registry.

Before you begin

The following instructions are for IBM Tivoli Directory Server. Make sure you have installed IBM Tivoli Directory Server and completed initial configuration as described in its documentation before continuing with the following steps.

Procedure

1. Stop the IBM LDAP server.

   AIX, Linux, or Solaris:

   ```
   # ibmdirctl -D cn=root -w passw0rd stop
   ```

   Windows

   Use the Services icon.

2. Add the suffix: `# idscfgsuf -s "cn=itfim"`.
3. Start the IBM LDAP server.

   AIX, Linux, and Solaris:

   ```
   # ibmdirctl -D cn=root -w passw0rd start
   ```

   Windows

   Use the Services icon.

4. Use **ldapmodify** to update the LDAP schema file. For example, on Linux:

```
ldapmodify -D cn=root -w passw0rd -f
  /opt/IBM/FIM/etc/itfim-secuser.ldif
```

Planning configuration of the alias service properties

Use these instructions to specify the alias service properties for accessing one or more LDAP servers.

About this task

The alias service manages aliases by accessing an LDAP user registry. The alias service needs to know a number of pieces of information about the LDAP environment in which it operates. The management console provides a GUI interface that you can use to specify the necessary properties. The properties are stored in a Tivoli Federated Identity Manager property file specific to the current Tivoli Federated Identity Manager domain.

This topic describes the properties that you need to specify. It also provides a worksheet that you can use to enter the values for your environment. For many properties, you can use a default value.

The value to set for some of the properties correspond to values that you specified previously, when you planned the use of the tfimcfg utility. At that time, you identified values to edit in the file ldapconfig.properties. The tables in the following task sequence identify the GUI fields with values that should match the properties in ldapconfig.properties.

Procedure

1. Determine the value for the LDAP search property.

   The following table describes the Root suffix, the LDAP search property that is configurable through the GUI. You can expedite the configuration by identifying at this time the appropriate value for your deployment environment.

Table 14. LDAP Search property

Property	Description	Your value
Root suffix	Specifies the root suffix where alias settings are written. This property can have one value (suffix) only. The value of this property matches the value for the following property in ldapconfig.properties: `ldap.suffix.alias.dn` For example: `cn=itfim`.	

2. Determine values for LDAP environment properties.

Table 15. LDAP environment properties

Property	Description	Your value
SSL Enabled	A check box that specifies whether communication between the alias service and the LDAP servers should be secured using Secure Socket Layer (SSL). When the LDAP servers are configured to use SSL, the alias service must use SSL when communicating with them. This value of this property corresponds to the value for the following property in ldapconfig.properties: `ldap.security.enabled` When using SSL, the **SSL Enabled** check box should be selected, and the value of ldap.security.enabled should be `true`.	
Keystore	When the **SSL Enabled** check box is selected, you must select a keystore from the **Keystore** menu list. The selected keystore is the name of the trusted keystore containing the CA certificate of the LDAP server. **Note:** The certificate authority certificates for all LDAP servers must be in the same keystore. This value of this property corresponds to the value for the following property in ldapconfig.properties: `ldap.security.trusted.jks.filename`	

3. Determine values for LDAP server properties

Table 16. LDAP server properties

Property	Description	Your value
LDAP Hostname	The **LDAP Hosts** box lists the configured servers in order of preference. The alias service tries first to contact the server at the start (top) of the list. If that contact is unsuccessful, the alias service attempts to contact the next server on the list. This value of this property includes the value for the following property in ldapconfig.properties: `ldap.hostname` The ldapconfig.properties file holds only one value for this property, but you can specify multiple values for LDAP Hostname.	
Port	The port on which the LDAP server listens. This value of this property matches the value for the following property in ldapconfig.properties: `ldap.port` Default port for non-SSL communication: `389` Default port for SSL communication: `636`	
Bind DN	The distinguished name (DN) that the alias service uses to bind to the LDAP server. This value of this property matches the value for the following property in ldapconfig.properties: `ldap.fim.server.bind.dn` The GUI panel provides a default of cn=root. However, root access is not required to complete the bind. You can specify the DN of the alias service. Default value: `uid=fimserver`	

Table 16. LDAP server properties (continued)

Property	Description	Your value
Bind Password	The password for the DN specified in the **Bind DN** field. This value of this property matches the value for the following property in ldapconfig.properties: `ldap.fim.server.bind.password`	
Mode	The default is read-write. When you configure multiple LDAP servers, typically only one should be read-write. In this scenario, the other LDAP servers are typically deployed for failover purposes, and are expected to have read-only copies of the user registry.	
Minimum number of connections	The initial number of connections (binds) for the alias service to establish to the LDAP server. The minimum valid number is zero (0). The maximum valid number is limited only by the maximum value supported by the data type. The default value is 2. Use the default value unless you have a specific need to increase it.	
Maximum number of connections	The maximum number of connections (binds) for the alias service to establish to the LDAP server. The maximum valid number is limited only by the maximum value supported by the data type. The default value is 10. Use the default value unless you have a specific need to increase it.	

Modifying alias service settings for LDAP

Learn about modifying the setting for your name identifier database

Procedure

1. Log on to the console. The Alias Service Settings portlet opens.
2. Click **Tivoli Federated Identity Manager** > **Manage Configuration** > **Alias Service Settings**.
3. Select **LDAP**.

   Specify the properties that you put in the worksheet in "Planning configuration of the alias service properties" on page 139
4. If you select SSL for communication with the LDAP, select the name of the trusted keystore containing the CA of the LDAP server. If you have not moved the LDAP CA to the Tivoli Federated Manager key service yet, you can retrieve the certificate over SSL as follows:
   a. In the console, select **Tivoli Federated Identity Manager** > **Key Service**.
   b. Select the truststore where you want to store the certificate in the Keystore table. The View Keys button is activated.
   c. Click **Retrieve Certificate from SSL Connection**. The Password panel opens.
   d. Type your truststore password.
   e. Click **OK**.
   f. Complete the fields to specify the host name and port name from which you retrieve the certificate.

      (Optional) Click **Show Signer Info** to view the certificate before retrieving.

g. Complete the **Alias** field with the name you want to use for the certificate. Then, click **OK**. The certificate is added to the truststore.

5. Click **Apply**.
6. Click **OK**.

Configuring Oracle alias service database

You can configure IBM Tivoli Federated Identity Manager to use Oracle as the alias service.

About this task

The instructions describe how to configure IBM Tivoli Federated Identity Manager for use of Oracle as the alias service.

Procedure

1. Create a backup of the itfim.ear file. Use the commands:

   ```
   mkdir /tmp/work
   rm FIM_INSTALL_DIR/pkg/release/itfim.ear
   ```

2. Modify the EAR to be Oracle-aware for deployment. Use the commands:

   ```
   mkdir /tmp/work
   rm FIM_INSTALL_DIR/pkg/release/itfim.ear
   WEBSPHERE_INSTALL_DIR/AppServer/bin/ejbdeploy.sh FIM_INSTALL_DIR/pkg/
   release/itfim-orig.ear /tmp/work FIM_INSTALL_DIR/pkg/release/itfim-
   oracle.ear -dbschema FIMAliasesSchema -dbname FIMAliases -dbvendor
   ORACLE_V10G -trace
   cp FIM_INSTALL_DIR/pkg/release/itfim-oracle.ear FIM_INSTALL_DIR/pkg/
   release/itfim.ear
   rm -rf /tmp/work
   ```

3. A new `FIM_INSTALL_DIR/pkg/release/itfim.ear` is available for deployment to work with Oracle. Use a text editor to update the file FIM_INSTALL_DIR/pkg/software.properties to change the property `com.tivoli.am.fim.rte.software.serialId` to a different value for example, increment.
4. Use the IBM Tivoli Federated Identity Manager console to navigate to **Tivoli Federated Identity Manager** > **Domain Management** > **Runtime Node Management** A message indicating that a new run time is available for deployment.
5. Use the console to deploy the new run time.
6. Restart the WebSphere process where the run time is deployed.

Chapter 14. Planning the mapping of user identities

Plan the mapping of user identities appropriate to your deployment.

Task overview:

1. Read this series of topics on the mapping of user identities
2. Review the default mapping rules files for your protocol. Decide if you can use them, either as they are, or by making your own modifications as appropriate for your deployment.
3. If the requirements for your deployment cannot be met by the use of a mapping rule, you can choose one of the succeeding options:
   - Use the Tivoli Directory Integrator mapping module that is provided withTivoli Federated Identity Manager.
   - Develop a custom mapping module.

A primary function of the Tivoli Federated Identity Manager trust service is the transfer of user identity information (credentials) between partners in a single sign-on federation. This transfer requires changing user identity information formats several times to move between formats local to each partner and the agreed token format for exchanging credentials.

The identity information transfer includes an identity mapping step where the user information is mapped from the structure provided by one credential or token type, into the required structure by another token type.

To complete this mapping step, choose one of the succeeding options:

- Write an identity mapping rule
- Deploy the Tivoli Directory Integrator mapping module

  Use of this module requires an understanding of the Tivoli Directory Integrator features and configuration. See the product documentation for Tivoli Directory Integrator.

  Tivoli Federated Identity Manager provides a user interface for setting some configuration properties. See “Tivoli Directory Integrator identity mapping module” on page 151.
- Develop a custom mapping module.

  Building your own module that is tailored to the needs of the applications in your deployment is a development task. See “Creating a custom mapping module” on page 162.

If you choose to write an identity mapping rule, use the eXtensible Stylesheet Language (XSL), and save it to disk as an XSL file. When you create a federation, the federation wizard prompts you to supply the name of your mapping rule file. The wizard imports this file into the configuration for the federation.

Each identity mapping rule file is specific to a particular role and a particular federation. For example, when you create a SAML federation for an identity provider, use a different mapping rule from the rule you use to create a SAML federation service provider. The identity mapping rule for a Liberty federation is also different from the mapping rule for a SAML federation on an identity provider.

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

You must create and save a mapping rule file before you create a federation.

Note: An identity mapping rule specifies the attributes that are associated with a user credential. Users can access multiple applications after they authenticate, so you must make sure that your rule sets the appropriate attributes for all of the applications that the user accesses.

The Tivoli Federated Identity Manager management console provides a Federation wizard that guides you through the configuration of a single sign-on federation. The wizard contains an Identity Mapping panel, which prompts the administrator to supply the name of an identity mapping rule file. The wizard imports the file, and uses it when building the configuration for the trust module chain that is specific to the federation.

The administrator must create the identity mapping file before using the wizard to configure the federation. The wizard panel expects that the administrator has created an eXtensible Stylesheet Language (XSL) file that describes identity mapping rules. The identity mapping rules are used to convert information that must move across the federation between the partners (identity provider and service provider). Each identity mapping rule must provide:

- The information structure that is required by the security token to be generated.
- The information content (identity attributes) that is required by applications that use the federation.

To write an identity mapping rule, you must understand:

- The role of the identity mapping module.
- The expression of user identity information in XML files.
- The use of the XSL language to specify rules for manipulating the user identity information.

Identity mapping overview

When exchanging security tokens with partners, it is not enough to understand the different token standards. It is important to know what information a particular partner is expecting in tokens from your site, and what information you expect to receive from partners. You can use the Tivoli Federated Identity Manager identity mapping and trust service to customize the format and content of incoming and outgoing tokens to meet the requirements of each partner.

In a single sign-on federation, an identity provider authenticates the user. This authentication creates user credentials in the identity provider environment. For example, an identity provider might require users to authenticate with a user name and password. The user information is validated against the user registry of the identity provider. Then, a local credential that contains group membership data along with optional attributes about the user is created.

In the most typical use case of SAML 2.0, the user name is not carried by the assertion. Instead, the user is represented by an alias.

A service provider also has specific requirements for its user credentials, which users must access applications. In many cases, the credentials that the service provider requires, differ in format or content from the credentials created by the

identity provider. For example, the service provider might want a specific attribute to be included in the credential, such as the social security number of the user. Therefore, the identity credentials must be mapped between the identity provider and the service provider.

In Tivoli Federated Identity Manager, on the identity provider side, the locally authenticated user (the input identity) can be mapped to a different user before the creation of the single sign-on token (the output identity). Similarly, on the service provider side, the identity that is received from the sign-on token (the input identity) can be mapped to a local identity that is needed for access to service provider applications (the output identity). The mapping process is shown in Figure 7.

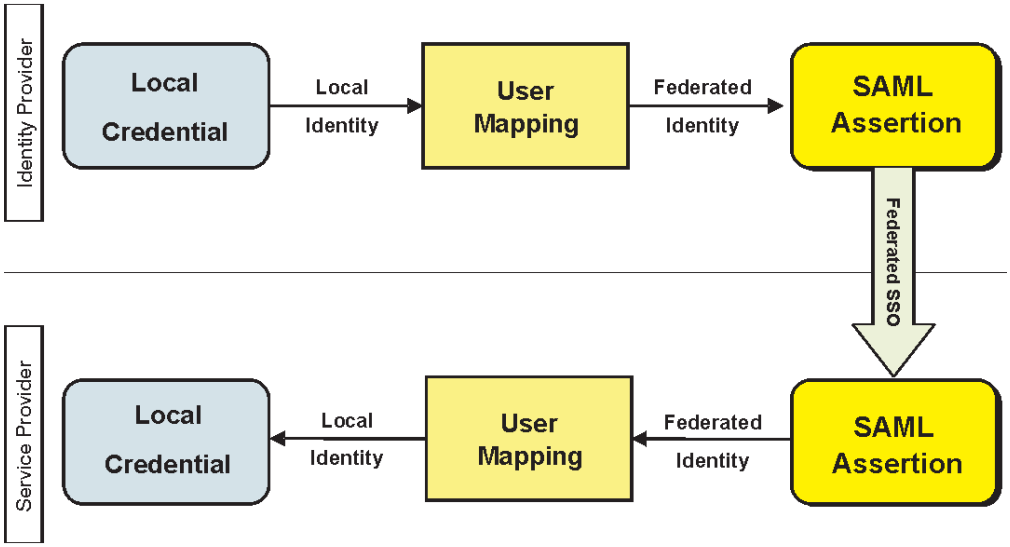


Figure 7. Example of identity mapping

Several methods can be used during the user mapping activity to achieve the required output identity. For example, hard-coded information can be added to the outgoing token. Or Java code can be developed and used to acquire information from external sources and that information can be added to the outgoing token. This flexibility is achieved through *identity mapping rules* that are defined in either of two ways:

- An eXtensible Stylesheet Language Transformation (XSLT) file and processed by the Tivoli Federated Identity Manager Identity Mapping module.
- A custom mapping module that you create using Java.

Before you decide which method to use, you must understand the following factors:

- how user identities are represented in Tivoli Federated Identity Manager
- how tokens are processed, and
- how identities are mapped between partners.

Security Token Service (STS) Universal User document

To ensure that an incoming token can be converted properly into an outgoing token that contains the content and format that is required by the partner, Tivoli Federated Identity Manager creates an intermediate document in a generic XML

format that holds identity information. This document is called the STS Universal User or STSUU. The STSUU document contains three sections:

- Principal information
- Group information
- Attribute information

To create the STSUU document, Tivoli Federated Identity Manager uses an XML schema that specifies the structure. The schema is defined in the file stsuuser.xsd. The following code sample contains the entire contents of the secure token service universal user XML schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:ibm:names:ITFIM:1.0:stsuuser"
xmlns:stsuuser="urn:ibm:names:ITFIM:1.0:stsuuser"
elementFormDefault="qualified">

 <xsd:element name="STSUniversalUser">
  <xsd:complexType>
   <xsd:sequence>
     <xsd:element name="Principal" type="stsuuser:PrincipalType"
             minOccurs="1" maxOccurs="1"/>
     <xsd:element name="GroupList" type="stsuuser:GroupListType"
             minOccurs="0" maxOccurs="1"/>
     <xsd:element name="AttributeList" type="stsuuser:AttributeListType"
             minOccurs="0" maxOccurs="1"/>
     <xsd:element name="RequestSecurityToken" type="stsuuser:RequestSecurityTokenType"
             minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
   <xsd:attribute name="version" type="xsd:string" use="required"/>
  </xsd:complexType>
 </xsd:element>

 <xsd:complexType name="PrincipalType">
  <xsd:sequence>
   <xsd:element name="Attribute" type="stsuuser:AttributeType"
         minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="RequestSecurityTokenType">
  <xsd:sequence>
   <xsd:element name="Attribute" type="stsuuser:AttributeType"
         minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="AttributeType">
                 <xsd:sequence>
                  <xsd:element name="Value" type="xsd:string"
                        minOccurs="0" maxOccurs="unbounded"/>
                  </xsd:sequence>
                 <xsd:attribute name="name" type="xsd:string" use="required"/>
                 <xsd:attribute name="type" type="xsd:string" use="optional" />
                 <xsd:attribute name="nickname" type="xsd:string" use="optional" />
                  <xsd:attribute name="preferEncryption" type="xsd:boolean" use="optional" />
 </xsd:complexType>

 <xsd:complexType name="AttributeListType">
  <xsd:sequence>
   <xsd:element name="Attribute" type="stsuuser:AttributeType"
         minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="GroupListType">
  <xsd:sequence>
   <xsd:element name="Group" type="stsuuser:GroupType"
         minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="GroupType">
  <xsd:sequence>
   <xsd:element name="Attribute" type="stsuuser:AttributeType"
         minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required" />
  <xsd:attribute name="type" type="xsd:string" use="optional" />
 </xsd:complexType>

</xsd:schema>
```

Figure 8. STS Universal User document schema

Although the schema is used as the base for all STSUU documents, the exact information contained in any specific STSUU document is dependent on the token

type for the security token that was used as input. The information required in an STSUU document after transformation by identity mapping depends on:

- The token type to be generated.
- The specific mapping rule being used for the conversion.

During token processing for a typical single sign-on configuration, two STSUUs are created. One is an input STSUU, which is created from the original input token. The other is an output STSUU, which is created after the identity mapping rules are applied. For more information, see "Token processing."

Token processing

In a typical single sign-on configuration, tokens are processed by the Tivoli Federated Identity Manager trust service and three specific types of modules. When used in combination, the modules are referred to as a *trust chain*. Figure 9 provides a diagram of token processing. The input to the trust chain is the input security token. This token is created using the local credentials that are received when a user logs in.

The first module in the trust chain converts the input token to an STSUU document. All attributes that are in the input token are available in the STSUU document. The STSUU document is now used as input to the identity mapping module. This module can be the Tivoli Federated Identity Manager mapping module that is used with an XSLT file. Or it can be a custom mapping module that you have created.

A given mapping module might be used in common for many partners in the federation or one that is unique to a specific partner. The output of the mapping module is another STSUU document. This "output" STSUU document is used as input to the third token module, which converts the "output" STSUU to the output token. The output token is then sent to the partner.

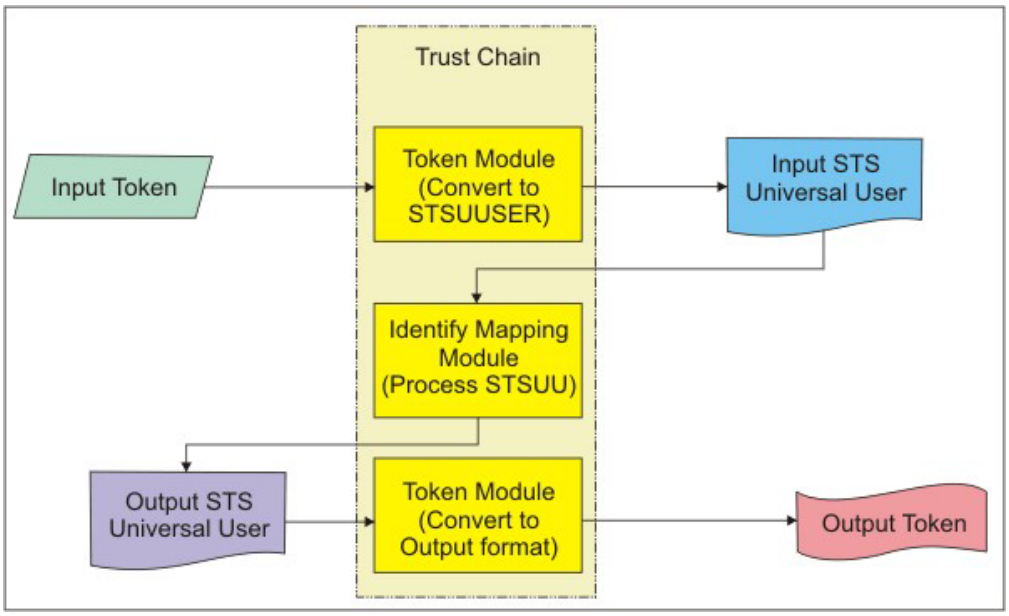


Figure 9. Token processing

Use of XSL language for creating mapping rules files

The identity mapping module uses the Java API for XML Parsing (JAXP) to transform the input document. The transformation is done based on XSL stylesheet that you specify in an XSL file.

XSL is a language that can be used to transform and format documents. XSL is used to define style sheets for HTML and to format XML data, so that it can be shown in a web browser. Part of the XSL standard defines transformations for moving data from one form to another. The transformation language can include conditional statements, variables, and callouts to Java programs.

The trust service uses XSL to create mapping rules. The mapping rules created specify how to transform an input STS universal user document into an output STS universal user document. The output STS universal user document is used as input to the next module in the chain. This module is often used to generate an output token, but it can also be another mapping module. The XSL parser processes XSL documents by looking for matching templates. When a template is found, the contents of the template are processed.

The main tasks that are performed in mapping rules are:

- Moving identity information between elements.
- Reformatting existing identity information.
- Adding new elements with new identity information.
- Removing unwanted identity information.

You can use the IBM Rational® Application Developer tool set to run an XSL debugger from a command line. Use the developer tool set to test your XSL code without running the trust service.

Tivoli Federated Identity Manager provides two sets of sample identity mapping files. The first set shows the minimum contents of each type of mapping, while the second set does advanced functionality that is used in the demonstration application.

The location of the basic mapping files is:

```
/opt/IBM/FIM/examples/mapping_rules/
```

The following table lists the example mapping rules files.

Table 17. Example mapping rules

File name	Mapping description
ip_liberty.xsl	Maps a Tivoli Access Manager credential or a local user identity to a Liberty token.
ip_saml_1x.xsl	Maps a local user identity to a SAML 1.0 or SAML 1.1 token.
ip_saml_20.xsl	Uses a token to maps a local user identity to a SAML 2.0 token.
ip_saml_20_email_nameid.xsl	Uses the email address of the user for the identity without an alias, to map a local user identity to a SAML 2.0 token .
ip_wsfederation.xsl	Maps a Tivoli Access Manager credential or a local user identity to a SAML token.

Table 17. Example mapping rules (continued)

File name	Mapping description
ip_infocard.xsl	Maps an incoming token or a local user identity to a SAML 1.1 token. The primary purpose of this rule is to populate the requested claims attributes with values.
ip_openid.xsl	Maps an IVCred token or a local user identity to a Security Token Service Universal User (STSUU) token. The primary purpose of this rule is to populate requested attributes (SREG and AX) and to act on requested PAPE policies.
rp_infocard.xsl	Maps a SAML 1.1 token or a local user identity to an IVCred token.
sp_liberty.xsl	Maps a Liberty token to a Tivoli Access Manager credential or a local user identity.
sp_saml_20.xsl	Maps a SAML 2.0 token to a local user identity.
sp_saml_1x.xsl	Maps a SAML 1.0 or 1.1 token to a local user identity.
sp_saml_1x_ext.xsl	Maps a SAML 1.0 or 1.1 token to a local user identity and verifies that the authentication method is an acceptable one. It demonstrates that the service provider can require the authentication at identity provider to be at a certain level. In this mapping rule, password authentication is not accepted. It produces an error if password authentication was used.
sp_wsfederation.xsl	Maps a SAML token to a Tivoli Access Manager credential or a local user identity.
sp_tagvalue.xsl	Maps a SAML token to a Tivoli Access Manager IV Cred credential with WebSEAL tag/value attributes or a local user identity.
username_ivcred.xsl	Maps a Username token to a Tivoli Access Manager credential or a local user identity.
sp_oauth_10.xsl	Supports OAuth 1.0 flow.
sp_oauth_20.xsl	Supports OAuth 2.0 flow.

Note: For more information about the sample mapping rules for each protocol, see the protocol-specific configuration instructions in this guide.

The demonstration application provides sample XSL identity mapping rules files. These files expand upon the minimal mapping rules described in the preceding table to perform mapping that is customized for the user accounts. The demonstration application configuration scripts create the user accounts.

The location of the sample mapping scripts for the demonstration application is:
`/opt/IBM/FIM/examples/demo/demo_rules/`

Note: The file names are the same as the minimal mapping rules, but the files are located in different directories.

The sample mapping files are automatically installed during installation.

The following table lists the files for each federation type on each provider type.

Table 18. Sample mapping rules files for the demonstration application

Provider	Federation Type	Mapping rule file
Identity Provider	Liberty	`ip_liberty.xsl`
	SAML 1.0	`ip_saml_10.xsl`
	SAML 1.1	`ip_saml_11.xsl`
	SAML 2.0	`ip_saml_20.xsl`
	WS-Federation	`ip_wsfederation.xsl`
	Information Card	`ip_openid.xsl`
	OpenID	`ip_infocard.xsl`
Service Provider	Liberty	`sp_liberty.xsl`
	SAML 1.0 or 1.1	`sp_saml_1x.xsl`
	SAML 2.0	`sp_saml_20.xsl`
	WS-Federation	`sp_wsfederation.xsl`
	Information Card	`rp_infocard.xsl`
	OpenID	`sp_openid.xsl`
	OAuth 1.0	`sp_oauth_10.xsl`
	OAuth 2.0	`sp_oauth_20.xsl`

Tivoli Directory Integrator identity mapping module

This module performs generic user and attribute mapping functions.

An assembly line executing on a Tivoli Directory Integrator 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 the server, including LDAP and relational databases. Custom code is also supported through JavaScript connectors.

Tivoli Federated Identity Manager provides a demonstration Tivoli Directory Integrator mappings file. The file is located with the other example files. For example, on Linux or UNIX, the file location is

`/opt/IBM/FIM/examples/tdi_mappings/tdi_demo_mappings.xml`

Deployment of this module requires:

- Configuration of the Tivoli Directory Integrator trust module settings
- Configuration of the Tivoli Directory Integrator server
- Configuration of SSL communication between the Tivoli Directory Integrator server and the client, also known as the trust module

Complete the configuration instructions in:

- "Configuring the Tivoli Directory Integrator trust module"
- "Configuring the Tivoli Directory Integrator Server" on page 153
- "Configuring SSL for Tivoli Directory Integrator trust module" on page 155

Configuring the Tivoli Directory Integrator trust module

You must supply the configuration properties for the Tivoli Directory Integrator security token module during the creation of a trust chain.

The properties are described in this topic, and a worksheet is provided for you to consult when you use the administration console to configure your module.

Configuration properties

Server Hostname
Host name or IP address of the computer on which the Tivoli Directory Integrator server is running. The default value is localhost. For example, `tdiserver.company.com`

Server Port
Port number on which the Tivoli Directory Integrator server is configured to run. The default value is 1099.

Assembly Line Handler Pool Size
Number of assembly line handlers to maintain for this trust chain. The value must be a positive integer. The default value is 10.

Number of Wait Threads
Maximum number of threads that can be waiting for an assembly line handler for this chain. The value must be a positive integer. The default value is 0.

Amount of time for threads to wait for an assembly line handler to become available
Determine the amount of time for threads to wait for an assembly line handler to become available. Select one of these options.

Wait indefinitely
Do not put a limit on the time for threads to wait for the assembly line handler to become available. This option is the default choice.

Do not wait for assembly line handler after initial try
Threads must not wait for an assembly line handler. If an assembly line handler 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)
The maximum time a thread waits 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. When connected, you can 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 is derived from the configuration file you have selected in the preceding field.

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
Select the Work Entry attribute identification format. Select one from the following options:

Attribute name
The Work Entry is going to use the name to identify its attributes.

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

Table 19. Tivoli Directory Integrator Module configuration properties worksheet

Property	Your value
Server hostname	
Server Port	
Assembly Line Handler Pool Size	
Number of Wait Threads	
Amount of time for threads to wait for an assembly line handles to become available	Configuration panel provides 3 options: • Wait indefinitely • Do not wait for assembly line handler after initial try • Use the following maximum wait value: Maximum Wait Time (milliseconds)
Method for selecting the assembly line settings	2 choices: • Discover configuration settings • Enter configuration settings manually
Configuration file	
Assembly Line Name	
Identification format for the Work Entry attributes	2 choices: • Attribute name • Attribute name and attribute type

Configuring the Tivoli Directory Integrator Server

This topic contains the minimum procedure required for configuring a default installation of the Tivoli Directory Integrator (TDI) server. This procedure applies to the TDI server versions 6.1.1, 7.0, and 7.1.

About this task

Configure the Tivoli Directory Integrator server to run assembly lines with Tivoli Federated Identity Manager and the TDI STS module. The `tdi_demo_mappings.xml` file is used as an example configuration.

For more detailed TDI configuration instructions, see the Tivoli Directory Integrator Information Center: http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.IBMDI.doc_7.1/welcome.htm

The TDI installation prompts you to select a solutions directory from the following choices:

- TDI subdirectory under your home directory (default)
- Installation directory
- Select a directory to use

This procedure assumes the use of TDI server version 7.1, and the default solutions directory.

Procedure

1. Establish solution files.

   After the initial installation, a subdirectory under the home directory of the root user: `/root/TDI` is created. This directory originally does not contain any solution files.

   To populate solution files in the `/root/TDI` directory, start the TDI server without any parameters:

   ```
   # /opt/IBM/TDI/V7.1/ibmdisrv
   ```

   Another option is to start the TDI configuration editor. After the server starts, solution files are generated in the `/root/TDI` directory, including the `solutions.properties` file.

2. Update the `solutions.properties` file.

   `api.remote.on`
   This property allows the use of the remote server API used by the TDI STS module. Change the default value from `false` to `true`.

   `api.remote.ssl.on`
   These instructions show the TDI configuration without SSL. The SSL configuration is stated in the *Configuring SSL for Tivoli Directory Integrator trust module* section. Change the default value from `true` to `false`.

   `api.remote.nonssl.hosts`
   This property is needed when the TDI server is running on a different host from the Tivoli Federated Identity Manager runtime, and when SSL is not used. Specify the IP address of machine running the runtime (trust server).

3. Establish and populate the TDI configuration directory.

   The `solutions.properties` file contains a setting which describes the location for the TDI configuration files you can edit through the server API. This property, and its default value, is:

   ```
   api.config.folder=/opt/IBM/TDI/V7.1/configs
   ```

   You can choose a different directory. Ensure that the directory exists, and that it has the configuration file that you want the TDI STS module to use. For example, create a directory and copy the sample TDI configuration file into that directory, as follows:

   ```
   # mkdir /opt/IBM/TDI/V7.1/configs
   # cp /opt/IBM/FIM/examples/tdi_mappings/tdi_demo_mappings.xml
       /opt/IBM/TDI/V7.1/configs
   ```

4. Start the TDI server in daemon mode.

   Enter this command to start the server without SSL support:

   ```
   # /opt/IBM/TDI/V7.1/ibmdisrv -d
   ```

The server must be running, and log information can be viewed in `/root/TDI/logs/ibmdi.log`.

Results

The TDI STS module is now able to load and run assembly lines.

Configuring SSL for Tivoli Directory Integrator trust module

The Tivoli Directory Integrator Security Token Service module acts as a client to the Tivoli Directory Integrator server. Configuration of SSL communication between the two can be done in a number of different ways. Configuration of the server and the client is done separately.

See the Tivoli Directory Integrator Information Center for details about security in general, and SSL configuration. There are several server API authentication scenarios available, but this document describes only mutually authenticated SSL. This scenario is the supported deployment pattern for secured installations with the Tivoli Directory Integrator Security Token Service module.

Configuring SSL for the Tivoli Directory Integrator Server

Learn how to configure mutually authenticated SSL for Tivoli Directory Integrator server versions 6.1,1, 7.0, and 7.1.

About this task

This topic contains the following information:

- One of the various ways of configuring SSL.
- One possible scenario when configuring SSL for the Tivoli Directory Integrator server.

Complete information can be found in the Tivoli Directory Integrator Information Center:

- For Tivoli Directory Integrator version 6.1.1

  Tivoli Directory Integrator version 6.1.1 information center. See the topic on *Secure Sockets Layer (SSL) Support* from the *Administration Guide*.
- For Tivoli Directory Integrator version 7.0

  Tivoli Directory Integrator version 7.0 information center. See the topic on *Secure Sockets Layer (SSL) Support* from the *Installation and Administration Guide*.
- For Tivoli Directory Integrator version 7.1

  Tivoli Directory Integrator version 7.1 information center. See the topic on *Secure Sockets Layer (SSL) Support* from the *Installation and Administration Guide*.

The Tivoli Directory Integrator server needs two pieces of information for a mutually authenticated SSL configuration.

- A private key and certificate for the server
- The public certificate or trusted signer of the client

To enable mutually authenticated SSL support on the Tivoli Directory Integrator server, complete the steps in this procedure:

Procedure

1. Store the private key and certificate in a Java keystore file, such as `server_identity.jks`.

The certificate alias of the private key in that keystore is called `tdi_server`. In this example, the `.jks` files have been created with the IBM iKeyman utility, which requires a keystore password.

The utility, however, does not create a separate key password for the individual private key. This other key password is important when creating the Tivoli Directory Integrator server stash file. In this example, the keystore password is `passw0rd`, and the private key password does not exist.

2. Store the public certificate or trusted signer of the client in a Java keystore, such as `client_signer.jks`.

   Note: The certificate alias of the trusted signer certificate is not important for this configuration. The password used for the keystore is important, and this example uses `passw0rd`.
   The distinguished name (DN) of the client certificate is important. In this example, the DN of the client certificate is:

   ```
   CN=tdi_client, O=ibm, C=US
   ```

3. Modify `solution.properties` for Tivoli Directory Integrator server SSL support:

   Note: The properties vary depending on the version of the Tivoli Directory Integrator server. Do not add properties that are not in the `solution.properties` file of the Tivoli Directory Integrator server version you are working on.

   com.ibm.di.server.keystore *(for version 6.1.1) or* **api.keystore** *(for versions 7.0 and 7.1)*
   Specifies the keystore file which contains the private key and certificate of the Tivoli Directory Integrator server.

   The `server_identity.jks` file must be located in the solutions directory (`/root/TDI` for our example). Change the default of `testserver.jks` to `server_identity.jks`.

   com.ibm.di.server.key.alias *(for version 6.1.1) or* **api.key.alias** *(for versions 7.0 and 7.1)*
   Specifies the certificate alias in the server keystore file which represents the private key of the Tivoli Directory Integrator server.

   Change the *default of server* to `tdi_server`.

   {protect}-api.keystore.password
   Specifies the keystore password for the keystore in the `api.keystore` property.

   {protect}-api.key.password
   Specifies the password for the key alias in the `api.key.alias` property.

   com.ibm.di.server.encryption.keystore
   Specifies the data encryption for the keystore that hosts the key used by the Tivoli Directory Integrator server.

   com.ibm.di.server.encryption.key.alias
   Specifies the encryption keystore key alias.

   com.ibm.di.server.encryption.keystoretype
   Specifies the type of keystore that hosts the encryption key of the Tivoli Directory Integrator server.

com.ibm.di.server.encryption.transformation
Specifies the name of the cryptography transformation used for encryption. This value can be set to either *RSA* (public key encryption) or to some secret key transformation.

api.truststore
Specifies a keystore containing trusted signer certificates and CA for server API clients.

The `client_signer.jks` file must be located in the solutions directory (`/root/TDI` for our example). Change the default of `testserver.jks` to `client_signer.jks`.

{protect}-api.truststore.pass
Specifies the keystore password for the keystore in the `api.truststore` property.

Put the prefix `{protect}-`, to automatically encrypt it the next time the server is run. Change the default of `{encr}-key_string` to `passw0rd`.

api.remote.ssl.on
Set this property to `true` to enable SSL.

4. Create the Tivoli Directory Integrator server stash file.

   The Tivoli Directory Integrator server stash file is `idisrv.sth`, which is located in the solutions directory. This file can contain one or two passwords.

   The first password opens the keystore containing the server identity (`server_identity.jks`). The second (optional) password is for the key itself within that keystore. If only one password is specified, the new password is assumed to be the same as the first.

   When using the IBM iKeyman utility to create a self-signed certificate in a keystore file, the keystore password is manually specified when you create the keystore. However, there is *no* private key password for the private key.

   You must create the Tivoli Directory Integrator server stash file with a keystore password. Then, set the private key password to null (the empty string), as follows:

   ```
   # /opt/IBM/TDI/V6.1.1/bin/createstash.sh passw0rd ""
   ```

5. Update the Tivoli Directory Integrator server registry to recognize the DN of the client as an administrator.

   The Tivoli Directory Integrator server performs authorization on authenticated server API requests through a user registry and its assigned roles. The default registry is a text file, located in:

   `<`*`solutions_directory`*`>/serverapi/registry.txt`

   Add the following text to the `registry.txt` file:

   ```
   [USER]
   [ID]:CN=tdi_client, O=ibm, C=US
   [ROLE]:admin
   [ENDUSER]
   ```

   The text must be the same with the values for *issued to* and *issued by*. To check these values, select the label for `tdi_client.jks` in *iKeyman* and click **View/Edit**.

   For more advanced registry configuration, see the Tivoli Directory Integrator information center.

6. Start the Tivoli Directory Integrator server and validate that the startup message indicates SSL is in use, as follows:

```
# /opt/IBM/TDI/V6.1.1/ibmdisrv -d
CTGDKD024I Remote API successfully started on port:1099,
bound to:'SessionFactory'. SSL and Client Authentication
are enabled.
```

Results

The server-side SSL configuration is complete.

Configuring SSL for the Tivoli Directory Integrator client

You can configure the Tivoli Directory Integrator client-side to achieve a mutually authenticated SSL in many ways.

About this task

This topic contains the following information:

- One of the various ways of configuring SSL.
- One possible scenario when configuring it for the Tivoli Directory Integrator client.

Complete information can be found in the Tivoli Directory Integrator Information Center:

- For Tivoli Directory Integrator version 6.1.1

  Tivoli Directory Integrator version 6.1.1 information center. See the topic on *Secure Sockets Layer (SSL) Support* from the *Administration Guide*.
- For Tivoli Directory Integrator version 7.0

  Tivoli Directory Integrator version 7.0 information center. See the topic on *Secure Sockets Layer (SSL) Support* from the *Installation and Administration Guide*.
- For Tivoli Directory Integrator version 7.1

  Tivoli Directory Integrator version 7.1 information center. See the topic on *Secure Sockets Layer (SSL) Support* from the *Installation and Administration Guide*.

The Tivoli Directory Integrator Security Token Services module acts as an SSL client, and can operate in either one of the configurations:

- WebSphere Application Server JSSE

  Using the WebSphere Application Server JSSE configuration for SSL support.

  Note: You must use this option with embedded WebSphere.
- Java system properties

  Specifying Java system properties to control which keystore and truststore the Tivoli Directory Integrator server API uses.

  Note: This option does not work with embedded WebSphere.

For both options, the client needs two pieces of information for a mutually authenticated SSL configuration:

- A private key and certificate for the client identity.
- The public certificate or trusted signer of the server.

To configure the client-side SSL, complete the steps in this procedure:

Procedure

1. Store the private key and certificate in a Java keystore file, such as `client_identity.jks`. The certificate alias of the private key in that keystore is called `tdi_client`.

   Note: In this example, the `.jks` file is created with the IBM iKeyman utility, and the keystore password is `passw0rd`. iKeyman does not assign a second password assigned to the key. To successfully start the Java Virtual Machine, it is necessary to assign a password for the key. Use the same password as the keystore password.
2. Store the public certificate or trusted signer of the server in a Java keystore, such as `server_signer.jks`.

   Note: The certificate alias of the trusted signer certificate is not important for this configuration, but the keystore password keystore is needed. Set the password to `passw0rd`.
3. Use the Java keytool parameter to modify the keystore that was created with iKeyman and assign a password to the key, as follows:

   ```
   # /opt/IBM/WebSphere/AppServer/java/bin/keytool -keypasswd
   -alias tdi_client -new passw0rd -keystore client_identity.jks
   -storepass passw0rd
   ```
4. Use the Tivoli Federated Identity Manager key service to import both keystore files into the WebSphere configuration repository, for future reference.
   a. From the administration console, select **Configure Key Service** > **Keystores** menu to import `client_identity.jks` and `server_signer.jks`.
   b. On the user interface panel for the `client_identity.jks`, specify a keystore name of `tdi_client` and a type of signing or encryption keys.
   c. On the user interface panel for `server_signer.jks`, specify a keystore name of `tdi_server` and a type of CA certificates.

   The files in the WebSphere configuration repository file system are at:

   ```
   <config_root>/itfim/<fim_domain>/etc/jks/tdi_client.jks
   <config_root>/itfim/<fim_domain>/etc/jks/tdi_server.jks
   ```

   This configuration example uses a `fim_domain` value of `idp`.
5. Use either one of the following methods for client-side SSL configuration of the Tivoli Directory Integrator Security Token Services module:
   - "Configuring client-side SSL using WebSphere JSSE"
   - "Configuring client-side SSL using Java system properties" on page 161

   Note: You must complete only one of the two methods.

Configuring client-side SSL using WebSphere JSSE

Use the WebSphere JSSE to configure the client-side SSL.

About this task

This topic summarizes information described in detail in the following locations:

- WebSphere Application Server information center.
- developerWorks topic: *SSL, certificate, and key management enhancements for even stronger security in WebSphere Application Server version 6.1*:

  http://www-128.ibm.com/developerworks/websphere/techjournal/0612_birk/0612_birk.html?ca=drs-

The dynamic outbound endpoint SSL configuration cannot be used for the following reasons:

- It requires the SSL client to use the WebSphere `JSSEHelper` class to set specific connection information parameters.
- Tivoli Directory Integrator uses only standard Java JSSE interfaces.

Consequently, the scoped SSL configuration for the server, which is running the Tivoli Federated Identity Manager runtime, must be modified. Depending on whether you are running a cluster or a stand-alone application server, you can modify it at either cell or node level.

This example uses a stand-alone application server and modifies the node default keystore and node default truststore. The node default keystore is named as *NodeDefaultKeyStore*, and the node default truststore is named as *NodeDefaultTrustStore*.

Follow these tasks:

- Import the client private key and certificate to update the *NodeDefaultKeyStore*.
- Import the public certificate of the server to update the *NodeDefaultTrustStore*.

Procedure

1. Use the WebSphere administration console to import the client private key and certificate into the *NodeDefaultKeyStore*.
   a. Select **Security > SSL certificate and key management > Key stores and certificates > NodeDefaultKeyStore > Personal certificates**.
   b. Click **Import** to import a new key, and enter the following values:

      Key file name
      `/opt/IBM/WebSphere/AppServer/profiles/idp/config/itfim/idp/etc/jks/tdi_client.jks`

      Type
      `JKS`

      Key file password
      `passw0rd`

      Note: Click **Get key file aliases**.

      Certificate alias to import
      `tdi_client`

      Imported certificate alias
      `tdi_client`

   c. After the import, the key must show in the *Alias* column as `tdi_client`. Save the WebSphere configuration after loading the key.

   Before the public certificate of the server is imported into the *NodeDefaultTrustStore*, the server certificate must be in a simple file format rather than in the JKS. For example, PEM ASCII format or DER binary format. Use IBM *iKeyman* or *keytool* to export the public certificate of the server from the file:

   `<config_root>/itfim/<fim_domain>/etc/jks/tdi_server.jks`

   For example, you can export the public key using iKeyman into a PEM ASCII format in a file called:

   `/root/keys/tdi_server.arm`

2. Use the WebSphere administration console to import the server public certificate into the *NodeDefaultTrustStore*.
   a. Select **Security > SSL certificate and key management > Key stores and certificates > NodeDefaultTrustStore > Signer certificates**.
   b. Click **Add** to add a certificate.
      The Add Signer Certificate panel opens.
   c. Enter the following values:

      Alias
      `tdi_server`

      File name
      `/root/keys/tdi_server.arm`

      Data type
      `Base-64 encoded ASCII data`
   d. The certificate named `tdi_server` must now be in the list of certificates. Save the WebSphere configuration after committing this change.
3. Click **Retrieve from Port**.
4. Enter the general properties section which consists of **Host**, **Port**, and **SSL configuration for outbound connection**.
5. Restart WebSphere Application Server instance.

Results

The client is configured for SSL.

Configuring client-side SSL using Java system properties

Use the Java system properties to select keystores and certificates for SSL communications.

About this task

The Java system properties for client-side SSL are described in the following locations:

- For Tivoli Directory Integrator version 6.1.1
  Tivoli Directory Integrator version 6.1.1 information center. See the topic on *Remote Server API* from the *Administration Guide*.
- For Tivoli Directory Integrator version 7.0
  Tivoli Directory Integrator version 7.0 information center. See the topic on *Server API Access Security* from the *Installation and Administration Guide*.
- For Tivoli Directory Integrator version 7.1
  Tivoli Directory Integrator version 7.1 information center. See the topic on *Server API Access Security* from the *Installation and Administration Guide*.

Note: Configuring the client-side using Java system properties is not available for embedded WebSphere installations.

These Java system properties can be used select keystores and certificates for SSL communications:

`api.client.ssl.custom.properties.on`
Instructs the Tivoli Directory Integrator server API to use custom properties for keystore and truststore configuration rather than the JSSE configuration. For example: `true`.

api.client.keystore
Specifies the keystore containing the client certificate. For example:
`${USER_INSTALL_ROOT}/config/itfim/idp/etc/jks/tdi_client.jks`

api.client.keystore.pass
Specifies the password for the file specified in `api.client.keystore`. For example, `passw0rd`.

api.client.key.pass
Specifies the password for the actual key in `api.client.keystore`.

Leave unspecified as the *keytool* utility is used to make the key password the same as the keystore password.

api.truststore
Specifies the keystore containing the Tivoli Directory Integrator server public certificate. For example:
`${USER_INSTALL_ROOT}/config/itfim/idp/etc/jks/tdi_server.jks`

api.truststore.pass
Specifies the password for the file specified in `api.truststore`. For example, `passw0rd`.

Use the WebSphere administration console to update the servers Java Virtual Machine startup parameters.

Procedure

1. Select **Servers** > **Application Servers** > **server1** > **Java and Process Management** > **Process Definition** > **Java Virtual Machine**.
2. Update the properties:

   Generic Java Virtual Machine arguments:
   -api.client.ssl.custom.properties
3. Restart WebSphere Application Server instance.

Results

The client is configured for SSL.

Creating a custom mapping module

Creating a custom mapping module is a programming-intensive procedure that involves writing a Java class and installing the class into the plug-ins directory for your domain.

Before you begin

To create a custom mapping module, you must be familiar with the structure of Tivoli Federated Identity Manager trust service modules and the proper procedures for creating them and adding them to your environment.

About this task

Learn more about trust service modules in:

- *Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions* (SG24-6394-01). This book is available in PDF (Portable Document

Format) at http://www.redbooks.ibm.com/redbooks/pdfs/sg246394.pdf or in HTML (Hypertext Markup Language) at http://www.redbooks.ibm.com/redbooks/SG246394/

- A developerWorks article titled *Tivoli Federated Identity Manager: Implementing and deploying custom trust modules* at http://www-128.ibm.com/developerworks/tivoli/library/t-sts-custom/

Adding a custom mapping module

To add a custom mapping module that you have created, you must first define the module as a new module type in the Tivoli Federated Identity Manager environment.

Before you begin

You must write a Java class for a new module type and install the class into the plug-ins directory for your domain. You can then use the following instructions to create a new module type entry in the console.

About this task

This task is necessary only when the XSL Transformation module that is supplied with Tivoli Federated Identity Manager does not meet the requirements of your deployment.

Procedure

1. Click **Tivoli Federated Identity Manager** > **Manage Configuration** > **Runtime Node Management**. The Runtime Node Management panel opens.
2. Click the **Publish plug-ins** button.
3. When prompted, click the **Load configuration changes to Tivoli Federated Identity Manager runtime**. The new module type shows in the Module Type list.

What to do next

Continue with the task for adding an instance of the mapping file in "Adding an instance of a custom mapping module."

Adding an instance of a custom mapping module

After you have created your mapping module and added it as a module type, you must create an instance of that module type to use it in the Tivoli Federated Identity Manager environment.

Before you begin

Be sure that you have completed the following tasks before continuing with these instructions:

- "Creating a custom mapping module" on page 162
- "Adding a custom mapping module"

About this task

The console provides a wizard to guide you through adding the module instance.

Procedure

1. Select **Tivoli Federated Identity Manager** > **Configure Trust Service** > **Module Instances**. The Module Instances panel shows 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.
6. Click **Finish**. See the online help for descriptions of the fields.

What to do next

The new mapping file is available in the list of modules that you can choose from when establishing a federation.

Chapter 15. SAML federations overview

SAML (Security Assertion Markup Language) is an XML standard for exchanging single sign-on information. It relies on the use of SOAP among other technologies to exchange XML messages over computer networks. The XML messages are exchanged through a series of requests and responses. In this process, one of the federation partners sends a request message to the other federation partner. Then, that receiving partner immediately sends a response message to the partner who sent the request.

Tivoli Federated Identity Manager supports the following OASIS Security specifications for exchanging information in a federation:

- SAML 1.0 and 1.1 (1.x)
- SAML 2.0

The SAML specifications include descriptors to establish a federation, initialize, and manage single sign-on. The following descriptors specify the structure, content of the messages, and the way the messages are communicated between partners and users.

Assertions
: XML-formatted tokens that are used to transfer user identity information, such as the authentication, attribute, and entitlement information, in the messages.

Protocols
: The types of request messages and response messages that are used for obtaining authentication data and for managing identities.

Bindings
: The communication method used to transport the messages.

Profiles
: Combinations of protocols, assertions, and bindings that are used together to create a federation and enable federated single sign-on.

When using Tivoli Federated Identity Manager, you and your partner must do the following tasks:

- Use the same SAML specification (1.0, 1.1, or 2.0).
- Agree on which protocols, bindings, and profiles to use.

The next topics provide brief descriptions of how SAML 1.x and SAML 2.0 specifications are used in Tivoli Federated Identity Manager. However, these descriptions do not provide all of the details of the specifications. See the OASIS specification documents at http://www.oasis-open.org/specs/index.php for more details.

SAML 1.x

Tivoli Federated Identity Manager supports both SAML 1.0 and SAML 1.1. These specifications are referred to collectively as SAML 1.x.

If you and your partner choose to use SAML 1.x in your federation, you need to understand the SAML 1.x support that is provided in Tivoli Federated Identity Manager.

Assertions

The assertions created by Tivoli Federation Identity Manager contain authentication statements, which assert that the principal (that is, the entity requesting access) was authenticated. Assertions can also carry attributes about the user that the identity provider wants to make available to the service provider.

Assertions are usually passed from the identity provider to the service provider.

The following variables control the content of the assertions created by Tivoli Federated Identity Manager:

- The specification (SAML 1.0 or 1.1) that you select when you establish a federation.
- The definitions used in the TFIM identity mapping method that you configure.

Identity mapping specifies how identities are mapped between federation partners.

The Tivoli Federated Identity Manager identity mapping method can either be a custom mapping module or an XSL transformation file.

Protocol

In Tivoli Federated Identity Manager, SAML 1.x uses a simple request-response protocol to make authentication requests.

Binding

SAML 1.x both plain HTTP (using browser redirects) or SOAP for the transportation of messages. The *profile* used in the federation further specifies how the communication of the messages takes place.

Profiles

SAML 1.x specifies two options for profiles:

Browser artifact
: Browser artifact uses SOAP-based communications (also called the SOAP backchannel) to exchange an artifact during the establishment and use of a trusted session between an identity provider, a service provider, and a client (browser).

Browser POST
: Browser POST uses a self-posting form during the establishment and use of the trusted session between an identity provider, a service provider, and a client (browser).

Tivoli Federated Identity Manager supports browser artifact by default when you select SAML 1.0 or SAML 1.1 as the profile for your federation. However, you can use browser POST in your federation on a per-partner basis. For example, if you are a service provider, you can specify that your identity provider partner uses Browser POST when you configure that partner. If you are an identity provider, you can enable the IBM PROTOCOL extension when configuring a SAML 1.x federation.

The URL that is used to initiate single sign-on differs depending on whether the identity provider is using this extension. For more information about URLs, see "SAML 1.x initial URL" on page 767.

SAML 2.0

The SAML 2.0 specification introduced more flexibility than the previous SAML 1.x specifications.

Assertions

The assertions created by Tivoli Federated Identity Manager contain authentication statements. These authentication statements assert that the principal (that is, the entity requesting access) was authenticated. Assertions can also carry attributes about the user that the identity provider wants to make available to the service provider.

Assertions are typically passed from the identity provider to the service provider.

The content of the assertions that are created is controlled by the specification (SAML 2.0). Select these assertions when you establish a federation. You can also select these assertions by the definitions used in the Tivoli Federated Identity Manager identity mapping method that you configure.

The identity mapping method can either be a custom mapping module or an XSL transform file. The identity mapping also specifies how identities are mapped between federation partners.

Protocols

SAML 2.0 defines several request-response protocols, all correspond to the action being communicated in the message. The SAML 2.0 protocols that are supported in Tivoli Federated Identity Manager are:

- Authentication request
- Single logout
- Artifact resolution
- Name identifier management

Bindings

When you use SAML 2.0 in Tivoli Federated Identity Manager, you have several binding options. These options specify the way in which messages can be transported:

HTTP redirect

HTTP redirect enables SAML protocol messages to be transmitted within URL parameters. It enables SAML requestors and responders to communicate using an HTTP user agent as an intermediary.

The intermediary might be necessary if the communicating entities do not have a direct path of communication. The intermediary might also be necessary if the responder requires interaction with a user agent, such as an authentication agent.

HTTP redirect is sometimes called browser redirect in single sign-on operations. This profile is selected by default.

HTTP POST

HTTP POST enables SAML protocol messages to be transmitted within an HTML form using base64-encoded content. It enables SAML requestors and responders to communicate using an HTTP user agent as an intermediary.

The agent might be necessary if the communicating entities do not have a direct path of communication. The intermediary might also be necessary if the responder requires interaction with a user agent such as an authentication agent.

HTTP POST is sometimes called Browser POST, particularly when used in single sign-on operations. It uses a self-posting form during the establishment and use of a trusted session between an identity provider, a service provider, and a client (browser).

HTTP artifact

HTTP artifact is a binding in which a SAML request or response (or both) is transmitted by reference using a unique identifier called an artifact.

A separate binding, such as a SOAP binding, is used to exchange the artifact for the actual protocol message. It enables SAML requestors and responders to communicate using an HTTP user agent as an intermediary.

This setting is used when it is not preferable to expose the message content to the intermediary.

HTTP artifact is sometimes called browser artifact, particularly when used in single sign-on operations. The HTTP artifact uses a SOAP back channel. The SOAP back channel is used to exchange an artifact during the establishment and use of a trusted session between an identity provider, a service provider, and a client (browser).

SOAP

SOAP is a binding that uses Simple Object Access Protocol (SOAP) for communication.

The choice of binding you have depends on the profile you choose to use in your federation.

Profiles

Tivoli Federated Identity Manager supports the configuration of the single sign-on profile on a per-partner basis. The profiles supported are:

Web browser single sign-on

The Web Browser SSO profile is the consolidation of the browser artifact and browser POST profiles that were introduced in SAML 1.x.

Using this profile, an authentication request message is sent from a service provider to an identity provider. A response message containing a SAML assertion is sent from the identity provider to the service provider. Additional messages are sent related to artifact resolution, if that binding is used.

This profile provides options regarding the initiation of the message flow and the transport of the messages:

Message initiation

The message flow can be initiated from the identity provider or the service provider.

When the identity provider initiates the message flow, a **RelayState** parameter can be provided in the unsolicited response delivered by the identity provider to the service provider. This parameter contains the URL-encoded value of the Target element provided in the single sign-on service initial URL (identity provider).

Bindings

In a Tivoli Federated Identity Manager environment, the following bindings can be used in the Web browser SSO profile:

- HTTP Redirect (available only in an identity provider configuration)
- HTTP POST
- HTTP artifact

The choice of binding depends on the type of messages being sent. For example, an authentication request message can be sent from a service provider to an identity provider. The response message can be sent from an identity provider to a service provider using either HTTP POST or HTTP artifact. A pair of partners in a federation do not need to use the same binding.

Options

The Web Browser single sign-on profile in Tivoli Federated Identity Manager also provides the following option:

Enhanced Client Proxy This profile option enables an enhanced client or proxy (ECP) to communicate with an identity provider and service provider on behalf of a user (client).

For example, a user might request a resource from a service provider. The service provider might not know which identity provider to access to authenticate the user.

Using the ECP profile option, the service provider can contact the ECP, which knows how to locate and access the appropriate identity provider. The ECP profile supports SOAP and reverse SOAP (PAOS) bindings during the processing of authentication requests.

Single Logout

The Single Logout profile is used to terminate all the login sessions currently active for a specified user within the federation. A user who achieves single sign-on to a federation establishes sessions with more than one participant in the federation.

The sessions are managed by a session authority, which in many cases is an identity provider. When the user wants to end sessions with all session participants, the session authority can use the single logout profile to globally terminate all active sessions.

Message initiation

The message flow can be initiated from the identity provider or the service provider.

Bindings

In a Tivoli Federated Identity Manager environment, the following bindings can be used in the Single Logout profile:

- HTTP Redirect
- HTTP POST

- HTTP artifact
- SOAP

Name Identifier Management

The Name Identifier Management profile manages user identities that are exchanged between identity providers and service providers.

The profile enables identity providers to notify service providers. Service providers are notified when there is a change to the content or format of an identity for a given user (principal).

The profile enables service providers to specify unique *aliases* for the principal. Service providers can also send those aliases to the identity provider to be used instead of the principal name.

The profile also enables either provider. The profile notifies its partner when it decides to no longer issue or accept messages that use the identity of the principal.

To manage the aliases, Tivoli Federated Identity Manager uses a function called the *alias service*. The alias service stores and retrieves aliases that are related to a federated identity. Aliases can be used in the following ways:

Persistent aliases

When persistent aliases are used, the identity of the user is federated by the identity provider to the identity of the user at the service provider. A persistent SAML name identifier is used. The user remains in the federation permanently, that is, until a request is made to terminate the federation.

Transient aliases

When transient aliases are used, a temporary identifier is used to federate between the identity provider and service provider. A temporary identifier is used only for the life of the single sign-on session of the user.

In a Tivoli Federated Identity Manager environment, aliases are stored in and retrieved from one of the following types of repositories:

- An LDAP database.
- A relational database that supports JDBC.

During the configuration of Tivoli Federated Identity Manager, you can configure your environment to use one of these repository types.

Message initiation

The message flow can be initiated from the identity provider or the service provider.

Bindings

The following bindings can be used in the Name Identifier Management profile:

- HTTP Redirect
- HTTP POST
- HTTP artifact
- SOAP

Identity Provider Discovery

The Identity Provider Discovery profile is used by service providers to discover which identity provider is used by a user (principal) during Web browser single sign-on.

Some deployments have more than one identity provider, and the service provider must be able to determine which identity provider a principal uses.

The Identity Provider Discovery profile uses a cookie. The cookie is created in a domain that is common between identity providers and service providers in a given deployment. The cookie contains the list of identity providers and is called the *common domain cookie*.

When you configure your federation using the Tivoli Federated Identity Manager console, your profile options are:

Basic: Web Browser SSO, Single Logout
This setting enables the following profiles and bindings:

- Web Browser single sign-on with HTTP POST and HTTP Artifact bindings.
- Single logout, with HTTP POST and HTTP Artifact bindings.

Typical: Web Browser SSO, Single Logout, and Name Identifier
This setting enables the following profiles and bindings:

- Web Browser single sign-on, with HTTP POST and HTTP Artifact bindings.
- Single logout, with HTTP POST and HTTP Artifact bindings.
- Enhanced client or proxy
- Name Identifier Management, with HTTP POST and HTTP Artifact bindings.

Enable all profiles and bindings
This setting enables all the available profiles and bindings:

- Web Browser single sign-on, with HTTP POST, HTTP Artifact, and HTTP Redirect bindings.

  Note: HTTP Redirect is available only in an identity provider configuration.
- Enhanced client or proxy.
- Single logout, with HTTP Redirect, HTTP POST, and HTTP Artifact bindings.
- Name Identifier Management, with HTTP Redirect, HTTP POST, HTTP Artifact, and SOAP
- Identity Provider Discovery

Manual: Choose individual profiles and bindings
All supported profiles and available bindings are presented so that you can choose the ones you want to use.

Account linkage

In SAML 2.0, account linkage enables a user to link an identity provider account to a service provider. The link happens during the single sign-on initiated at the identity provider and service provider. In both scenarios, account linkage requires a user to be authenticated at both the service provider and identity provider.

An administrator can enable this feature in the partner settings panel. If this feature is enabled, the user must authenticate in the service provider when a persistent alias is received. The alias can not have been previously linked to an account in the service provider for the authentication to occur.

After the user authenticates, the SAML 2.0 implementation stores the alias at the service provider alias service and establishes account linkage.

Handling an unknown alias

SAML 2.0 supports aliases to communicate user identities between partners.

An administrator can configure the SAML 2.0 partner settings to handle an unknown alias in one of the following ways:

- The authentication page shows an error page when the service provider does not know the alias received from the identity provider. This setting is the default when you
  - Do not select **Force authentication to achieve account linkage**.
  - Do not select **Map unknown name identifiers to the anonymous username**.
- The SAML 2.0 implementation maps the identity of the user to the default user account. A guest account establishes the single sign-on session. This setting requires that you
  - Do not select **Force authentication to achieve account linkage**.
  - Select **Map unknown name identifiers to the anonymous username**.
- The user must authenticate at the service provider, which enables account linkage. This setting requires that you
  - Select **Force authentication to achieve account linkage**.
  - Do not select **Map unknown name identifiers to the anonymous username**.

Chapter 16. SAML endpoints and URLs

Communications within a federation take place through endpoints on the servers of the identity provider and service provider partners.

In a Tivoli Federated Identity Manager environment, endpoints fall into two categories:

- Endpoints that are specified by the federation specification (such as SAML 1.x or SAML 2.0) and are used for partner-to-partner communication.
- Endpoints that end users can access to initiate a single sign-on activity.

All endpoints can be accessed through URLs. The syntax of the URLs is specific to the purpose of the access and whether the access is by a partner or by an end user.

URLs for partner communication

The URLs that are used for partner-to-partner communication, such as the exchange of requests, in both SAML 1.x and SAML 2.0 federations are referred to collectively as *endpoint URLs* or individually by the name of the protocol and binding or service that they are related to. Administrators who are responsible for installing, configuring, and maintaining the Tivoli Federated Identity Manager environment and the partner-to-partner communication in that environment will see references to these endpoint URLs and might find it helpful to understand their purpose. See "SAML 1.x endpoints and URLs" on page 174 or "SAML 2.0 endpoints and URLs" on page 177.

URLs for user access

While the SAML specifications define the endpoints for partner-to-partner communication, they provide limited or no guidance about the endpoints or methods that end users must use to initiate single sign-on actions. Tivoli Federated Identity Manager supports specific URLs for end-user initiation of single sign-on actions.

In a SAML 1.x federation, the single sign-on process is always initiated at the *intersite transfer service*. The method by which the request arrives at this endpoint is not specified in the SAML specification. The syntax for the intersite transfer service URL in a Tivoli Federated Identity Manager environment is described in "SAML 1.x initial URL" on page 767.

In a SAML 2.0 federation, single sign-on actions can be initiated at the identity provider site or the service provider site. URLs that can be used by users to initiate a sign-on action are specific to the a single sign-on action (such as initiate a federated sign on, perform a single logout, or end account linkage) and to whether the action is being initiated at the identity provider or service provider site. In a Tivoli Federated Identity Manager environment, the URLs that can be used for initiating sign-on actions are referred to as *profile initial URLs*. Architects and application developers, who design and implement the interactions of their users with the single sign-on process, need to understand profile initial URLs. See "SAML 2.0 profile initial URLs" on page 769.

SAML 1.x endpoints and URLs

Several endpoints are configured on your point of contact server so that you and your partner can communicate. These endpoints are configured when you configure your federation in Tivoli Federated Identity Manager. The endpoints are accessible through URLs, and are used by the partners in the federation.

If you are responsible for installing, configuring, or maintaining a federation in Tivoli Federated Identity Manager, you might find it helpful to be familiar with the SAML 1.x endpoints and URLs.

The following endpoints are used in a SAML 1.x federation.

Point of contact server

The endpoint on the point of contact server where communication takes place. The syntax of the point of contact server URL is:

`https://`*`hostname:port_number`*

Where:

https https might be http if SSL is not enabled on the server.

hostname

The host name of the point of contact server.

port_number

The port number where communications take place on the server. The default port number on a WebSphere Application Server is 9443, if SSL is enabled, or 9080 if SSL is not enabled.

You are prompted for your point of contact server URL when you configure your federation. After the configuration, your point of contact server URL has `/sps` appended to it so that the syntax of the configured point of contact server URL is

`https://`*`hostname:port_number`*`/sps`

The `/sps` indicates that the URL is defined for single sign-on services.

Intersite transfer service

The endpoint on the identity provider point of contact server where the sign-on request process begins. This endpoint is where the single sign-on requests are sent. SAML does not specify how the requests arrive at this endpoint.

If you are an identity provider using Tivoli Federated Identity Manager, the method used depends on how and where the users are logging in. For example, if the users log in at the service provider partner website, your service provider partner needs the URL for your inter-site transfer service.

Your service provider partner also must configure some type of redirect that takes the users from their site to your login form.

The URL is based on the URL that you specify for your point of contact server. The syntax is:

`https://`*`hostname:port_number`*`/sps/`*`federation_name`*`/saml`*`xx`*`/login`

Where:

https

https might be http if SSL is not enabled on the server.

hostname
The host name of the point of contact server.

port_number
The port number where communications take place on the server.

sps

The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml*xx*
The version of SAML that is configured for the federation. The values can be:

- saml (for SAML 1.0)
- saml11 (for SAML 1.1)

login

The designation of what type of endpoint is using the port. **login** is used for the intersite transfer service in SAML 1.x federations.

This endpoint is used only on identity provider configurations and is defined automatically for you when you configure your federation.

Artifact resolution service
The endpoint on the identity provider point of contact server where artifacts are exchanged for assertions. This endpoint is the location where the federation partners communicate. It is sometimes referred to as the *SOAP endpoint* on the identity provider point of contact server.

Note: You might also be familiar with this endpoint as the *responder service*.

The URL is based on the URL that you specify for your point of contact server. The syntax is:

`https://hostname:port_number/sps/federation_name/samlxx/soap`

Where:

https

https might be http if SSL is not enabled on the server.

hostname
The host name of the point of contact server.

port_number
The port number where communications take place on the server. The default port number is 9444.

sps

The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml*xx*
The version of SAML that is configured for the federation. The values can be:

- saml (for SAML 1.0)
- saml11 (for SAML 1.1)

soap

The designation of what type of endpoint is using the port. **soap** is used for the artifact resolution service in SAML 1.x federations.

This endpoint is used only on identity provider configurations and is defined automatically for you when you configure your federation.

Assertion consumer service

The endpoint on the service provider point of contact server that receives assertions or artifacts. This endpoint is the location where the federation partners communicate. This endpoint is sometimes referred to as the *SOAP endpoint* on the service provider point of contact server.

Note: If you are using the browser artifact profile, you might be familiar with this endpoint as the *artifact consumer service* or the *artifact receiver service*.

The URL is based on the URL that you specify for your point of contact server. The syntax is:

`https://`*`hostname:port_number`*`/sps/`*`federation_name`*`/saml`*`xx`*`/login`

Where:

https

https might be http if SSL is not enabled on the server.

hostname

The host name of the point of contact server.

port_number

The port number where communications take place on the server. The default port number on a WebSphere Application Server is 9443.

sps

The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name

The name you give to the federation when you configure it.

saml*xx*

The version of SAML that is configured for the federation. The values can be:

- saml (for SAML 1.0)
- saml11 (for SAML 1.1)

login

The designation of what type of endpoint is using the port. **login** is used for the assertion consumer service.

This endpoint is used only on service provider configurations in SAML 1.x federations and is defined automatically for you when you configure your federation.

SAML 2.0 endpoints and URLs

Several endpoints are configured on your point of contact server so that communications can be exchanged between you and your partner. These endpoints are configured when you configure your federation in Tivoli Federated Identity Manager. The endpoints are accessible through URLs and are used by the partners in the federation.

If you are responsible for installing, configuring, or maintaining a federation in Tivoli Federated Identity Manager, you might find it helpful to be familiar with these endpoints and URLs.

The following endpoints are used in a SAML 2.0 federation.

Point of contact server

The endpoint on the point of contact server where communication takes place. The point of contact server URL is also used as the provider ID. The syntax of the point of contact server URL is:

`https://hostname:port_number`

Where:

https

https might be http if SSL is not enabled on the server.

hostname

The host name of the point of contact server.

port_number

The port number where communications take place on the server. The port number where communications take place on the server. The default port number on a WebSphere Application Server is 9443, if SSL is enabled, or 9080 if SSL is not enabled.

You are prompted for your point of contact server URL when you configure your federation. After the configuration, your point of contact server URL has `/sps` appended to it so that the syntax of the configured point of contact server URL is

`https://hostname:port_number/sps`

The `/sps` indicates that URL is defined for single sign-on services.

Artifact resolution service (or SOAP endpoint)

The endpoint on either the identity provider or service provider point of contact server where artifacts are exchanged for SAML messages. This endpoint is the location where the federation partners communicate. It is sometimes called as the *SOAP endpoint*.

Note: You might also be familiar with this endpoint as the *responder service*.

The URL is based on the URL that you specify for the point of contact server. The syntax is:

`https://hostname:port_number/sps/federation_name/saml20/soap`

Where:

https

https might be http if SSL is not enabled on the server.

hostname
The host name of the point of contact server.

port_number
The port number where communications take place on the server. The default port number is 9444.

sps

The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml20
The designation of the SAML protocol you choose to use in your federation.

soap

The designation of what type of endpoint is using the port. **soap** is used for the artifact resolution service in SAML 2.0 federations.

This endpoint is defined automatically for you when you configure your federation.

Assertion consumer service
The endpoint on the service provider point of contact server that receives assertions or artifacts. This endpoint is the location where the federation partners communicate.

The URL is based on the URL that you specify for the point of contact server. The syntax is:

`https://`*`hostname:port_number`*`/sps/`*`federation_name`*`/saml20/login`

Where:

https

https might be http if SSL is not enabled on the server.

hostname
The host name of the point of contact server.

port_number
The port number where communications take place on the server. The default port number on a WebSphere Application Server is 9443, if SSL is enabled, or 9080 if SSL is not enabled.

sps

The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml20
The designation of the SAML protocol you choose to use in your federation.

login

The designation of what type of endpoint is using the port. **login** is used for the assertion consumer service in SAML 2.0 federations.

This endpoint is used only on service provider configurations in SAML 2.0 federations and is defined automatically for you when you configure your federation.

Single sign-on service
The endpoint on the identity provider point of contact server that receives authentication requests.

The URL is based on the URL that you specify for the point of contact server. The syntax is:

`https://hostname:port_number/sps/federation_name/saml20/login`

Where:

https

https might be http if SSL is not enabled on the server.

hostname
The host name of the point of contact server.

port_number
The port number where communications take place on the server. The default port number on a WebSphere Application Server is 9443, if SSL is enabled, or 9080 if SSL is not enabled.

sps

The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml20
The designation of the SAML protocol you choose to use in your federation.

login

The designation of what type of endpoint is using the port. **login** is used for the assertion consumer service in SAML 2.0 federations.

This endpoint is used only on identity provider configurations in SAML 2.0 federations and is defined automatically for you when you configure your federation.

Single logout service
The endpoint on the identity provider or service provider point of contact server that receives logout requests.

The URL is based on the URL that you specify for the point of contact server. The syntax is:

`https://hostname:port_number/sps/federation_name/saml20/slo`

Where:

https

https might be http if SSL is not enabled on the server.

hostname
The host name of the point of contact server.

port_number
The port number where communications take place on the server. The default port number on a WebSphere Application Server is 9443, if SSL is enabled, or 9080 if SSL is not enabled.

The port is assigned with the default value unless the port is unavailable when Tivoli Federated Identity Manager is installed. If the default port is unavailable, the installation program adds a value of 1 to the port number until it finds an available port of that number.

sps

The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml20
The designation of the SAML protocol you choose to use in your federation.

slo

The designation of what type of endpoint is using the port. **slo** is used for the single logout service in SAML 2.0 federations.

Name identifier management service
The endpoint on the identity provider or service provider point of contact server that receives messages related to name management. The URL is based on the URL that you specify for the point of contact server and on the binding that is used.

The syntax for HTTP Redirect, HTTP POST, and HTTP Artifact is:

`https://hostname:port_number/sps/federation_name/saml20/mnids`

The syntax for SOAP is:

`https://hostname:port_number/sps/federation_name/saml20/soap`

Where:

https

https might be http if SSL is not enabled on the server.

hostname
The host name of the point of contact server.

port_number
The port number where communications take place on the server. The port depends on the binding being used. The default ports are:

HTTP SOAP: 9444

HTTP POST, HTTP Artifact, HTTP Redirect: 9443

sps

The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml20
The designation of the SAML protocol you choose to use in your federation.

mnids or soap
The designation of what type of endpoint is using the port. **mnids** is used for the name identifier management service in SAML 2.0 federations that use HTTP Redirect, HTTP POST, or HTTP Artifact. **soap** is used when SOAP is used as the binding.

Common Domain Cookie Service URL used by the Identity Provider Discovery service

By default, Tivoli Federated Identity Manager provides a common domain service implementation that makes it possible for an identity provider to inform a service provider that a specific user is ready to use a federation.

The default URL is used internally and specifies if the common domain cookie service is going to read or write (get or set) the values using cdcwriter (the identity provider) or cdcreader (the service provider) appended to the end of the URL. The default syntax for the URL is:

`https://common_domain_name/sps/federation_name/saml20/[cdcreader|cdcwriter}`

Where:

https
https might be http if SSL is not enabled on the server.

common_domain_name
The shared common domain name.

sps
The context root for the single sign-on application on WebSphere Application Server. This part of the URL cannot be changed.

federation_name
The name you give to the federation when you configure it.

saml20
The designation of the SAML protocol you choose to use in your federation.

cdcwriter *or* **cdcreader**
The designation of what type of action (read/get or write/set) is used.

Note: Tivoli Federated Identity Manager also supports the use of a third-party or custom discovery service.

Chapter 17. Sample identity mapping rules for SAML federations

The following topics show the sample identity mapping rules that are provided for SAML federations. If you have decided to use identity mapping rules for your federation, you can review the XSLT rules.

For an overview of identity mapping, including discussion of identity mapping options that do not use XSLT mapping rules files, see Chapter 14, "Planning the mapping of user identities," on page 143

- "Mapping a local user identity to a SAML 1.x token"
- "Mapping a SAML 1.x token to a local user identity" on page 184
- "Mapping a local identity to a SAML 2.0 token using an alias" on page 185
- "Mapping a SAML 2.0 token to a local identity" on page 186

Mapping a local user identity to a SAML 1.x token

This scenario occurs when messages are exchanged between partners in a SAML 1.0 or SAML 1.1 single sign-on federation.

When a user request is received (for example, for access to a remote resource) Tivoli Federated Identity Manager contacts the point of contact server (for example, WebSphere Application Server) and obtains a local user identity.

The Tivoli Federated Identity Manager server places the local user identity information into an XML document that conforms to the security token service universal user (STSUUSER) schema. The server then consults its configuration entry for the federation partner (for example, the destination that hosts a requested resource). The configuration indicates the type of token to be created. In this case, the token type is SAML.

The identity mapping module then modifies the XML document to contain the information required to build a SAML token.

Table 20. STSUUSER entries used to generate a SAML token

STSUUSER element	SAML Token Information	Required?
Principal Attr: Name	AuthenticationStatement/Subject/NameIdentifier	Required
Attribute List	Additional custom attributes	Optional

The mapping module is responsible for two tasks:

1. Mapping Principal Attr Name to a Principal Name entry.

   The type must be valid for SAML. For example:

   ```
   urn:oasis:names:tc:SAML:1.0:assertion#emailAddress
   ```

   Figure 10 on page 184 shows part of the default mapping rule file, ip_saml_1x.xsl.

```
<!--
  This template replaces the entire Principal element with one that contains
  just the iv user name.
 -->
 <xsl:template match="//stsuuser:Principal">
  <stsuuser:Principal>
   <stsuuser:Attribute name="name" type="urn:oasis:names:tc:SAML:1.0:assertion#emailAddress">
    <stsuuser:Value>
     <xsl:value-of select="//stsuuser:Principal/stsuuser:Attribute[@name='name']
     [@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
    </stsuuser:Value>
   </stsuuser:Attribute>
  </stsuuser:Principal>
 </xsl:template>
```

Figure 10. XSL code sample showing mapping of a local user identity into a Principal name for a SAML token

In this example, the local user identity is referred to as the *iv user name*.

```
<stsuuser:Value>
     <xsl:value-of select="//stsuuser:Principal/stsuuser:Attribute[@name='name']
     [@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
```

2. Setting the authentication method to the `password` mechanism. This action is required by the SAML standard.

   See Figure 11.

```
<xsl:template match="//stsuuser:AttributeList">
     <stsuuser:AttributeList>
         <!-- First the authentcation method attribute -->
         <stsuuser:Attribute name="AuthenticationMethod"
              type="urn:oasis:names:tc:SAML:1.0:assertion">
            <stsuuser:Value>urn:oasis:names:tc:SAML:1.0:am:password</stsuuser:Value>
         </stsuuser:Attribute>
     </stsuuser:AttributeList>
</xsl:template>
```

Figure 11. XSL code sample showing assignment of authentication method as an Attribute for a SAML token

Mapping a SAML 1.x token to a local user identity

The service provider receives a SAML 1.0 or SAML 1.1 token. Tivoli Federated Identity Manager converts the token contents into a XML file that conforms to the security token service universal user schema.

Table 21. SAML token information that is converted into a STS universal user document

SAML Token Information	STSUUSER element
AuthenticationStatement/Subject/NameIdentifier	Principal Attr: Name

Tivoli Federated Identity Manager converts this information to a local user identity.

- The NameIdentifier is used to populate the `name` attribute of the Principal.

  Figure 12 on page 185 shows the assignment of a set value for the Principal name. This code sample is from the default mapping file sp_saml_1x.xsl

```
<!--
  This will replace the principal name with the user's local name.
 -->
 <xsl:template match="//stsuuser:Principal/stsuuser:Attribute[@name='name']">
  <stsuuser:Attribute name="name" type="urn:ibm:names:ITFIM:5.1:accessmanager">
            <stsuuser:Value><xsl:value-of
select="//stsuuser:Principal/stsuuser:Attribute[@name='name']/stsuuser:Value"/>
           </stsuuser:Value>
            </stsuuser:Attribute>
 </xsl:template>
```

Figure 12. XSL code sample showing assignment of a value for the Principal name for a SAML token.

Another sample mapping file that maps a SAML 1.x token to a local identity is sp_saml_1x_ext.xsl. This file performs the mapping as described, but adds a section that verifies if the identity provider has used an acceptable level of authentication. In this sample file, an exception is thrown if the identity provider has used password authentication.

```
<xsl:param name="message">Detected an unacceptable authentication method.
  A higher level of authentication is required.</xsl:param>
 <xsl:template match="//stsuuser:AttributeList">
 <xsl:variable name="result" select="//stsuuser:AttributeList/
  stsuuser:Attribute[@name='AuthenticationMethod']/stsuuser:Value"/>
 <xsl:if test="(contains($result,'password')) = 'true'">
  <xsl:value-of select="mapping-ext:throwSTSException($message)" />
 </xsl:if>
 </xsl:template>
```

Figure 13. XSL code sample showing verification of a value for the AuthenticationMethod

Mapping a local identity to a SAML 2.0 token using an alias

This scenario occurs when messages are exchanged between partners in a SAML 2.0 single sign-on federation.

When a user request is received, Tivoli Federated Identity Manager contacts the point of contact server, and obtains a local user identity. For example, a request can be made to access a remote resource, and the point of contact server can be a WebSphere Application Server. The scenario described here uses the ip_saml_20.xsl sample mapping file in which an alias is used for the identity.

The Tivoli Federated Identity Manager server places the local user identity information into an XML document that conforms to the security token service universal user (STSUUSER) schema. The server then consults its configuration entry for the federation partner (for example, the destination that hosts a requested resource). The configuration indicates the type of token to be created. In this case, the token type is SAML.

The identity mapping module then modifies the XML document to contain the information required to build a SAML 2.0 token.

Table 22. STSUUSER entries used to generate a SAML token, using an alias

STSUUSER element	SAML Token Information	Required?
Attribute: AuthContextClassRef	The authentication context class reference. This element is set to `password` by default regardless of the authentication method that is set in the credential. You can change the value for this element in the mapping rule.	Required
Attribute: AudienceRestriction	The audience of the audience restriction condition.	Optional
Attribute List	Additional custom attributes.	Optional

The mapping module is responsible for the following tasks:

1. Mapping Principal Attr Name to a Principal Name entry. When the token module generates the token, this Principal name is not directly used. Instead, the value in the **Name** field is sent as input to the Tivoli Federated Identity Manager alias service. The alias service obtains the alias name, name identifier, for the principal, and places the returned alias in the generated token module.

   The type must be valid for SAML. For example:

   `urn:oasis:names:tc:SAML:2.0:assertion`
2. Setting the authentication method to the `password` mechanism. This action is required by the SAML standard.

   The following code sample shows part of the default mapping rule file, ip_saml_20.xsl.

```
<!--
Note: No Principal template is necessary for identity provider on SAML 2.0 since name identifiers
will be carried in the Subject element of the assertion.
-->

 <xsl:template match="//stsuuser:AttributeList">
 <stsuuser:AttributeList>
                       <!-- First the authentcation context class ref. attribute -->
  <stsuuser:Attribute name="AuthnContextClassRef" type="urn:oasis:names:tc:SAML:2.0:assertion">
   <stsuuser:Value>urn:oasis:names:tc:SAML:2.0:ac:classes:Password</stsuuser:Value>
   </stsuuser:Attribute>
  </stsuuser:AttributeList>
 </xsl:template>
```

Figure 14. XSL code sample showing mapping of a local user identity into a SAML token, using an alias

3. Setting the audience of the audience restriction condition to the value of the STSUU element `AudienceRestriction`. If this STSUU element is not present, the audience is set to the Provider ID of the federation partner.
4. Populating the attribute statement of the assertion with the attributes in the AttributeList in the In-STSUU. This information becomes custom information in the token.

   There can be custom attributes that are required by applications that uses information that is to be transmitted between federation partners.

Mapping a SAML 2.0 token to a local identity

Map a SAML 2.0 token to a local identity to conform to the security token service universal user schema.

The service provider receives a SAML 2.0. Then, Tivoli Federated Identity Manager converts the token contents into an STSUU document that conforms to the security token service universal user schema.

Table 23. SAML token information that is converted into an STS universal user document

SAML Token Information	STSUUSER element
AuthenticationStatement/Subject/NameIdentifier	Principal Attr: Name
Additional custom attributes	AttributeList (Optional)

The token module reads the token and obtains the NameIdentifier. The token module passes the NameIdentifier, an alias, to the alias service. The alias service converts the received alias to the local identity. The token module puts the local identity into the Principal element in the STSUU document.

- The NameIdentifier alias that is returned is used to populate the `name` attribute of the Principal. This is the local user ID.

  The following code example shows the assignment of a set value for the Principal name. This code sample is from the default mapping file sp_saml_20.xsl.

```
<!--
  This will replace the principal name with the user's local name.
 -->
 <xsl:template match="//stsuuser:Principal/stsuuser:Attribute[@name='name']">
  <stsuuser:Attribute name="name" type="urn:ibm:names:ITFIM:5.1:accessmanager">
   <stsuuser:Value>
    <xsl:value-of select="//stsuuser:Principal/stsuuser:Attribute[@name='name']/stsuuser:Value"/>
   </stsuuser:Value>
  </stsuuser:Attribute>
 </xsl:template>
```

Figure 15. XSL code sample showing assignment of a value for the Principal name for a SAML 2.0 token.

- Other information from the token is used to populate Attributes in the AttributeList.

  The following code example shows the optional assignment of additional values to attributes. This code sample is from the default mapping file sp_saml_20.xsl.

```
<xsl:variable name="department">
<xsl:value-of select="//stsuuser:AttributeList/stsuuser:Attribute[@name='Department']/stsuuser:Value"/>
 </xsl:variable>

<xsl:template match="//stsuuser:AttributeList">
 <stsuuser:AttributeList>
  <stsuuser:Attribute type="urn:ibm:names:ITFIM:5.1:accessmanager">
   <xsl:attribute name="Department">
    <stsuuser:Value>
  <xsl:value-of select="//stsuuser:AttributeList/stsuuser:Attribute[@name='Department']
  /stsuuser:Value"/>
    </stsuuser:Value>
   </stsuuser:Attribute>
 </stsuuser:AttributeList>
</xsl:template>
```

Figure 16. XSL code sample showing AttributeList for a SAML 2.0 token.

Chapter 18. SAML 2.0 Attribute query

The SAML 2.0 attribute query feature extends the capability of the SAML 2.0 protocol. Traditional SAML 2.0 function requires that the identity provider sends *all* required user attributes to the federation partner. The attributes are included as part of the assertion generated during the single sign-on flow.

The SAML 2.0 attribute query feature eliminates this limitation. Administrators for identity providers can include in the single sign-on flow only the attributes that are used by most targeted applications. Applications can use a SAML 2.0 attribute query flow to obtain any attribute requirements or specialized values.

Support for attribute query provides a set of core attributes when the initial authentication context is established. You can query user information as needed during the application runtime operation. Different applications require different user information. For example, applications that require fine grained authorization require specific user entitlements to make the authorization decisions.

Attribute query supports the following modes:

Direct mode
: The requesting application issues a direct call to the identity provider to obtain any required attributes.

On-behalf mode
: The requesting application contacts the service provider, which proxies the attribute request to the identity provider.

Direct mode

In direct mode, the requesting application sends an AttributeQuery request to the SAML 2.0 federation SOAP endpoint on the identity provider. The SOAP delegate protocol finishes the necessary protocol actions and issues a SAML assertion. The SAML attribute query function uses the attribute query secure token service (STS) module to issue the assertion.

The direct mode requires the application (attribute requester) to be known to the identity provider. To make an application known at the identity provider, use the command-line interface command `manageItfimPartner` to import the requester metadata.

The single sign-on flow for direct mode is:

1. The user requires access to a resource or application and initiates a federated single sign-on flow.
2. The identity provider authenticates the user and issues a SAML assertion with a subset of attributes that most applications or resources require.
3. The application or resource determines if any additional attributes are required. If so, the application issues an `AttributeQuery` to the identity provider obtain them.
4. The identity provider returns a SAML assertion with the requested attributes.
5. The application or resource obtains the attributes returned by the identity provider in the attribute query SAML response message.

On-behalf mode

On-behalf mode requires that applications send query requests to the service provider, which then proxies them to the identity provider. The identity provider supplies the requested attributes. On-behalf mode supports two different types of requests:

- SAML 2.0 *<AttributeQuery>* requests

  The application must send AttributeQuery messages to the service provider SOAP endpoint. If an AttributeQuery request message is used, the service provider returns a SAML Response message with the corresponding assertion.
- WS-Trust Request Security Token messages.

  For this protocol, the application must send WS-Trust messages to the trust service endpoint. If the requesting application sends a WS-Trust message, the response message is a Universal User Token.

  Note: If your application is a WS-Trust client, you can use this option instead of using the SAML protocol.

The on-behalf mode limits the amount of configuration required at the identity provider for many service provider applications to query user attributes. In this mode, the service provider is the only known entity at the identity provider.

The single sign-on flow for on-behalf mode is:

1. The user requires access to a resource or application on the service provider and initiates a federated single sign-on flow.
2. The identity provider authenticates the user and issues a SAML assertion with a subset of attributes that most applications or resources require.
3. The service provider selects which attributes to make available to the resource or application. The service provider then creates the authenticated session for the user.
4. The application or resource determines if any additional attributes are required. If so, the application issues an `AttributeQuery` or a WS-Trust `RequestSecurityToken` to obtain them. The application sends the request to the service provider. The service provider proxies the request to the identity provider.
5. The Identity Provider returns a SAML assertion with the requested attributes.
6. The application or resource obtains the attributes returned by the Identity Provider in the attribute query SAML response message. If a WS-Trust request is made, the attributes are returned to the client application using a Universal User Token. If the request is a SAML AttributeQuery request, the attributes are returned in a SAMLResponse generated by the Service Provider.

Attribute query request partner

The Attribute query feature defines a new type of role. Application partners to a SAML 2.0 federation can now act in an *attribute query requester* role. This role is different from the role of service provider partner or identity provider partner.

An attribute query requester is an entity that makes SOAP-based `<AttributeQuery>` request calls to obtain user attributes.

If you plan to configure an *attribute query requester partner*, you must generate a metadata file as specified by the SAML 2.0 specification. Tivoli Federated Identity

Manager uses this metadata file to create the attribute request partner. You must use the `manageItfimPartner` command to create the partner. This command uses a response file, which contains a parameter that specifies the location of the metadata file.

Developing an attribute query STS module

The attribute query function uses an STS token module called the *attribute query module*. You must configure the module for the STS trust chain for the SAML 2.0 federation.

Before you configure attribute query, you must:

1. Determine the attributes that your resource or application wants to request from the identity provider.
2. Develop a script or module that requests the attributes. This request can be made by an XSLT or JavaScript file, a Tivoli Directory Integrator assembly line, or a custom secure token service (STS) mapping module.

Limitation with migrating from previous release of Tivoli Federated Identity Manager

Tivoli Federated Identity Manager supports migration of SAML 2.0 federations from the previous release to the current release. The attribute query feature was not available in previous releases. Without the attribute query feature, attribute query is not automatically enabled in the new release when you migrate SAML 2.0 federations from the previous release.

To enable attribute query for the federation, take the following steps after you have migrated the federation:

- Select the check box on the federation properties page to enable attribute query.
- Use the graphical user interface on the federation properties page to configure an attribute query module.
- Use the Add Partner wizard to add all partners that previously existed for the federation.

Configuring attribute query

You can configure SAML 2.0 federations and partners to support the attribute query feature.

The steps for attribute query configuration vary depending on the deployment scenario. Deployment includes the creation of a federation and the addition of a partner to the federation.

When you configure the federations, the identity provider partners, and the service provider partners, you can use a graphical user interface that prompts for attribute query settings. The section provides detailed descriptions of these settings.

Some settings for attribute query use existing values for SAML 2.0 federations. For these settings, you are not prompted for additional configuration for attribute query.

For example, providers sign or validate assertions based on the configuration settings established for the SAML 2.0 federation or partner. The federation or

partner signs or validates the attribute query assertions as required by the federation partner. You are not required to specify additional settings to enforce signing or validation.

If you install SAML 2.0 with the Typical or All profiles, signing and validation are activated automatically. If you select manual installation of profiles, the wizard prompts you to specify whether to sign and validate messages. The wizard requires these settings whether the attribute query feature is configured or not configured.

To configure your federation and partner in direct mode, complete the following tasks:

- "Creating a federation as an attribute authority"
- "Creating an attribute query request partner" on page 196

To configure your federation and partner in on-behalf mode, complete the following tasks:

- "Creating a federation as an attribute authority"
- "Creating an identity provider partner or service provider partner for an attribute authority federation" on page 194
- "Creating an attribute query request partner" on page 196

Creating a federation as an attribute authority

You can use either the administration console or the command-line interface to create a SAML 2.0 federation as an attribute authority.

Choose one of the following methods:

- "Using the administration console to create a federation as an attribute authority"
- "Using the command line interface to create a federation as an attribute authority" on page 193

Using the administration console to create a federation as an attribute authority

You can use the administration console to create a SAML 2.0 federation as an attribute authority.

About this task

The configuration for attribute query uses the same wizard as is used for all SAML federations. When you use the wizard, you activate attribute query, and are prompted to provide configuration settings.

Parameters for attribute query are described in the worksheets for federation configuration. See the topic for your partner type:

- "SAML 2.0 identity provider worksheet" on page 208.
- "SAML 2.0 service provider worksheet" on page 203.

Note: Combine the information in this procedure below with the step by step configuration instructions for SAML 2.0 federations in Chapter 19, "Establishing a SAML federation," on page 199.

Procedure

1. On the Profiles panel in the wizard, select **All** or **Manual**.
2. On the Profile Details panel, go to Attribute Query and select **Enabled**. Selection of this check box causes additional panels to be shown.
3. On the SAML Assertions panel, specify the amount of time before the issue date that an assertion is considered valid. Specify also the amount of time that the assertion is valid after being issued.

   Note: When you are using a *service provider* federation for attribute query, the federation must issue assertions. This requirement means that when you have activated attribute query for a service provider federation, the SAML assertions panel opens, and you must specify values. When you configure a service provider federation without attribute query, you are not required to set values for SAML assertions.

   The SAML Assertions panel is shown for *identity provider* federation creation regardless of whether attribute query is selected. In this type of federation, SAML assertions are issued for multiple purposes.
4. On the Attribute Module Selection panel, select one of the following choices:
   - XSLT or JavaScript transformation
   - Tivoli Directory Integrator module
   - Custom mapping module.

   Base your selection on the method you identified for your deployment when you planned the configuration.

Using the command line interface to create a federation as an attribute authority

You can use the command-line interface to create a SAML 2.0 federation as an attribute authority.

About this task

When using the command-line interface to create a SAML 2 federation, you must first create and populate a SAML 2 federation response file. To establish the SAML 2 federation as an attribute authority, you must set values in the response file for the following parameters:

- `AttributeQueryMappingRule`
- `AttributeQueryMappingRuleFileName`
- `AttributeAuthorityEnabled`
- `SignAttributeQueryRequest`
- `SignAttributeQueryResponse`

For descriptions of the parameters needed, see “SAML 2.0 attribute query federation response file parameters” on page 197.

For more information about using the command-line interface to create a SAML 2 federation and a response file, see the *IBM Tivoli Federated Identity Manager Administration Guide*.

Procedure

1. Create a SAML 2 response file.

   For example, to create a SAML 2 response file based on an existing federation:

   ```
   $AdminTask manageItfimFederation {-operation createResponseFile
   -fimDomainName domain1 -federationName idpsaml2
   -fileId c:\temp\saml2idp.rsp }
   ```

2. Edit the SAML 2 response file to set the attribute query parameters.

   In the example, the response file is `c:\temp\saml2idp.rsp`

3. Create the SAML 2 federation as an attribute authority.

   To create an identity provider or service provider federation that is activated for attribute query, use the standard syntax. There are no additional options to specify.

   For example, if the response file is `c:\temp\saml2idp.rsp`:

   ```
   $AdminTask manageItfimFederation { -operation create -fimDomainName domain1
   -fileId c:\temp\saml2idp.rsp }
   ```

Creating an identity provider partner or service provider partner for an attribute authority federation

You can create an identity provider partner or service provider partner for a SAML 2.0 federation that has been configured as an attribute authority.

When a federation has been configured to as an attribute authority, you can add partners of the following types:

- Service provider partner

  Add a traditional service provider partner to an identity provider federation. You can configure this partner to exchange attribute query request-responses with the federation provider.

- Identity provider partner

  Add a traditional identity provider partner to a service provider federation. You can configure this partner to exchange attribute query request-responses with the federation provider.

- Attribute query request partner

  This type of partner is a special case for use when the requesting application or resource does not have Tivoli Federated Identity Manager installed.

  Note: The instructions in this topic do not apply to attribute query request partners. See "Creating an attribute query request partner" on page 196.

To add either an identity provider partner or a service provider partner, see:

- "Using the administration console to create a service provider or identity provider partner"
- "Using a command-line interface to create a service provider or identity provider partner" on page 196

Using the administration console to create a service provider or identity provider partner

You can use the administration console to create a service provider or identity provider partner.

About this task

You can use the Add Partner wizard to add a service provider partner or identity provider partner to a federation. This wizard is also used for adding SAML 2.0 partners without attribute query.

When you use the wizard to add a partner to a federation, the configuration program determines if the federation is configured as an attribute query authority. If the federation is an attribute query authority, additional panels prompt you to enter more information.

The configuration panels differ slightly for identity provider partners or service provider partners. See the following table.

Configuration panel	Partner type	Description
SAML Assertions	Identity provider partner for a service provider federation *only*	The SAML Assertions settings panel permits you to specify which attributes to include in the assertion. The default value is to include all attributes. You can use this setting to specify a base set of attributes. The SAML Assertions panel also permits you to specify which attributes get encrypted, and which encryptions algorithms to use.
Attribute Module Selection	Both identity provider partner for a service provider federation and service provider partner for an identity partner federation	On the Attribute Module Selection panel, you must select one of: • XSLT or JavaScript transformation • Tivoli Directory Integrator module • Custom mapping module. Base your selection on the method you identified for your deployment when you planned the configuration.

The parameters for partner configuration for attribute query are described in the worksheets for SAML 2.0 partner configuration. See the topic for your partner type:

- "SAML 2.0 identity provider partner worksheet" on page 237
- "SAML 2.0 service provider partner worksheet" on page 230

The graphical user interface wizard for adding SAML 2.0 partners includes the panels for attribute query configuration. To configure the identity provider or service provider partner, see the SAML 2.0 instructions: "Adding your partner" on page 245

Using a command-line interface to create a service provider or identity provider partner

You can create an identity provider partner or service provider partner for a SAML 2.0 federation that has been configured as an attribute authority.

About this task

When using the command-line interface to create a partner, you must first create and populate a SAML 2 federation response file. To configure the partners to use the attribute query capability, you must set values for the following parameters in the response file:

- `AttributeQueryMappingRule`
- `AttributeQueryMappingRuleFileName`
- `ValidateAttributeQueryRequest`
- `ValidateAttributeQueryResponse`

For information about using the command-line interface to create a SAML 2 partner and partner response file, see the *IBM Tivoli Federated Identity Manager Administration Guide*.

Procedure

1. Create a SAML 2 partner response file.

   For example, to create a SAML 2 partner response file based on an existing partner:

   ```
   $AdminTask manageItfimPartner {-operation createResponseFile
   -fimDomainName domain1  -federationName fed1
   -partnerName idppartner -fileId c:\temp\saml2idp.rsp }
   ```

2. Edit the SAML 2 partner response file to set the attribute query parameters.

   In the example, the response file is `c:\temp\saml2idp.rsp`

   For descriptions of the attribute query response file parameters, see "SAML 2.0 attribute query partner response file parameters" on page 198.

3. To create an identity provider partner that is configured for attribute query, use the standard syntax.

   You can optionally specify the partner role in the command line. You are not required to specify the partner role. When the role is not specified the program automatically sets the partner role based on the federation role.

   For example, if the response file is `c:\temp\saml2idp.rsp`:

   ```
   $AdminTask manageItfimPartner { -operation create -fimDomainName domain1
   -federationName idpsaml2 -partnerName idpartner
   -fileId c:\temp\saml2idp.rsp
   -signingKeystorePwd testonly -encryptionKeystorePwd testonly }
   ```

   If you want to specify the partner role in the command line, add the `-partnerRole` option, and specify either `sp` or `idp`. For example, to specify a service provider partner:

   ```
   $AdminTask manageItfimPartner { -operation create -fimDomainName domain1
   -federationName idpsaml2 -partnerName idpartner
   -partnerRole sp
   -fileId c:\temp\saml2sp.rsp
   -signingKeystorePwd testonly -encryptionKeystorePwd testonly }
   ```

Creating an attribute query request partner

Use the command-line interface to create an attribute query request partner.

About this task

You must use the command-line interface to add an attribute query request partner to a federation. The administration graphical user interface does not provide a wizard for this task.

Use the `manageItfimPartner` command to create the partner. This command supports a partner role parameter `qr` that indicates that a query requester partner is created.

Procedure

1. Create a SAML 2 partner response file.

   For example, to create a SAML 2 attribute query request partner response file based on an existing partner:

   ```
   $AdminTask manageItfimPartner { -operation createResponseFile
   -fimDomainName fimipdomain  -federationName saml20ip
   -partnerRole qr -fileId /downloads/qr.out }
   ```

2. Edit the response file to show the location of the metadata file from the attribute query request partner. This file name is a parameter in the response file. You also must add information specific to the partner.

   For information about using the command-line interface to create a SAML 2 partner and partner response file, see the *IBM Tivoli Federated Identity Manager Administration Guide*.

3. Create an attribute requester partner:

   ```
   $AdminTask manageItfimPartner { -operation create
   -fimDomainName fimipdomain
   -federationName saml20ip -partnerName samlqr
   -partnerRole qr -fileId /downloads/qr.out
   -signingKeystorePwd testonly
   -encryptionKeystorePwd testonly}
   ```

SAML 2.0 attribute query federation response file parameters

The SAML 2.0 federation response file contains parameters that are used by attribute query.

Table 24. Attribute query parameters for federation response file

Parameter	Value	Description
AttributeQueryMappingRule	*contents of the mapping rule file*	Contains the actual mapping rule contents (XSL) that are used to format the rule, so that it can be contained in the XML response file. Use this property to specify a mapping rule without using a file on the file system. Use this property also if you are modifying a federation. If you want to edit the XSLT rule as a regular file, supply it to the response file using the **AttributeQueryMappingRuleFileName** property. This rule is used for attribute query operations.

Table 24. Attribute query parameters for federation response file (continued)

Parameter	Value	Description
AttributeQueryMappingRuleFileName	*path and file name*	Specifies path name to an XSLT file that is used as a mapping rule. When defined, it takes precedence over the **AttributeQueryMappingRule** property. This rule is used for attribute query operations.
AttributeAuthorityEnabled	*true or false*	Specifies whether the attribute query feature is configured in the federation. The value `true` activates attribute query. The value `false` disables attribute query. Default: false
SignAttributeQueryResponse	*true or false*	Specifies whether attribute query responses are signed.
SignAttributeQueryRequest	*true or false*	Specifies whether attribute query requests are signed.

SAML 2.0 attribute query partner response file parameters

The SAML 2.0 partner response file contains parameters that are used by attribute query.

Table 25. Attribute query parameters for partner response file

Parameter	Value	Description
AttributeQueryMappingRule	*contents of the mapping rule file*	Contains the actual mapping rule contents (XSL) that are used to format the rule, so that it can be contained in the XML response file. Use this property if you want to specify a mapping rule without using a file on the file system. Use this property also if you are modifying a federation. If you want to edit the XSLT rule as a regular file, supply it to the response file using the **AttributeQueryMappingRuleFileName** property. This rule is used for attribute query operations.
AttributeQueryMappingRuleFileName	*path and file name*	Specifies path name to an XSLT file that is used as a mapping rule. When defined, it takes precedence over the **AttributeQueryMappingRule** property. This rule is used for attribute query operations.
ValidateAttributeQueryResponse	*true or false*	Specifies that validation of the partner signatures takes places on attribute query responses that are received. An error is shown if the message is not signed.
ValidateAttributeQueryRequest	*true or false*	Specifies that an attribute query request that was received from the partner signature is validated. An error is shown if the message is not signed.

Chapter 19. Establishing a SAML federation

Establish a SAML federation to complete the configuration of your federation.

Complete the following tasks to configure your federation:

1. "Gathering your federation configuration information."
2. "Creating your role in the federation" on page 214.
3. "Providing guidance to your partner" on page 216.
4. "Obtaining federation configuration data from your partner" on page 218.
5. "Adding your partner" on page 245.
6. "Providing federation properties to your partner" on page 247.

Gathering your federation configuration information

The Federation wizard prompts you for information that is used in your federation. Before starting the wizard, prepare for the configuration process by gathering your configuration information using the appropriate worksheet.

About this task

Choose a worksheet based on the SAML standard that you want to use in the federation and your role in the federation.

- "SAML 1.x service provider worksheet"
- "SAML 1.x identity provider worksheet" on page 201
- "SAML 2.0 service provider worksheet" on page 203
- "SAML 2.0 identity provider worksheet" on page 208

SAML 1.x service provider worksheet

If you assume the role of the service provider in the federation, and use SAML 1.0 or SAML 1.1, record your configuration information in the following tables.

Table 26. General information for service provider in SAML 1.x federation

General Information	Description	Your value
Federation name	The unique name you give to the federation.	
Role	The role you provide in the federation. (In these instructions, you are the service provider.)	Service provider

Table 27. Contact information for service provider in SAML 1.x federation

Contact Information	Description	Your value
Company name, Company URL, and contact name and information.	Your company name and other optional information about the contact associated with your role in the federation.	

Table 28. Federation protocol for service provider in SAML 1.x federation

Federation Protocol	Description	Your value
Protocol	The SAML protocol you and your partner use in the federation.	One of the following: • SAML 1.0 • SAML 1.1

Table 29. Point of contact server information for service provider in SAML 1.x federation

Point of contact server	Description	Your value
Point of contact server URL	The URL that provides access to the endpoints on the point of contact server.	

Table 30. Signature information for service provider in SAML 1.x federation

Signatures	Description	Your value
Sign Artifact Resolution Requests	A check box that indicates that you will sign request messages. Default value: No signing. The check box is not selected.	One of the following: • Sign request messages. (Select check box.) • Do not sign request messages. (Clear check box.)
Select Signing Key • Keystore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the keystore • Private key you will use to sign request messages	If you select the check box, you must supply the signing key that you will use to sign the requests. **Note:** Be sure you have created the key and imported it into the appropriate keystore in the Tivoli Federated Identity Manager key service prior to this task. See Chapter 8, "Setting up message security," on page 49.	Keystore name: Keystore password: Key alias name:

Table 31. Identity mapping information for service provider in SAML 1.x federation

Identity mapping	Description	Your value
Identity mapping options One of the following: • An XSL transformation (XSLT) file containing mapping rules • A custom mapping module	The type of identity mapping you will use. You must know whether to use an XSLT file for identity mapping or a custom mapping module. Custom mapping is an advanced option. If you plan to use this option, your mapping module must be created and added to the environment as a module type and module instance *before* you can use it in your configuration. If you choose to use an XSLT file, you must have the file ready to use for the federation.	One of the following values: • XSLT file (path and name): • Custom mapping module instance name:

When you have completed the tables, continue with the instructions in "Creating your role in the federation" on page 214.

SAML 1.x identity provider worksheet

If you assume the role of the identity provider in the federation, and use SAML 1.0 or SAML 1.1, record your configuration information in the following tables.

Table 32. General information for identity provider in SAML 1.x federation

General Information	Description	Your value
Federation name	The unique name you give to the federation.	
Role	The role you provide in the federation. (In these instructions, you are the identity provider.)	Identity provider

Table 33. Contact information for identity provider in SAML 1.x federation

Contact Information	Description	Your values
Company name, Company URL, and contact name and information.	Company name and optionally other information about the contact associated with the federation.	Company name:

Table 34. Federation protocol information for identity provider in SAML 1.x federation

Federation Protocol	Description	Your value
Protocol	The SAML protocol you and your partner use in the federation.	One of the following: • SAML 1.0 • SAML 1.1

Table 35. Point of contact server for identity provider in SAML 1.x federation

Point of Contact Server	Description	Your value
Point of contact server URL	The URL that provides access to the endpoints on the point of contact server.	

Table 36. Signing information for identity provider in SAML 1.x federation

Signatures	Description	Your value
Signature options: • **SAML messages for Browser POST profile are signed** (required) • **Sign SAML messages for artifact profile** (optional)	• When browser POST is used as the profile, SAML messages must be signed. Therefore, it is pre-selected and cannot be deselected. • You have the option of also signing the SAML messages when browser artifact is used.	One of the following: • Sign browser artifact messages. (Select check box.) • Do not sign browser artifact messages. (Clear check box.)
Select Signing Key • Keystore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the keystore • Private key you will use for signing	Because Browser POST messages must be signed, you are required to supply a signing key. If you select to also sign messages when browser artifact is used, the same signing key is used to sign them. **Note:** Be sure you have created the key and imported it into the appropriate keystore in the Tivoli Federated Identity Manager key service prior to this task. See Chapter 8, "Setting up message security," on page 49.	Keystore name: Keystore password: Key alias name:

Table 37. SAML Message Settings information for identity provider in SAML 1.x federation

SAML Message Settings	Description	Your value
Artifact Resolution Service URL	The URL for your artifact resolution endpoint. (**Note:** The value for this field is filled in automatically using the point of contact server URL you specified earlier.)	
Artifact Cache Lifetime (seconds)	The artifact cache lifetime in seconds. Default value: 30 seconds.	

Table 37. SAML Message Settings information for identity provider in SAML 1.x federation (continued)

SAML Message Settings	Description	Your value
Allow IBM Protocol Extension	You must specify whether you will allow the use of the IBM PROTOCOL extension. The extension allows a query-string parameter that specifies whether browser artifact or browser POST is used. For more information, see "SAML 1.x" on page 165.	One of the following: • Allow IBM Protocol Extension. (Select the check box.) • Do not allow Protocol Extension. (Clear the check box.)

Table 38. Token Settings information for identity provider in SAML 1.x federation

Configure Token Settings	Description	Your value
Amount of time before the issue date that an assertion is considered valid	The number of seconds that an assertion is considered valid before its issue date. Default value: 60	
Amount of time the assertion is valid after being issued	The number of seconds that an assertion is considered valid after its issue date. Default value: 60	

Table 39. Identity mapping information for identity provider in SAML 1.x federation

Identity mapping	Description	Your value
Identity mapping options One of the following: • An XSL transformation (XSLT) file containing mapping rules • A custom mapping module	The type of identity mapping you will use. You must know whether to use an XSLT file for identity mapping or a custom mapping module. Custom mapping is an advanced option. If you plan to use this option, your mapping module must be created and added to the environment as a module type and module instance *before* you can use it in your configuration. If you choose to use an XSLT file, you must have the file ready to use for the federation.	One of the following values: • XSLT file (path and name): • Custom mapping module instance name:

When you have completed the tables, continue with the instructions in "Creating your role in the federation" on page 214.

SAML 2.0 service provider worksheet

If you assume the role of the service provider in the federation, and use SAML 2.0, record your configuration information in the following tables.

Table 40. General information for service provider in SAML 2.0 federation

General Information	Description	Your value
Federation name	The unique name you give to the federation.	
Role	The role you provide in the federation. (In these instructions, you are the service provider.)	Service provider

Table 41. Contact information for service provider in SAML 2.0 federation

Contact Information	Description	Your value
Company name, Company URL, and contact name and information.	Your company name and optionally other information about the contact associated with your role in the federation.	

Table 42. Federation protocol for service provider in SAML 2.0 federation

Federation Protocol	Description	Your value
Protocol	The SAML protocol you and your partner use in the federation.	SAML 2.0

Table 43. Point of contact server information for service provider in SAML 2.0 federation

Point of contact server	Description	Your value
Point of contact server URL	The URL that provides access to the endpoints on the point of contact server.	

Table 44. Profile selection and configuration information for service provider in SAML 2.0 federation

Profile selection	Description	Your value
SAML 2.0 profile options: Choose one of the following profile options:	The profile for your federation. For more information about profiles, see "SAML 2.0" on page 167.	One of the following: • Basic • Typical • All • Manual
Basic: Web Browser SSO, Single Logout	This setting enables the following profiles with all supported bindings: • Web browser SSO • Single Logout	(No additional values required.)

Table 44. Profile selection and configuration information for service provider in SAML 2.0 federation (continued)

Profile selection	Description	Your value
Typical: Web Browser SSO, Single Logout and Name Identifier Management	This setting enables the following profiles with all supported bindings: • Web browser SSO • Single Logout • Enhanced client or proxy • Name Identifier Management	(No additional values required.)
Enable all profiles and bindings	If you choose **Enable all profiles and bindings**, you must be ready to provide the following information on subsequent panels: **Identity Provider Discovery Settings panel:** • Common domain name • Common domain cookie service URL **Enhanced Client Proxy panel:** • HTTP headers	**Identity Provider Discovery Settings** • Common domain name: • Common domain cookie service URL: **Enhanced Client Proxy** HTTP headers:
Manual: Choose individual profiles and bindings	If you choose **Manual**, you must be ready to select individual profiles and supported bindings.	Profiles and bindings:

Table 45. Signature information for service provider in SAML 2.0 federation

Signatures	Description	Your value
Require signature on incoming messages and assertions	A check box that specifies that your partner uses its private key to sign the message and assertion. Default value: The check box is checked.	One of the following: • Partner will sign. (Check box is selected.) • Partner will not sign. (Check box is not selected.)
Select which outgoing messages and assertions you will sign	Buttons that indicate which outgoing messages you will sign. The default setting is for the typical set of outgoing SAML messages and assertions (except for ArtifactResponse and AuthnResponse) to be signed.	One of the following: • Typical set of outgoing SAML messages are signed. • All outgoing SAML messages and assertions are signed. • No outgoing SAML messages and assertions are signed.

Table 45. Signature information for service provider in SAML 2.0 federation (continued)

Signatures	Description	Your value
Select Signing Key • Keystore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the keystore • Private key you will use to sign messages	If you sign messages and assertions, you must supply the signing key that you use to sign them. **Note:** Be sure you have created the key and imported it into the appropriate keystore in the Tivoli Federated Identity Manager key service prior to this task. See Chapter 8, "Setting up message security," on page 49.	Keystore name: Keystore password: Key alias name:

Table 46. Encryption information for service provider in SAML 2.0 federation

Encryption	Description	Your value
Encryption Key: • Keystore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the keystore • Public/private key pair that will be used for data you receive from your partner.	A public/private key pair used in encryption. Your partner uses the public key to encrypt data to you. You will use the private key to decrypt data that your partner sends to you. You must specify the key pair to use. **Note:** Be sure you have created the key and imported it into the appropriate keystore in the Tivoli Federated Identity Manager key service prior to this task.	Keystore name: Keystore password: Key alias name:

Table 47. SAML message settings for service provider in SAML 2.0 federation

Message settings	Description	Your value
Message Options: • Message Lifetime in seconds • Artifact Lifetime in seconds • Session Timeout	Amount of time in seconds that messages, artifacts, and sessions are valid. The default values are: • Message lifetime: 300 • Artifact lifetime: 120 • Session timeout: 7200	Message Lifetime in seconds: Artifact Lifetime in seconds: Session Timeout:
Single sign-On Options • Identity Provider is allowed to interact with user • Single sign-on is passive • Force Identity Provider to authenticate user	Specifies how the identity provider is to interact with the users.	One of the following: • Identity Provider is allowed to interact with user • Single sign-on is passive • Force Identity Provider to authenticate user

Table 47. SAML message settings for service provider in SAML 2.0 federation (continued)

Message settings	Description	Your value
SOAP Endpoint	The URL of the SOAP endpoint. Default value: The value in this field is based on the point of contact server URL that you supplied earlier. **Note:** If the SOAP binding is not used in the profile you selected, this field is not shown.	

Table 48. Attribute query information for service provider

Attribute query	Description	Your value
Enabled	Indicates if the provider is permitted to act as the attribute authority. If the check box is selected, the attribute query profile is activated.	
Amount of time before the issue date that an assertion is considered valid	The number of seconds that an assertion is considered valid before its issue date. Default value: 60	
Amount of time the assertion is valid after being issued	The number of seconds that an assertion is considered valid after its issue date. Default value: 60	

Table 49. Attribute query mapping information for service provider in SAML 2.0 federation

Attribute query mapping	Description	Your value
Attribute query mapping options One of the following: • An XSL transformation file or JavaScript containing mapping rules • Tivoli Directory Integrator mapping module • A custom mapping module	The type of attribute query mapping you are using. You must select either an XSLT file, a Tivoli Directory Integrator mapping module, or a custom mapping module. If you use an XSLT file, you must have the file created before you configure the federation. The Tivoli Directory Integrator mapping module is an STS module. Custom mapping is an advanced option. If you use this option, you must create and add a new module type and module instance *before* you can use it in your configuration.	One of the following values: • XSLT file path • Tivoli Directory Integrator mapping module • Custom mapping module instance name

Table 50. Identity mapping information for service provider in SAML 2.0 federation

Identity mapping	Description	Your value
Identity mapping options One of the following: • An XSL transformation file containing mapping rules • A custom mapping module	The type of identity mapping you will use. You must know whether to use an XSLT file for identity mapping or a custom mapping module. Custom mapping is an advanced option. If you plan to use this option, your mapping module must be created and added to the environment as a module type and module instance *before* you can use it in your configuration. If you choose to use an XSLT file, you must have the file ready to use for the federation.	One of the following values: • XSLT file (path and name): • Custom mapping module instance name:

When you have completed the tables, continue with the instructions in "Creating your role in the federation" on page 214.

SAML 2.0 identity provider worksheet

If you will be the identity provider in the federation and will use SAML 2.0, record your configuration information in the following tables.

Table 51. General information for identity provider in SAML 2.0 federation

General Information	Description	Your value
Federation name	The unique name that you give to the federation.	
Role	The role that you provide in the federation. (In these instructions, you are the identity provider.)	Identity provider

Table 52. Contact information for identity provider in SAML 2.0 federation

Contact Information	Description	Your value
Company name, Company URL, and contact name and information.	Your company name and other optional information about the contact associated with your role in the federation.	

Table 53. Federation protocol for identity provider in SAML 2.0 federation

Federation Protocol	Description	Your value
Protocol	The SAML protocol you and your partner use in the federation.	SAML 2.0

Table 54. Point of contact server information for identity provider in SAML 2.0 federation

Point of contact server	Description	Your value
Point of contact server URL	The URL that provides access to the endpoints on the point of contact server.	

Table 55. Profile selection and configuration information for identity provider in SAML 2.0 federation

Profile selection	Description	Your value
SAML 2.0 profile options: Choose one of the following profile options:	The profile for your federation. For more information about profiles, see "SAML 2.0" on page 167.	One of the following: • Basic • Typical • All • Manual
Basic: Web Browser SSO, Single Logout	This setting enables the following profiles with all supported bindings: • Web browser SSO • Single Logout	(No additional values required.)

Table 55. Profile selection and configuration information for identity provider in SAML 2.0 federation (continued)

Profile selection	Description	Your value
Typical: Web Browser SSO, Single Logout and Name Identifier Management	This setting enables the following profiles with all supported bindings: • Web browser SSO • Single Logout • Enhanced client or proxy • Name Identifier Management	(No additional values required.)
Enable all profiles and bindings	If you choose **Enable all profiles and bindings**, you must be ready to provide the following information on subsequent panels: **Identity Provider Discovery Settings panel** • Common domain name • Common domain cookie service URL • Common domain cookie lifetime in seconds. Default value: 1 **Enhanced Client Proxy panel** • HTTP headers	**Identity Provider Discovery Settings panel** • Common domain name • Common domain cookie service URL • Common domain cookie lifetime in seconds. Default value: 1 **Enhanced Client Proxy panel** • HTTP headers
Manual: Choose individual profiles and bindings	If you choose **Manual**, you must be ready to select individual profiles and supported bindings.	Profiles and bindings:

Table 56. Signature information for identity provider in SAML 2.0 federation

Signatures	Description	Your value
Require signature on incoming messages and assertions	A check box that specifies that your partner uses its private key to sign the message and assertion. Default value: The check box is checked.	One of the following: • Partner will sign. (Check box is selected.) • Partner will not sign. (Check box is not selected.)
Select which outgoing messages and assertions you will sign	Buttons that indicate which outgoing messages you sign. The default setting is for the typical set of outgoing SAML messages and assertions (except for ArtifactResponse and AuthnResponse) to be signed.	One of the following: • Typical set of outgoing SAML messages are signed. • All outgoing SAML messages and assertions are signed. • No outgoing SAML messages and assertions are signed.

Table 56. Signature information for identity provider in SAML 2.0 federation (continued)

Signatures	Description	Your value
Select Signing Key • Keystore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the keystore • Private key you will use to sign messages	If you sign messages and assertions, you must supply the signing key that you use to sign them. **Note:** Be sure you have created the key and imported it into the appropriate keystore in theTivoli Federated Identity Manager key service prior to this task.	Keystore name: Keystore password: Key alias name:

Table 57. Encryption information for identity provider in SAML 2.0 federation

Encryption	Description	Your value
Encryption Key: • Keystore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the keystore • Public/private key pair that will be used for data you receive from your partner.	A public/private key pair used in encryption. Your partner uses the public key to encrypt data to you. You use the private key to decrypt data that your partner sends to you. You must specify the key pair to use. **Note:** Be sure you have created the key and imported it into the appropriate keystore in the Tivoli Federated Identity Manager key service prior to this task. See Chapter 8, "Setting up message security," on page 49.	Keystore name: Keystore password: Key alias name:

Table 58. SAML message settings for identity provider in SAML 2.0 federation

Message settings	Description	Your value
Message Options: • Message Lifetime in seconds • Artifact Lifetime in seconds • Session Timeout	Amount of time in seconds that messages, artifacts, and sessions are valid. The default values are: • Message lifetime: 300 • Artifact lifetime: 120 • Session timeout: 7200	Message Lifetime in seconds: Artifact Lifetime in seconds: Session Timeout:
Require Consent to Federate	If you select this check box, you are required to present a page to the user to verify that the user has made a federation request. Default value: Requires consent to federate.	One of the following: • Require Consent to Federate (Check box is selected.) • Do not require consent to federate. (Check box is not selected.)

Table 58. SAML message settings for identity provider in SAML 2.0 federation (continued)

Message settings	Description	Your value
SOAP Endpoint	The URL of the SOAP endpoint. Default value: The value in this field is based on the point of contact server URL that you supplied earlier. **Note:** If the SOAP binding is not used in the profile you selected, this field is not displayed.	

Table 59. Token Settings information for identity provider in SAML 2.0 federation

Configure Token Settings	Description	Your value
Amount of time before the issue date that an assertion is considered valid	The number of seconds that an assertion will be considered valid before its issue date. Default value: 60	
Amount of time the assertion is valid after being issued	The number of seconds that an assertion will be considered valid after its issue date. Default value: 60	

Table 60. Attribute query information for identity provider

Attribute query	Description	Your value
Enabled	Indicates if the provider is permitted to act as the attribute authority. If selected, the attribute query profile is activated.	

Table 61. Attribute query mapping information for identity provider

Attribute query mapping	Description	Your value
Attribute query mapping options One of the following: • An XSL transformation file or JavaScript containing mapping rules • Tivoli Directory Integrator mapping module • A custom mapping module	The type of attribute query mapping you are using. You must select either an XSLT file, a Tivoli Directory Integrator mapping module, or a custom mapping module. If you use an XSLT file, you must have the file created before you configure the federation. The Tivoli Directory Integrator mapping module is an STS module. Custom mapping is an advanced option. If you use this option, you must create and add a new module type and module instance *before* you can use it in your configuration.	One of the following values: • XSLT file path • Tivoli Directory Integrator mapping module • Custom mapping module instance name

Table 62. Identity mapping information for identity provider in SAML 2.0 federation

Identity mapping	Description	Your value
Identity mapping options One of the following: • An XSL transformation file containing mapping rules • A custom mapping module	The type of identity mapping you will use. You must know whether to use an XSLT file for identity mapping or a custom mapping module. Custom mapping is an advanced option. If you plan to use this option, your mapping module must be created and added to the environment as a module type and module instance *before* you can use it in your configuration. If you choose to use an XSLT file, you must have the file ready to use for the federation.	One of the following values: • XSLT file (path and name): • Custom mapping module instance name:

When you have completed the tables, continue with the instructions in "Creating your role in the federation" on page 214.

Creating your role in the federation

Use the console to create a federation. To begin, the Federation Wizard prompts you to supply the necessary information about your role in the federation. For descriptions of the fields that you are prompted for by the wizard, see the online help.

Before you begin

Before beginning this procedure, complete the worksheet that is appropriate for the SAML standard you will use and for your role in the federation:

- "SAML 1.x service provider worksheet" on page 199
- "SAML 1.x identity provider worksheet" on page 201
- "SAML 2.0 service provider worksheet" on page 203
- "SAML 2.0 identity provider worksheet" on page 208

About this task

During the configuration, you might be prompted to restart WebSphere Application Server. Make sure the server has restarted completely before continuing with the task.

Procedure

1. Log on to the console
2. Select **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations**. The Federations portlet shows several action buttons.
3. Click **Create**. The Federation Wizard starts. The General Information panel opens.
4. Use your worksheet to complete the panels that the Federation wizard open. Use your completed worksheet as a guide for completing the fields that are displayed. If you need to go back to a previous panel, click **Back**. If you want to end the configuration, click **Cancel**. Otherwise, click **Next** after you complete each panel. When you have completed all configuration panels, the Summary panel opens.
5. Verify that the configuration settings are correct.
6. Click **Finish**. The Create Federation Complete portlet opens.
7. You can add your partner now or later. Choose one:
   - Click **Add partner** to start the Partner Wizard and add your partner's configuration using the steps described in:
     a. "Obtaining federation configuration data from your partner" on page 218, including complete the appropriate worksheet for your partner's role in the federation.
     b. "Adding your partner" on page 245.
   - To add your partner at a later time, click **Done**. You will return to the Federations panel.

Configuring a WebSEAL point of contact server for the SAML federation

When you plan to use WebSEAL as the point of contact server, you must configure it for SAML federation.

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.

These instructions assume that the WebSEAL point of contact profile has been activated.

About this task

The federation wizard provides a button that you can use to obtain the Tivoli Federated Identity Manager configuration utility tool. You must obtain the tool and run it. To configure WebSEAL as the point of contact server, complete the following steps in this procedure.

Procedure

1. After creating the federation, click **Load configuration changes to Tivoli Federated Identity Manager runtime** to reload your changes.

   Note: The management console gives you the option of adding a partner now, but for this initial configuration of the federation other tasks are completed first.
2. Click **Done** to return to the Federations panel.
3. Click **Download Tivoli Access Manager Configuration Tool**.
4. Save the configuration tool to the file system on the computer that hosts the WebSEAL server.
5. Run the configuration tool from a command line. The syntax is:

   ```
   java -jar /download_dir/tfimcfg.jar -action tamconfig
   -cfgfile webseald-instance_name.conf
   ```

   Note: If Federal Information Processing Standards (FIPS) is enabled in your environment, the secure socket connection factory must be specified. For example:

   ```
   java -jar /download_dir/tfimcfg.jar -action tamconfig
   -cfgfile webseald-instance_name.conf -sslfactory TLS
   ```

   You must know the Tivoli Access Manager administration user (default: sec_master) and administration user password. The utility configures endpoints on the WebSEAL server, creates a WebSEAL junction, attaches the appropriate ACLs, and enables the necessary authentication methods.

Example

For example, when you have placed tfimcfg.jar in `/tmp`, and the WebSEAL instance name is `default`, the command is:

```
java -jar /tmp/tfimcfg.jar -action tamconfig -cfgfile webseald-default
```

For more information, see Appendix A, “tfimcfg reference,” on page 753.

Configuring WebSphere as a point of contact server

Tivoli Federated Identity Manager is configured by default to use Tivoli Access Manager WebSEAL as the default point of contact server. To configure WebSphere as your point of contact server, you must make a configuration change.

Procedure

1. Log on to the administration console.
2. Select **Tivoli Federated Identity Manager** > **Manage Configuration** > **Point of Contact**.
3. Select **WebSphere**
4. Click **Make Active**.

Results

The WebSphere server is now configured to be the point of contact server.

Providing guidance to your partner

When you are working with partners to establish a federation, provide information to them and collect information from them.

Depending on the role you assume in your federations, you might find that you must give guide or assist to your partner, aside from providing them with configuration information. The experience of your partner can help you decide the best way to provide guidance.

Partners who have experience with single sign-on might need limited guidance, such as through phone calls or e-mail. However, partners who are new to single sign-on might need an orientation, such as through a tutorial or written description.

The time at which you provide guidance is up to you. You might want to provide it at the same time you solicit information from your partner. You might also want to share introductory information in the early stages of your federated relationship, before any configuration has taken place.

Use the outline below to help you prepare a guidance document for your partner. The outline assumes that you are the partner who is providing guidance; however, if you are the partner who needs guidance, consider providing the outline to your partner or modifying the outline into a questionnaire that you can use to request information from your partner.

Integration Guide outline

I. Introduction

a. Describe single sign-on and consider explaining your use of Tivoli Federated Identity Manager.

b. Define terminology such as federation, identity provider, service provider, and possibly protocol, profile, and binding.

c. Identify which role you assume and which your partner assumes in the federation.

d. Describe how the end users interact with your site and with the site of your partner.

For example, identify the service that the end users are trying to access. Consider including a graphic that identifies the flow of activities among the participants such as the end user signing on, the identity provider authenticating the user, the service provider granting access, and the end user accessing the service.

II. Technical specifications

a. State requirements or options for protocol, binding, and profile.

For example, perhaps you require your partner to use SAML 1.1 with Browser Artifact. Or perhaps you require your partner to use SAML 1.1 but the profile type is the partner's choice.

b. Explain assertion requirements or options.

For example, you might require that the partner include specific fields in the assertion, such as a user group mapping key with an individual identifier. Or, you might need to explain that assertion options need to be specified such as assertion lifetime, artifact lifetime (if using Browser Artifact), and signing information.

c. Present any limitations about the types of devices that can use the single sign-on function.

For example, the federation might support only Web browser interaction from end users.

d. Describe auditing and logging requirements. For more information, refer to the *IBM Tivoli Federated Identity Manager Auditing Guide*.

e. Explain how end users experience event messages when interacting with the federation.

For example, if you are a service provider you might provide customization options to your partner for how end users log out or receive system messages about timeouts or other events. If you are an identity provider, you might provide customization options to your partner for how its end users log in.

f. Agree how you and your partner will synchronize your system clocks.

III. Security

a. State SSL requirements.

b. Request certificate information (such as the name of the certificate authority that issued the certificate of your partner or a copy of the certificate of your partner).

c. Explain signing requirements or options.

IV Data exchange

Establish how federation data, including keys are exchanged.

In SAML 1.x federations, data can be exchanged through a metadata file or manually. In SAML 2.0, a metadata file must be used.

If a manual method is used, list the information that you require from your partner. Use the worksheets in “Obtaining federation configuration data from your partner” on page 218 and “Providing federation properties to your partner” on page 247.

V Testing

Explain your capability for testing the federation and include any requirements that your partner must follow before using the federation in a production environment. Consider including the URLs that are needed by your partner for testing purposes.

For example, if you are the service provider in a SAML 1.x federation, you might need to provide a target URL and assertion consumer URL to your partner.

VI. Production

Explain what conditions must be met before the federation is ready for production. You might provide production URLs or explain how you provide those URLs at a later time.

VII. Support

Explain how the end user or administrator support is handled in the federation.

VIII. Partner worksheet

At various points throughout the preceding sections, you might have requested information from your partner or explained why you would be requesting that information.

At the end of your document, consider adding a worksheet where your partner can record that requested information. The worksheet might contain fields, such as:

- Endpoint URLs for testing
- Endpoint URLs for production
- Contact information
- SSL certificate information (name of the certificate authority, and so on)
- Signing information (what is signed, what must be validated, and so on)
- Data exchange method (manual or metadata). If a manual method is used, you might need to add other fields to the worksheet to request the needed information.

Obtaining federation configuration data from your partner

You must obtain configuration information from your partner before you can add that partner to a federation.

The partner can export the federation configuration to a metadata file or, if the partner is using SAML 1.x, the partner can manually communicate the federation configuration to you. (Configuring partners manually is not supported in SAML 2.0 federations.)

To help you gather the appropriate information from your partner, complete the appropriate worksheet for the SAML standard you plan to use in the federation and for the role that your partner will have in the federation:

- *If you are the identity provider*, add a service provider partner. Use the service provider partner worksheet for the SAML standard you are using in your federation:
  - "SAML 1.x service provider partner worksheet" on page 219
  - "SAML 2.0 service provider partner worksheet" on page 230
- *If you are the service provider*, add an identity provider partner. Use the identity provider partner worksheet for the SAML standard you are using in your federation:
  - "SAML 1.x identity provider partner worksheet" on page 224
  - "SAML 2.0 identity provider partner worksheet" on page 237

When you have gathered the configuration information of your partner, you can then use the Partner wizard in the console to add the federation properties of your partner. See "Adding your partner" on page 245.

SAML 1.x service provider partner worksheet

You must add a service provider partner to your federation if you are an identity provider who uses SAML 1.x. Some information can be supplied to you in a metadata file, or all of the information can be supplied to you manually.

Use the following worksheet to gather the necessary information from your partner. Modify this worksheet to reflect the specific information that you need from your partner. You must also ask your partner to complete that modified worksheet.

Table 63. Metadata options for adding service provider partner in SAML 1.x federation

Metadata Options	Description	Your values
Enter SAML settings manually **Import metadata file**	Specifies how you must enter data about the partner. You can receive a metadata file from your partner or enter the information of your partner manually. If you choose to import a metadata file, you need the file name and its location.	Choose either: • Enter SAML settings manually • Import metadata file and specify file name and path:

Table 64. Contact information for service provider partner in SAML 1.x federation

Contact Information	Description	Your value
Note: This panel opens only if you are entering the partner information manually.		
Company Name, URL, and contact person information	Company name and optionally other information about the contact associated with the federation.	Company name:

Table 65. SAML message settings for service provider partner in SAML 1.x federation

SAML Message Settings	Description	Your value
Note: This panel opens only if you are entering the partner information manually.		
Provider ID	The URL for the point of contact server of the service provider, which is used as the Provider ID.	Provider ID:
Assertion Consumer Service URL	The URL for the assertion consumer service endpoint at the service provider site.	Assertion Consumer Service URL:
Partner uses Browser POST profile for Single Sign-On	A check box that indicates that the service provider partner uses Browser POST.	One of the following: • Partner uses Browser POST (Select check box.) • Partner does not use Browser POST (Clear check box.)

Table 66. Signature validation information for service provider partner in SAML 1.x federation

Signatures	Description	Your value
Signature Algorithm for signing SAML Messages	Specifies the signature algorithm to use for the transaction. The selected key used to sign the SAML messages must match the option chosen in the drop-down menu to prevent signature failure. Select the signature algorithm from the following options. • RSA-SHA1 • DSA-SHA1 • RSA-SHA256	
Validate Signatures on Artifact Requests	You can validate the SAML message signatures when browser artifact is used. To use this option, select the Validate Signatures check box.	One of the following: • Validate signatures for artifact. (Select check box.) • Do not validate signatures for artifact. (Clear check box.)
Select Validation truststore or key • Truststore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the truststore • Public key you will use for validation	If you select to validate messages when browser artifact is used, you must provide a key for the validation. The key must be the public key that corresponds to the private key that your partner uses to sign the messages. **Note:** If you are importing the data of your partner, the key is supplied in the metadata file. You are prompted to choose a keystore for the key. Be sure that you have created the keystore before this task. If you are manually entering the data of your partner, be sure that you have obtained the key. Then, import the key into the appropriate keystore in the Tivoli Federated Identity Manager key service before this task. See Chapter 8, "Setting up message security," on page 49.	**Metadata method:** • Truststore name: • Truststore password: • Label for key: **Manual method:** • Truststore name: • Truststore password: • Key alias name:

Table 67. Security token settings information for service provider partner in SAML 1.x federation

Configure Security Token	Description	Your value
Sign SAML Assertions	You have the option of signing SAML assertions.	One of the following: • Enable SAML signatures. (Select check box.) • Do not enable signatures. (Clear check box.)
Select Signing Key • Keystore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the keystore • Private key you will use for signing the assertion.	If you choose to sign the assertion signatures, you must select a keystore and a key. **Note:** Create the keystore and key before this task. See Chapter 8, "Setting up message security," on page 49.	• Keystore name: • Keystore password: • Key alias name:
Include the X509 certificate data	If you choose to sign the SAML assertion, specify whether you want the BASE64 encoded certificate data to be included with your signature. The default action is to include the X.509 certificate data (**Yes**). Or, you can also choose to exclude the X.509 certificate data (**No**).	
Include the X509 Subject Issuer Details	If you choose to sign the SAML assertion, specify whether you want the issuer name and the certificate serial number to be included with your signature. The default action is to exclude (**No**) the X.509 subject issuer details . Or, you can choose to include the X.509 subject issuer details (**Yes**).	
Include the X509 Subject Name	If you choose to sign the SAML assertion, specify whether you want the subject name to be included with your signature. The default action is to exclude the X.509 subject name (**No**). Or, you can choose to include the X.509 subject name (**Yes**).	

Table 67. Security token settings information for service provider partner in SAML 1.x federation (continued)

Configure Security Token	Description	Your value
Include the X509 Subject Key Identifier	If you choose to sign the SAML assertion, specify whether you want the X.509 subject key identifier to be included with your signature. The default action is to exclude the subject key identifier (**No**). Or, you can choose to include the X.509 subject key identifier (**Yes**).	
Include the Public Key	If you choose to sign the SAML assertion, specify whether you want the public key to be included with your signature. The default action is to exclude the public key (**No**). Or, you can choose to include the public key (**Yes**).	
Include the InclusiveNamespaces element	If you choose to sign the SAML assertion, you can select to use the InclusiveNamespaces element in the canonicalization of the assertion during signature creation. The default is unchecked.	
Signature Algorithm for signing SAML Assertions	Specifies the signature algorithm to use for the transaction. The selected key used to sign the SAML assertions must match the option chosen in the drop-down menu to prevent signature failure. Select the signature algorithm from the following options. • RSA-SHA1 • DSA-SHA1 • RSA-SHA256	

Table 67. Security token settings information for service provider partner in SAML 1.x federation (continued)

Configure Security Token	Description	Your value
Include the following attribute types	Select the check box to specify the types of attributes to include in the assertion. The asterisk (*), which is the default setting, indicates that all of the attribute types that are specified in the identity mapping file, or by the custom mapping module are included in the assertion. To specify one or more attribute types individually, type each attribute type in the box. For example, if you want to include only attributes of type urn:oasis:names:tc:SAML: 2.0:assertion, enter that value in the box. Use && to separate multiple attribute types.	

Table 68. Identity mapping information for service provider partner in SAML 1.x federation

Identity mapping	Description	Your value
Identity mapping options One of the following: • A custom mapping module • An XSL transformation file containing mapping rules • Leave all options blank to use the identity mapping option that is currently defined for the federation.	The type of identity mapping to use with this partner. You can leave these fields blank, if you want this partner to use the identity mapping option that is already configured for your federation. Or, you can choose a specific mapping option to use with this specific partner. To choose a mapping option, you must know whether to use an XSLT file for identity mapping or a custom mapping module. Custom mapping is an advanced option. If you plan to use this option, first, your mapping module must be created and added to the environment as a module type and module instance. Then, you can use it in your configuration. If you choose to use an XSLT file, you must have the file ready to use for the federation.	Leave all options blank to use existing mapping configuration. Or, use one of the following values: • XSLT file (path and name): • Custom mapping module instance name:

When you have completed this worksheet, continue with the steps in "Adding your partner" on page 245.

SAML 1.x identity provider partner worksheet

You must add an identity provider partner to your federation if you are a service provider who uses SAML 1.x. Some information can be supplied to you in a metadata file, or all of the information can be supplied to you manually.

Use the following worksheet to gather the necessary information from your partner. Modify this worksheet to reflect the specific information that you need from your partner. You must also ask your partner to complete the modified worksheet.

Table 69. Metadata options for adding identity provider partner in SAML 1.x federation

Metadata Options	Description	Your values
Enter SAML settings manually **Import metadata file**	Specifies how to enter data about the partner. You can receive a metadata file from your partner or enter the information of your partner manually. If you choose to import a metadata file, you need the file name and the location of the metadata.	Choose either: • Enter SAML settings manually • Import metadata file and specify file name and path:

Table 70. Contact information for identity provider partner in SAML 1.x federation

Contact Information	Description	Your value
Note: This panel opens only if you are entering the partner information manually.		
Company Name, URL, and contact person information	Company name and optionally other information about the contact associated with the federation.	Company name:

Table 71. SAML message settings for identity provider partner in SAML 1.x federation

SAML Message Settings	Description	Your value
Note: This panel opens only if you are entering the partner information manually.		
Provider ID	The URL for the point of contact server of the service provider, which is used as the Provider ID.	Provider ID:
Source ID • Generate Source ID automatically • Enter explicit value for source ID	You have the option of generating a source ID for the partner or providing one.	Source ID:
Endpoints • Intersite Transfer Service URL • Artifact Resolution Service URL	The URLs for the Intersite Transfer Service and Artifact Resolution Service endpoints.	Intersite Transfer Service URL: Artifact Resolution Service URL:

Table 72. Signature validation information for identity provider partner in SAML 1.x federation

Signature Validation	Description	Your value
Signature Algorithm for singing SAML Artifact Resolution Requests	Specifies the signature algorithm to use for the transaction. The selected key used to sign the SAML Artifact Resolution Requests must match the option chosen in the drop-down menu to prevent signature failure. **Note:** This option is not shown if you did not choose to sign the Artifact resolution Request for the Federation. Select the signature algorithm from the following options. • RSA-SHA1 • DSA-SHA1 • RSA-SHA256	
SAML Messages for Browser POST are signed and must be validated (required) **Validate Signatures on SAML Messages for Artifact Profile** (optional)	• When browser POST is used as the profile, SAML messages must be signed and validated. Therefore, this option is pre-selected and cannot be cleared. • You also have the option of also validating the SAML message signatures when browser artifact is used.	One of the following: • Validate signatures for artifact. (Select check box.) • Do not validate signatures for artifact. (Clear check box.)

Table 72. Signature validation information for identity provider partner in SAML 1.x federation (continued)

Signature Validation	Description	Your value
Select Validation Truststore or Key • Truststore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the truststore • Public key to use for validating the signature of your partner	Because Browser POST messages must be signed and validated, you are required to specify a key to validate the signature. If you select to also validate messages when browser artifact is used, the same validation key is used to validate them. The key you use is the public key that corresponds to the private key that your partner uses to sign messages. **Note:** If you are importing the data of your partner, the key is supplied in the metadata file. You are asked to choose a keystore for the key. Be sure that you have created the keystore before this task. If you are manually entering the data of your partner, be sure that you have obtained the key from your partner. Then import the key into the appropriate keystore in the Tivoli Federated Identity Manager key service before this task. See Chapter 8, "Setting up message security," on page 49.	**Metadata method:** • Truststore name: • Truststore password: • Label for key: **Manual method:** • Truststore name: • Truststore password: • Key alias name:

Table 73. Server certificate validation for your identity provider partner in a SAML 1.x federation

Server Certificate Validation for SOAP	Description	Your value
Select Server Validation Certificate	The public key for the certificate that shows during SSL communication with your partner. You and your partner must agree on which certificate to use. You must have already obtained the certificate and keystore for the certificate. See "Retrieving the server certificate from your partner" on page 80.	Truststore name: Truststore password: Certificate name:

Table 74. Client authentication for SOAP for your identity provider partner in a SAML 1.x federation

Client Authentication for SOAP	Description	Your value
Client authentication information Either: • **Basic authentication** – Username – Password • **Client certificate authentication** – Certificate you must present to the server of the identity provider. The certificate that you and your identity provider partner agreed that you would present. – Keystore in Tivoli Federated Identity Manager key service, where the key is stored – Password for the keystore	If your partner requires mutual authentication, you must know which type you must use. If it is basic authentication, you need a user name and password. If it is client certificate authentication, you need the certificate that you and your partner have agreed to use. **Note:** If you need a certificate, be sure that you have agreed with your partner where to get it. Then, import it into the appropriate keystore in the Tivoli Federated Identity Manager key service before this task. See "Obtaining your client certificate" on page 81	One of the following: • Basic authentication information: – Username: – Password: • Client certificate authentication information: – Keystore name: – Password for the keystore: – Key alias:

Table 75. Security token settings information for identity provider partner in SAML 1.x federation

Configure Security Token	Description	Your value
Enable Signature Validation	If your partner signs assertions, you can choose to validate those signatures. In some cases, your partner require you to validate the signatures.	One of the following: • Enable validation signatures. (Select check box.) • Do not validate signatures. (Clear check box.)
Select Validation Key	Specify the type of signature validation to use.	One of the following: • Use the KeyInfo of the XML signature to find X.509 certificate for signature validation • Use keystore alias to find public key for signature validation. (The default action.) • Specify the Subject DN expression for the allowable X.509 certificates.

Table 75. Security token settings information for identity provider partner in SAML 1.x federation (continued)

Configure Security Token	Description	Your value
Select key and truststore • Truststore in Tivoli Federated Identity Manager key service, where the key is stored • Password for the truststore • Public key to use for validating the signature	If you choose to validate the assertion signatures or your partner requires signature validation, you must select a keystore and a key. **Note:** The key you use must be the public key that corresponds to the private key that your partner uses to sign the assertions. Obtain this key and create the keystore before this task. (space betChapter 8, "Setting up message security," on page 49.	• Truststore name: • Truststore password: • Key alias name:
Create multiple attribute statements in the Universal User	Select this check box to keep multiple attribute statements in the groups they were received in. This option might be necessary if your custom identity mapping rules are written to operate on one or more specific groups of attribute statements. If this check box is not selected, multiple attribute statements are arranged into a single group (AttributeList) in the STSUniversalUser document. The default setting of the check box is not selected and this setting is appropriate for most configurations.	

Table 76. Identity mapping information for identity provider partner in SAML 1.x federation

Identity mapping	Description	Your value
Identity mapping options One of the following: • A custom mapping module • An XSL transformation file containing mapping rules • Leave all options blank to use the identity mapping option that is currently defined.	The type of identity mapping to use with this partner. You can leave these fields blank, if you want this partner to use the identity mapping option that is already configured for your federation. Or, you can choose a specific mapping option to use with this specific partner. To choose a mapping option, you must know whether to use an XSLT file for identity mapping or a custom mapping module. Custom mapping is an advanced option. If you plan to use this option, first, your mapping module must be created and added to the environment as a module type and module instance. Then you can use it in your configuration. If you choose to use an XSLT file, you must have the file ready to use for the federation.	Leave all options blank to use existing mapping configuration. Or, select one of the following values: • XSLT file (path and name): • Custom mapping module instance name:

When you have completed this worksheet, continue with the steps in "Adding your partner" on page 245.

SAML 2.0 service provider partner worksheet

If use SAML 2.0 in your role as an identity provider, you must add a service provider partner to your federation.

Use the following worksheet to gather the necessary information from your partner. Modify this worksheet to reflect the specific information that you need from your partner and ask your partner to complete that modified worksheet.

Table 77. Federation to which you are adding a service provider partner in a SAML 2.0 federation

Select Federation	Description	Your value
Federation name	The name of the federation to which you are adding the partner.	

Table 78. Metadata file from your service provider partner in a SAML 2.0 federation

Import metadata	Description	Your value
Metadata file	The name and path of the file you obtained from your partner that contains the configuration information of your partner.	

Table 79. Signature validation for your service provider partner in a SAML 2.0 federation

Signature Validation	Description	Your value
Select which incoming SAML messages and assertions require a signature	These options indicate which incoming messages your partner signs. The default setting is for the typical set of incoming SAML messages and assertions (except for ArtifactResponse and AuthnResponse) to be signed.	One of the following options: • Typical set of incoming SAML messages and assertions are signed. • All incoming SAML messages and assertions are signed. • No incoming SAML messages and assertions are signed.
Keystore	The truststore in which you store the key that your partner has provided to validate its signature in messages when signing messages and assertions. You must have already created the keystore for this key. See "Preparing the keystores" on page 49 for details.	Truststore name: Truststore password: Key label:

Table 80. Keystore for storing the encryption key from your service provider partner in a SAML 2.0 federation

Encryption	Description	Your value
Keystore	The truststore in which you store the key to encrypt messages to your partner. This option shows because your partner has provided a public key in its metadata for you to use for encryption. You must have already created the keystore for this key. See "Preparing the keystores" on page 49 for details.	Truststore name: Truststore password: Key label:

Table 81. Server certificate validation for your service provider partner in a SAML 2.0 federation

SSL Server Authentication for Artifact Resolution	Description	Your value
Select Server Validation Certificate	The public key for the certificate that shows during SSL communication with your partner. You and your partner must agree which certificate to use. You must have already obtained the certificate and added it to your truststore. See "Retrieving the server certificate from your partner" on page 80 for details.	Truststore name: Truststore password: Certificate name:

Table 82. Client authentication for your service provider partner in a SAML 2.0 federation

SSL Client Authentication for Artifact Resolution	Description	Your value
Client authentication information Either: • **Basic authentication** – Username – Password • **Client certificate authentication** – Certificate to present to the server of the identity provider. This certificate is the certificate that you and your identity provider partner agreed to present. – Keystore in Tivoli Federated Identity Manager key service, where the key is stored – Password for the keystore	If your partner requires mutual authentication, you must know that which type to use. If it is basic authentication, you need a user name and password. If it is client certificate authentication, you need the certificate that you and your partner have agreed to use. **Note:** If you need a certificate, be sure that you have agreed with your partner where it comes from. Obtain and import it into the appropriate keystore in the Tivoli Federated Identity Manager key service before this task. See "Obtaining your client certificate" on page 81 for details.	One of the following options: • Basic authentication information: – Username: – Password: • Client certificate authentication information: – Keystore name: – Password for the keystore: – Key alias:

Table 83. Partner settings for your service provider partner in a SAML 2.0 federation

Partner Settings	Description	Your value
Session Timeout (seconds)	The number of seconds that a session remains valid when there is no activity. Default value: 3600 seconds.	Session timeout:

Table 83. Partner settings for your service provider partner in a SAML 2.0 federation (continued)

Partner Settings	Description	Your value
Logout Request Lifetime (seconds)	Specifies the maximum time, in seconds, that the logout request remains valid. The default value is 120 seconds.	Logout lifetime:
Signature Algorithm: Either: • DSA-SHA1 OR • RSA-SHA1 • RSA-SHA256	**Note:** The **Signature Algorithm** options are only available in the Tivoli Federated Identity Manager, version 6.2.2 or later. The type of signature algorithm to generate the XML signatures for your partner. The list of options shows all possible algorithms based on the key type you have chosen for your Federation. Only the algorithms that are supported by the runtime are shown. If you use a DSA key, DSA-SHA1 is selected by default. If you use an RSA key, RSA-SHA1 is selected by default. **Note:** The system shows the RSA-SHA256 option depending on the WebSphere Application Server fix pack version that runs in your system.	Choose one of the following options: • DSA-SHA1 • RSA-SHA1 • RSA-SHA256

Table 83. Partner settings for your service provider partner in a SAML 2.0 federation (continued)

Partner Settings	Description	Your value
Alias Service Settings • Include federation ID when performing alias service operations	**Note:** The **Alias Service Settings** option is only available in the Tivoli Federated Identity Manager, version 6.2.2 or later. Indicates whether the key for indexing into the alias service combines the federation ID with the partner Provider ID when performing alias service operations. This feature is useful in scenarios where two or more federations, that use persistent name identifiers, import the same partner metadata. **Note:** In the previous Tivoli Federated Identity Manager versions (before 6.2.2), aliases were stored based on a key using only the partner Provider ID. Use the migration steps to migrate aliases between different formats on a per-partner level. See the **Migrating SAML 2.0 alias service entries** section in the *IBM Tivoli Federated Identity Manager Installation Guide* for more information about the migration steps.	Select or clear the check box.

Table 84. SAML Assertion settings for your service provider partner in a SAML 2.0 federation

SAML Assertion Settings	Description	Your value
Include the following attribute types	Select the check box to specify the types of attributes to include in the assertion. The asterisk (*), is the default setting. It indicates that all of the attribute types that are specified in the identity mapping file, or by the custom mapping module are included in the assertion. To specify one or more attribute types individually, type each attribute type in the box. For example, if you want to include only attributes of the following type, enter that value in the box: `urn:oasis:names:tc:SAML:2.0:assertion` Use && to separate multiple attribute types.	
Encryption options: • Encrypt name identifiers • Encrypt assertions • Encrypt all assertion attributes	These check boxes indicate which assertion parts to encrypt. If you do not make a selection and leave the boxes blank, no assertion parts in your messages are encrypted.	Leave blank or choose one or more of the following options: • Encrypt name identifiers • Encrypt assertions • Encrypt all assertion attributes
Encryption algorithm: • AES-128 • AES-256 • AES-192 • Triple DES	The type of encryption algorithm to use for encrypting data to your partner. If you do not select an algorithm, Triple DES is used.	Choose one of the following options, if you chose an encryption option: • AES-128 • AES-256 • AES-192 • Triple DES

Table 85. Attribute query mapping information for your service provider partner

Attribute query mapping	Description	Your value
Attribute query mapping options One of the following options: • An XSL transformation file or JavaScript containing mapping rules • Tivoli Directory Integrator mapping module • A custom mapping module	The type of attribute query mapping you are using. You must select either an XSLT file, a Tivoli Directory Integrator mapping module, or a custom mapping module. If you use an XSLT file, you create the file before you configure the federation. The Tivoli Directory Integrator mapping module is an STS module. Custom mapping is an advanced option. If you use this option, you must create and add a new module type and module instance *before* you can use it in your configuration.	One of the following values: • XSLT file path • Tivoli Directory Integrator mapping module • Custom mapping module instance name

Table 86. Identity Mapping options for your service provider partner in a SAML 2.0 federation

Identity Mapping Options	Description	Your value
Identity mapping options One of the following: • An XSL transformation file containing mapping rules • A custom mapping module • Leave all options blank to use the identity mapping option that is currently defined.	The type of identity mapping to use with this partner. You can leave these fields blank, if you want this partner to use the identity mapping option that is already configured for your federation. Or, you can choose a specific mapping option to use with this specific partner. To choose a mapping option, you must know whether to use an XSLT file for identity mapping or a custom mapping module. Custom mapping is an advanced option. If you plan to use this option, create and add your mapping module to the environment as a module type and module instance. Do so *before* you can use it in your configuration. If you choose to use an XSLT file, you must have the file ready to use for the federation.	Leave all options blank to use existing mapping configuration. One of the following values: • XSLT file (path and name): • Custom mapping module instance name:

When you have completed this worksheet, continue with the steps in "Adding your partner" on page 245.

SAML 2.0 identity provider partner worksheet

If you use SAML 2.0 in your role as a service provider, you must add an identity provider partner to your federation.

Use the following worksheet to gather the necessary information from your partner. Modify this worksheet to reflect the specific information that you need from your partner and ask your partner to complete that modified worksheet.

Table 87. Federation to which you are adding an identity provider partner in a SAML 2.0 federation

Select Federation	Description	Your value
Federation name	The name of the federation to which you are adding the partner.	

Table 88. Metadata file from your identity provider partner in a SAML 2.0 federation

Import metadata	Description	Your value
Metadata file	The name and path of the file you obtained from your partner that has their configuration information.	

Table 89. Signature validation for your identity provider partner in a SAML 2.0 federation

Signature Validation	Description	Your value
Select which incoming SAML messages and assertions require a signature	These options indicate which incoming messages your partner signs. The default setting is for the typical set of incoming SAML messages and assertions (except for ArtifactResponse and AuthnResponse) to be signed.	One of the following options: • Typical set of incoming SAML messages and assertions are signed. • All incoming SAML messages and assertions are signed. • No incoming SAML messages and assertions are signed.
Keystore	The truststore where you store the key from your partner to validate its signature when signing messages and assertions. You must have already created the keystore for this key. See "Preparing the keystores" on page 49 for details.	Truststore name: Truststore password: Key label:

Table 90. Keystore for storing the encryption key from your identity provider partner in a SAML 2.0 federation

Encryption	Description	Your value
Keystore	The truststore where you store the key to encrypt messages to your partner. The system shows this option because your partner has provided a public key in its metadata for you to use for encryption. You must have already obtained the certificate and imported it into a keystore. See Chapter 8, "Setting up message security," on page 49 for details.	Truststore name: Truststore password: Key label:

Table 91. Server certificate validation for your identity provider partner in a SAML 2.0 federation

SSL Server Authentication for Artifact Resolution	Description	Your value
Select Server Validation Certificate	The public key for the certificate that shows during SSL communication with your partner. You and your partner must agree on which certificate to use. You must have already obtained the certificate and added it to your truststore. See "Retrieving the server certificate from your partner" on page 80 for details.	Truststore name: Truststore password: Certificate name:

Table 92. Client authentication for your identity provider partner in a SAML 2.0 federation

SSL Client Authentication for Artifact Resolution	Description	Your value
Client authentication information Either: • **Basic authentication** – Username – Password • **Client certificate authentication** – Certificate to present to the identity provider server. This certificate is what you and your identity provider partner agreed to present. – Keystore in Tivoli Federated Identity Manager key service, where the key is stored – Password for the keystore	If your partner requires mutual authentication, you must know which type to use. If it is basic authentication, you need a user name and password. If it is client certificate authentication, you need the certificate that you and your partner have agreed to use. **Note:** If you need a certificate, be sure that you have agreed with your partner where it comes from. Obtain and import it into the appropriate keystore in the Tivoli Federated Identity Manager key service before this task. See "Obtaining your client certificate" on page 81 for details.	One of the following options: • Basic authentication information: – Username: – Password: • Client certificate authentication information: – Keystore name: – Password for the keystore: – Key alias:

Table 93. Partner settings for your identity provider partner in a SAML 2.0 federation

Partner Settings	Description	Your value
Default Post-Authentication Target URL	The location to which the user is redirected when the service provider does not provide a target URL during the initial request. This URL must be valid but does not have to be active.	

Table 93. Partner settings for your identity provider partner in a SAML 2.0 federation (continued)

Partner Settings	Description	Your value
Force authentication to achieve account linkage	Specifies if a user is forced to authenticate at the service provider to perform account linkage. This event occurs if a SAML response is received with an unknown alias in the service provider.	
Signature Algorithm: Either: • DSA-SHA1 OR • RSA-SHA1 • RSA-SHA256	**Note:** The **Signature Algorithm** options are only available in the Tivoli Federated Identity Manager, version 6.2.2 or later. The type of signature algorithm to generate the XML signatures for your partner. The list of options shows all possible algorithms based on the key type you have chosen for your Federation. Only the algorithms that are supported by the runtime are shown. If you use a DSA key, DSA-SHA1 is selected by default. If you use an RSA key, RSA-SHA1 is selected by default. **Note:** The system shows the RSA-SHA256 option depending on the WebSphere Application Server fix pack version that runs in your system.	Choose one of the following options: • DSA-SHA1 • RSA-SHA1 • RSA-SHA256

Table 93. Partner settings for your identity provider partner in a SAML 2.0 federation (continued)

Partner Settings	Description	Your value
Alias Service Settings • Include federation ID when performing alias service operations	**Note:** The **Alias Service Settings** option is only available in the Tivoli Federated Identity Manager, version 6.2.2 or later. Indicates whether the key for indexing into the alias service combines the federation ID with the partner Provider ID when performing alias service operations. This feature is useful in scenarios where two or more federations, that use persistent name identifiers, import the same partner metadata. **Note:** In versions of Tivoli Federated Identity Manager before 6.2.2, aliases were stored based on a key using only the partner Provider ID. Use the migration steps to migrate aliases between different formats on a per-partner level. For more information about the migration steps, see the 'Migrating SAML 2.0 alias service entries' topic in the *IBM Tivoli Federated Identity Manager Installation Guide*.	Select or clear the check box.

Table 94. SAML Assertion settings for your identity provider partner in a SAML 2.0 federation

SAML Assertion Settings	Description	Your value
Username to be used for anonymous users	Use this name identifier 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 gives users access to a resource on the service provider without establishing a federated identity. This feature is useful in scenarios where the service provider does not need to know the identity of the user account but must only know that the identity provider has authenticated (and can vouch for) the user.	
Map unknown name identifiers to the anonymous username	Specifies that the service provider can map an unknown persistent name identifier alias to the anonymous user account. By default, this option is disabled.	
Create multiple attribute statements in the universal user	Select this check box to keep multiple attribute statements in the groups they were received in. This option might be necessary if your custom identity mapping rules are written to operate on one or more specific groups of attribute statements. If this check box is not selected, multiple attribute statements are arranged into a single group (AttributeList) in the STSUniversalUser document and in the assertion. The default setting of the check box is not selected and this setting is appropriate for most configurations.	

Table 94. SAML Assertion settings for your identity provider partner in a SAML 2.0 federation (continued)

SAML Assertion Settings	Description	Your value
Encryption options: • Encrypt name identifiers	The check box indicates whether to encrypt the name identifiers in assertions or not.	Select or clear the check box.

Table 95. Attribute query information for identity provider partner

Attribute query	Description	Your value
Include the following attribute types	Select the check box to specify the types of attributes to include in the assertion. The asterisk (*), is the default setting. It indicates that all of the attribute types that are specified in the identity mapping file, or by the custom mapping module, are included in the assertion. To specify one or more attribute types individually, type each attribute type in the box. For example, if you want to include only attributes of the following type, enter that value in the box: `urn:oasis:names:tc:SAML:2.0:assertion` Use && to separate multiple attribute types.	
Encryption options: • Encrypt name identifiers • Encrypt assertions • Encrypt all assertion attributes	These check boxes indicate which assertion parts to encrypt. If you do not make a selection and leave the boxes blank, no assertion parts in your messages are encrypted. **Encrypt all assetion attributes** indicates whether all attributes in the assertion are encrypted. When this option is not selected (set to false), you can manage the encryption of specific attributes through an XSLT SAML token mapping rule.	Leave blank or choose one or more of the following options: • Encrypt name identifiers • Encrypt assertions • Encrypt all assertion attributes

Table 95. Attribute query information for identity provider partner (continued)

Attribute query	Description	Your value
Encryption algorithm: • AES-128 • AES-256 • AES-192 • Triple DES	The type of encryption algorithm to use for encrypting data to your partner. If you do not select an algorithm, Triple DES is used.	Choose one of the following encryption, if you chose an encryption option: • AES-128 • AES-256 • AES-192 • Triple DES

Table 96. Attribute query mapping information for identity provider partner

Attribute query mapping	Description	Your value
Attribute query mapping options One of the following options: • An XSL transformation file or JavaScript containing mapping rules • Tivoli Directory Integrator mapping module • A custom mapping module	The type of attribute query mapping you are using. You must select either an XSLT file, a Tivoli Directory Integrator mapping module, or a custom mapping module. If you use an XSLT file, you create the file before you configure the federation. The Tivoli Directory Integrator mapping module is an STS module. Custom mapping is an advanced option. If you use this option, you must create and add a new module type and module instance *before* you can use it in your configuration.	One of the following values: • XSLT file (path and name) • Tivoli Directory Integrator mapping module • Custom mapping module instance name

Table 97. Identity Mapping options for your identity provider partner in a SAML 2.0 federation

Identity Mapping Options	Description	Your value
Identity mapping options One of the following options: • An XSL transformation file containing mapping rules. • A custom mapping module. • Leave all options blank to use the identity mapping option that is currently defined.	The type of identity mapping to use with this partner. You can leave these fields blank, if you want this partner to use the identity mapping option that is already configured for your federation. Or, you can choose a specific mapping option to use with this specific partner. To choose a mapping option, you must know whether to use an XSLT file for identity mapping or a custom mapping module. Custom mapping is an advanced option. If you plan to use the custom mapping option, create and add the mapping module to the environment as a module type and module instance. Only then can you use it in your configuration. If you choose to use an XSLT file, you must have the file ready to use for the federation.	Leave all options blank to use existing mapping configuration. One of the following values: • XSLT file (path and name): • Custom mapping module instance name:

When you have completed this worksheet, continue with the steps in "Adding your partner."

Adding your partner

After you have configured your role in the federation and gathered information about your partner, you will need to add your partner.

Before you begin

Before beginning this procedure, complete the appropriate partner information worksheet.

- If you are the identity provider, you will add a service provider partner. Use the service provider worksheet for the SAML standard you are using in your federation:
  - "SAML 1.x service provider partner worksheet" on page 219
  - "SAML 2.0 service provider partner worksheet" on page 230
- If you are the service provider, you will add an identity provider partner. Use the identity provider worksheet for the SAML standard you are using in your federation:

– "SAML 1.x identity provider partner worksheet" on page 224
– "SAML 2.0 identity provider partner worksheet" on page 237

About this task

After completing the appropriate partner worksheet, use the Partner wizard in the console to add the partner. For descriptions of the fields that you are prompted for by the wizard, refer to the worksheet and the online help.

Note: During the configuration, you might be prompted to restart WebSphere Application Server. Make sure the server has restarted completely before continuing with the task.

Procedure

1. Make sure you have gathered the partner information as described in the worksheets.

   For example, if you are using a metadata file from your partner, copy the file to an easily accessible location on your computer.
2. Log on to the console.
3. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations**. The Federations panel opens.
4. Select the federation to which you will add the partner.
5. Click **Add partner**. Depending on the SAML standard you are using in the federation, one of the following panels open:

   Metadata Options
   This panel opens if you are adding a partner to a SAML 1.x federation. From this panel, click either:
   - **Import Metadata**
   - **Enter SAML data manually**

   Import Metadata
   This panel opens if you are adding a partner to a SAML 2.0 federation.
6. Use your completed worksheet as a guide for completing the fields that show in each panel.

   If you need to go back to a previous panel, click **Back**. If you want to end the configuration, click **Cancel**. Otherwise, click **Next** after you complete each panel.
7. Verify that the settings are correct.
8. Click **Finish**. The Add Partner Complete panel opens. The partner has been added to the federation, but is disabled by default as a security precaution. You must enable the partner.
9. Click **Enable Partner** to activate this partner.

What to do next

If you have not already provided your configuration information to your partner, you can do that now using the instructions in "Providing federation properties to your partner" on page 247.

Providing federation properties to your partner

When your partner wants to add you as a partner to their federation configuration, you must provide your partner with the necessary information.

The steps to take are specific to whether you can provide a metadata file or whether you must manually provide the information.

- **Metadata file method**

  If your partner has a way to import your data, you can use the metadata file method whether you have configured a SAML 1.x or SAML 2.0 federation.

  1. Use the console to generate a metadata file that contains the necessary federation configuration and a key for validating response message signatures, if you require validation of the signatures. Follow the instructions in "Exporting federation properties."
  2. You might also need to provide your partner with the appropriate keys and certificates for your role and SAML standard in the federation. See Chapter 8, "Setting up message security," on page 49.

- **Manual method**

  You have the option of manually collecting the necessary configuration instead of exporting the properties to a file, if you configured a SAML 1.x federation.

  Note: Use of a metadata file is preferable because it eliminates the chance of errors being made during the manual input of data.

  If you need to collect information manually, complete the following task.

  1. Use the Federation Properties in the console to obtain the properties. To display the Federation Properties panel for your federation, follow the instructions in "Viewing federation properties" on page 248.

     Use the contents of the Federation Properties panel to guide you to the properties that apply to your federation.
  2. You might also need to provide your partner with appropriate keys and certificates for your federation. See Chapter 8, "Setting up message security," on page 49.

Exporting federation properties

When you want to join the federation of a 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`

Place the file in an easily accessible location. Provide this file to your partner, when your partner wants to import configuration information for the federation.

7. Click **Save**.

Viewing federation properties

Use the Federations properties selection to view the details about an existing federation or to modify an existing federation. This task can be helpful if you need to manually collect your federation properties to share them with your partner.

Procedure

1. Log 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 properties for an existing federation.
5. Select the properties to modify. Federation properties are described in the online help.
6. When finished viewing or modifying properties, click **OK** to close the Federation Properties panel.

Synchronizing system clocks in the federation

Because security tokens have expiration times, you and your partner's system clocks must be synchronized.

About this task

In your environment, ensure that the clock on the system where you have the Tivoli Federated Identity Manager runtime and management services component installed is synchronized with your partner.

See the information of your operating system documentation for information about your system clock and time synchronization. Consider using the NTP time synchronization protocol.

Chapter 20. Configuring a SAML federation using CLI

To configure a SAML federation using CLI, you must configure the Identity Provider federation, configure the Service Provider federation, and provide the federation configuration properties to your partner.

The following topics detail configurations for specific SAML versions and roles using CLI

Configuring a SAML 1.x Identity Provider federation using CLI

Use CLI commands to configure a SAML Identity Provider federation by creating a response file and creating an Identity Provider federation.

About this task

This task requires the use of the command **manageItfimFederation**. The **manageItfimFederation** command requires specific parameters to execute operations on a federation. For more information, see the *IBM Tivoli Federated Identity Manager Administration Guide*.

Procedure

1. Create a response file by issuing the following command in the WebSphere **wsadmin** console:

   ```
   wsadmin>$AdminTask manageItfimFederation { -operation createResponseFile
   -fimDomainName fimipdomain -role ip -protocol SAML1_1 -fileId
   /downloads/saml11_ip_properties.xml }
   ```

   Note: Change the SAML protocol type depending on the SAML version you intend to use. Use one of the following parameters for the protocol type:

   - SAML1_1
   - SAML1_0

2. Edit the response file to modify the following values:

Table 98. Response file settings for identity provider in SAML 1.x federation

Configuration item	Description	Your value	CLI Properties or Names
Federation name	The unique name of the federation. (Required)	Any name For example, `saml11ip`	**FedName**
Company name	The name of the company that is associated with the federation. (Required)	Any name For example, `IDP Company Name`	**CompanyName**
Company URL	A URL for a website of the company that is associated with the federation. (Required)	URL of the website your company.	**CompanyUrl**
Provider ID	The SAML protocol provider ID used by the federation. A unique identifier that identifies the provider to its partner provider. (Required)	The value consists of the protocol and host name of the Identity Provider URL (Optional). It can include a port number. For example, for a federation named`saml_fed: https://idp.example.com/FIM/sps/saml_fed/saml`, set all the properties on the next column to the same value.	**ProviderID** **SAML11AssertionIssuerName**
Point of contact server URL	The URL that provides access to the endpoints on the point of contact server. (Required)	A URL For example, `https://www.idpexample.com/FIM/sps`	**BaseUrl**
SAML messages for Browser POST profile are signed	When the browser POST is used as the profile, SAML messages must be signed. Therefore, it is preselected and cannot be cleared. (Required)	Sign browser artifact messages. (Set to true.)	**SignArtifactResponse**

Table 98. Response file settings for identity provider in SAML 1.x federation (continued)

Configuration item	Description	Your value	CLI Properties or Names

Select Signing Key
 	
Keystore in Tivoli Federated Identity Manager key service, where the key is stored.
	Enter a signing key because the Browser POST messages must be signed.
	
	If you also select to sign messages when using the browser artifact, the same signing key is used to sign them. (Required)
	
	Note: Before you complete this task, create the key and import it into the appropriate keystore in the Tivoli Federated Identity Manager key service.
		Keystore name:
		
		Key alias name:
		
		This data is provided in the format of
		`"Keystore Name" _"Alias Name"`
		
		For example,
		`DefaultKeyStore_ testkey`
			`SigningKeyId`

Single Sign-on Service URL	The URL for your single sign-on endpoint. This setting is also known as the intersite transfer service URL, or the URL to which the Service Provider sends authentication requests. (Required)	Specify the assertion resolution service url. For example, for a federation named `saml_fed`: `https://idp.example.com/ FIM/sps/saml_fed/saml11/ login`	**`SignonEndpoint`**
Artifact Resolution Service URL	The URL for your artifact resolution endpoint. (Required)	Specify the assertion resolution service url For example, for a federation named `saml_fed`: `https://idp.example.com/ FIM/sps/saml_fed/saml11/ soap`	**`ArtifactResolutionServiceEndpoint`**
Artifact Cache Lifetime (seconds)	The artifact cache lifetime in seconds. Default value: 30 seconds.	Use the default value.	**`ArtifactCacheLifetime`**

Table 98. Response file settings for identity provider in SAML 1.x federation (continued)

Configuration item	Description	Your value	CLI Properties or Names
Allow IBM Protocol Extension	Specify whether to use the IBM PROTOCOL extension. The extension contains a query-string parameter that specifies whether to use browser artifact or browser POST.(Required)	Do not allow Protocol Extension. Set to false.	**AllowIBMProtocolExtension**
Amount of time before the issue date that an assertion is considered valid	The number of seconds that an assertion is considered valid before its issue date. Default value: 60	Use the default value.	**SAML11AssertionValidBefore**
Amount of time the assertion is valid after being issued	The number of seconds that an assertion is considered valid after its issue date. Default value: 60	Use the default value.	**SAML11AssertionValidAfter**
Identity mapping options • An XSL transformation (XSLT) file containing mapping rules	The type of identity mapping to use. Use an XSLT file for identity mapping, and have the file ready to use for the federation.(Required)	XSLT File that corresponds to the IP role for SAML 1.1 federations: `/opt/IBM/FIM/examples/mapping_rules/ip_saml_1x.xsl`	**MappingRuleFileName**

3. Type the following command in a command prompt to create the Identity Provider federation:

```
wsadmin>$AdminTask manageItfimFederation { -operation create
-fimDomainName fimipdomain -fileId
/downloads/saml11_ip_properties.xml }
```

The following confirmation message shows:

```
FBTADM001I Command completed successfully
```

What to do next

Continue with Configuring a SAML 1.x Service Provider federation using CLI.

Configuring a SAML 1.x Service Provider federation using CLI

Use CLI commands to configure a SAML 1.x Service Provider federation by creating a response file and creating a Service Provider federation.

About this task

This task requires the use of the command **manageItfimFederation**. The **manageItfimFederation** command requires specific parameters to execute operations on a federation. For more information, see the *IBM Tivoli Federated Identity Manager Administration Guide*.

Procedure

1. Create a response file by issuing the following command in the WebSphere **wsadmin** console:

   ```
   wsadmin>$AdminTask manageItfimFederation { -operation createResponseFile
   -fimDomainName fimspdomain -protocol SAML1_1 -role sp -fileId
   /downloads/saml11_sp_properties.xml }
   ```

   The following confirmation message shows:

   ```
   FBTADM001I Command completed successfully
   ```

   Note: Change the SAML protocol type depending on the SAML version you intend to use. Use one of the following parameters for the protocol type:

   - SAML1_1
   - SAML1_0

2. Edit the response file to modify the following values:

Table 99. Response file settings for service provider in SAML federation

Configuration item	Description	Your value	CLI Properties or Names
Federation name	The unique name of the federation. (Required)	Any name For example, `saml11ip`	**FedName**
Company name	The name of the company that is associated with the federation. (Required)	Any name For example, `SP Company Name`	**CompanyName**
Company URL	A URL for a website of the company that is associated with the federation. (Required)	URL of the website of your company	**CompanyUrl**
Protocol	The SAML protocol that you and your partner use in the federation. (Required)	One of the following values: • `SAML1_1` • `SAML1_0`	**Protocol**

Table 99. Response file settings for service provider in SAML federation (continued)

Configuration item	Description	Your value	CLI Properties or Names
ProviderId	The SAML protocol provider ID used by the federation. A unique identifier that identifies the provider to its partner provider. (Required)	The value consists of the protocol and host name of the Service Provider URL. (Optional) It can include a port number. For example, for a federation named `saml_fed`: `https://sp.example.com/FIM/sps/saml_fed/saml`, set all the properties on the next column to the same value.	**ProviderId**
Point of contact server URL	The URL that provides access to the endpoints on the point of contact server. (Required)	A URL For example, `https:/sp.example.com/FIM/sps`	**BaseUrl**
Sign Artifact Resolution Requests	The SAML artifact request must be signed.	Sign request messages. (Set to true.)	**SignArtifactRequest**
Select Signing Key Keystore in Tivoli Federated Identity Manager key service, where the key is stored.	If you also selected to sign the artifact request, enter a signing key. (Required) **Note:** Before you complete this task, create the key and import it into the appropriate keystore in the Tivoli Federated Identity Manager key service.	Keystore name: Key alias name: This data is provided in the format of `"Keystore Name"_"Alias Name"`. For example, `DefaultKeyStore_ testkey`	**SigningKeyId**
Single Sign-on Service URL	The URL for your single sign-on endpoint. This setting is also known as the intersite transfer service URL, or the URL to which the Service Provider sends authentication requests. (Required)	Specify the assertion resolution service URL For example: `https://sp.example.com/FIM/sps/saml_fed/saml11/login`	**SignonEndpoint**
Identity mapping options • An XSL transformation (XSLT) file containing mapping rules	The type of identity mapping to use. Use an XSLT file for identity mapping, and have the file ready to use for the federation. (Required)	XSLT File that corresponds to the IP role for SAML 1.1 federations: `/opt/IBM/FIM/examples/mapping_rules/sp_saml_1x.xsl`	**MappingRuleFileName**

3. Type the following command in a command prompt to create the Service Provider federation:

```
wsadmin>$AdminTask manageItfimFederation { -operation create
-fimDomainName fimspdomain -fileId
/downloads/saml11_sp_properties.xml }
```

The following confirmation message shows:

```
FBTADM001I Command completed successfully
```

What to do next

Continue with Importing a SAML 1.X Service Provider into the SAML Identity Provider federation.

Importing a SAML 1.x Service Provider into the SAML identity provider federation

To add a Service Provider to the Identity Provider federation, you must import the Service Provider configuration properties.

Procedure

1. Type the following command in a command prompt to export the Service Provider metadata and obtain most of the environmental information:

   ```
   wsadmin>$AdminTask manageItfimFederation { -operation export
   -fimDomainName fimspdomain -federationName saml11sp -fileId
   /downloads/saml11_sp_metadata.xml }
   ```

   The following confirmation message shows:

   ```
   FBTADM001I Command completed successfully
   ```

2. Create a Service Provider response file by issuing the following command in the WebSphere **wsadmin** console:

   ```
   wsadmin>$AdminTask manageItfimPartner { -operation createResponseFile
   -fimDomainName fimipdomain -federationName saml11ip -partnerRole sp -fileId
   /downloads/saml11_sp_partner_properties.xml }
   ```

   The following confirmation message shows:

   ```
   FBTADM001I Command completed successfully
   ```

3. Edit the response file to modify the following values:

Table 100. Response file settings for service provider partner in SAML 1.x federation

Configuration item	Description	Your value	CLI Properties or Names
Import metadata file	To import a metadata file, you need the file name and its location. (Required)	The fully specified metadata file name. For example: `/downloads/saml11_sp_metadata.xml`	**metadataFileName**
Validate Signatures on Artifact Requests	Validate the SAML message signatures when you use the browser artifact. (Optional)	Validate signatures for artifact. (Set to true.)	**ValidateArtifactRequest**

Table 100. Response file settings for service provider partner in SAML 1.x federation (continued)

Configuration item	Description	Your value	CLI Properties or Names
Validation Key Identifier	If you select to validate messages when the browser artifact is used, you must provide a key for the validation. The key must be the public key that corresponds to the private key that your partner uses to sign the messages. (Required) **Note:** If you are importing partner data, the key is supplied in the metadata file. Before importing the data, create a keystore, then specify a keystore for the key. Before entering partner data manually, obtain the key from the partner and import it into the appropriate keystore in the Tivoli Federated Identity Manager key service.	**Metadata method:** • Truststore name: • Label for key:	**ValidateKeyIdentifier**
Sign SAML Assertions	Sign SAML assertions. (Optional)	Enable SAML signatures. (Set to true.)	**com.tivoli.am.fim.sts.saml.1.1.assertion.sign**

Select Signing Key
• Keystore in Tivoli Federated Identity Manager key service, where the key is stored
	If you choose to sign the assertion signatures, you must select a keystore and a key. (Required)
	
	Note: Create the keystore and key before this task.
		• Keystore name:
		• Key alias name:
		
		This data is provided in the format of
		`"Keystore Name"_ "Alias Name"`
		
		For example:
		`DefaultKeyStore_ testkey`
		
		Set both properties to the same value.
			SigningKeyId,
			SAML11SigningKeyIdentifier

Table 100. Response file settings for service provider partner in SAML 1.x federation (continued)

Configuration item	Description	Your value	CLI Properties or Names
Include the following attribute types	Select the check box to specify the types of attributes to include in the assertion. The default setting (marked with an asterisk), indicates that all attribute types specified in the identity mapping file included in the assertion. (Required)	Use the default value (*)	**SAML11ExtendedAttributeTypes**
Partner uses Browser POST profile for Single Sign-On	A Boolean that indicates that the Service Provider partner uses Browser POST. (Required)	Partner uses Browser POST (Set to true.)	**PartnerUsesBrowserPost**

4. Type the following command in a command prompt to add the new Service Provider partner to the Identity Provider federation:

```
wsadmin>$AdminTask manageItfimPartner { -operation create
-fimDomainName fimipdomain
-federationName saml11ip -partnerName saml11sp -fileId
/downloads/saml11_sp_partner_properties.xml
-signingKeystorePwd testonly}
```

   The following confirmation message shows:

```
FBTADM001I Command completed successfully
```

What to do next

Continue with Importing a SAML 1.x Identity Provider into the SAML Service Provider federation.

Importing a SAML 1.x Identity Provider into the SAML Service Provider federation

To add an Identity Provider to the Service Provider federation, you must import the Identity Provider configuration properties.

Procedure

1. Type the following command in a command prompt to export the Identity Provider metadata and obtain most of the environmental information:

```
wsadmin>$AdminTask manageItfimFederation { -operation export
-fimDomainName fimipdomain -federationName saml11ip -fileId
/downloads/saml11_ip_metadata.xml }
```

   The following confirmation message shows:

```
FBTADM001I Command completed successfully
```

2. Create an Identity Provider response file by issuing the following command in the WebSphere **wsadmin** console:

```
wsadmin>$AdminTask manageItfimPartner { -operation createResponseFile
-fimDomainName fimspdomain -federationName saml11sp -partnerRole ip -fileId
/downloads/saml11_ip_partner_properties.xml }
```

The following confirmation message shows:

```
FBTADM001I Command completed successfully
```

3. Edit the response file to modify the following values:

Table 101. Response file settings for Identity Provider partner in SAML 1.x federation

Configuration item	Description	Your value	CLI Properties or Names
Import metadata file	To import a metadata file, you need the file name and its location. (Required)	The fully specified metadata file name. For example: `/downloads/saml11_ip_metadata.xml`	**metadataFileName**
Validate Signatures on Artifact Resolution Requests	You have the option of validating the SAML message signatures when browser artifact is used.	Validate signatures for artifact. (Set to true.)	**ValidateArtifactResponse**
Validation Key Identifier	Because Browser POST messages must be signed and validated, you must specify a key to validate the signature. If also you select to validate messages when using a browser artifact, use the same validation key to validate them. The key you use is the public key that corresponds to the private key that your partner uses to sign messages. **Note:** If you are importing partner data, the key is supplied in the metadata file. Before importing the data, create a keystore, then specify a keystore for the key. Before entering partner data manually, obtain the key from the partner and import it into the appropriate keystore in the Tivoli Federated Identity Manager key service.	**Metadata method:** • Truststore name: • Label for key:	**ValidateKeyIdentifier**
Server Certificate Validation	Enable server certificate validation	Set to true.	**UseSoapServerCertAuth**

Table 101. Response file settings for Identity Provider partner in SAML 1.x federation (continued)

Configuration item	Description	Your value	CLI Properties or Names
Select Server Validation Certificate	The public key for the displayed certificate during SSL communication with your partner. Determine the certificate used by you and your partner. You must already have the certificate and keystore for the certificate.	• Truststore password: • Certificate name: 30 access	**rtAuthKeyId**
Client authentication information Either: • Basic authentication – Username – Password • Client certificate authentication – Certificate to present to the server of the Identity Provider. The specified certificate is determined by you and your Identity Provider partner. – Keystore in Tivoli Federated Identity Manager key service, where the key is stored – Password for the keystore	If your partner requires mutual authentication, determine which type to use. • For basic authentication, specify a user name and password. • For client certificate authentication, specify the certificate that you and your partner agreed to use. **Note:** Before performing this task, be sure that you and your partner agreed where the certificate to be obtained, and imported it into the keystore in the Tivoli Federated Identity Manager key service.	Disable client authentication by setting the properties on the next column to false.	**UseClientBasicAuth** **UseSoapClientCertAuth**
Validate SAML Assertions Signature	Validate the SAML assertions signature. (Optional)	Enable SAML signature validation. (Set to true.)	**com.tivoli.am.fim .sts.saml.1.1 .assertion.verify .signatures**
Select Validation Key for Assertion signature	Specify the assertion signature validation key to use.	Use keystore alias to find public key for signature validation. (Default)	**SAML11ValidationKey**

Table 101. Response file settings for Identity Provider partner in SAML 1.x federation (continued)

Configuration item	Description	Your value	CLI Properties or Names
Create multiple attribute statements in the Universal User	Select this option to keep multiple attribute statements in the groups they were received in. This option might be necessary if your custom identity mapping rules are written to operate on one or more specific groups of attribute statements. If this option is not selected, multiple attribute statements are arranged into a single group (**AttributeList**) in the **STSUniversalUser** document.	Set the value to false.	**SAML11Create** **MultipleUniversal** **UserAttributes**

4. Type the following command in a command prompt to add the new Service Provider partner to the Identity Provider federation:

```
wsadmin>$AdminTask manageItfimPartner { -operation create -fimDomainName
fimspdomain -federationName saml11sp -partnerName saml11ip -fileId
/downloads/saml11_ip_partner_properties.xml -signingKeystorePwd testonly}
```

   The following confirmation message shows:

```
FBTADM001I Command completed successfully
```

Configuring a SAML 2.0 Identity Provider federation using CLI

Use CLI commands to configure a SAML Identity Provider federation by creating a response file and creating an Identity Provider federation.

About this task

This task requires the use of the command **manageItfimFederation**. The **manageItfimFederation** command requires specific parameters to execute operations on a federation. For more information, see the *IBM Tivoli Federated Identity Manager Administration Guide*.

Procedure

1. Create a response file by issuing the following command in the WebSphere **wsadmin** console:

```
wsadmin>$AdminTask manageItfimFederation { -operation createResponseFile
-fimDomainName fimipdomain -role ip -protocol SAML2_0 -fileId
/downloads/saml20_ip_properties.xml }
```

2. Edit the response file to modify the following values:

Table 102. Response file settings for Identity Provider in SAML 2.0 federation

Configuration item	Description	Your value	CLI Properties or Names
Federation name	The unique name of the federation. (Required)	Any name For example, `saml20ip`	**FedName**

Table 102. Response file settings for Identity Provider in SAML 2.0 federation (continued)

Configuration item	Description	Your value	CLI Properties or Names
Company name	The name of the company that is associated with the federation. (Required)	Any name For example, `IDP Company Name`	**`CompanyName`**
Company URL	A URL for a website of the company that is associated with the federation. (Required)	URL of the website of your company	**`CompanyUrl`**
Point of Contact Server	The URL of the point of contact server with the federation name and the protocol name, such as /saml20, appended to it. (Required)	A URL For example, for a federation named `saml_fed: https://idp.example.com/FIM/sps/saml_fed/saml20`	**`BaseUrl`**
Provider ID	A URL or URN that uniquely identifies the provider. By default Tivoli Federated Identity Manager uses the URL of the point of contact server with the federation name and the protocol name, such as /saml20, appended to it.	A URL For example, for a federation named saml_fed:`https://idp.example.com/FIM/sps/saml_fed/saml20`)	**`ProviderId`**

Select Signing Key
 	
Keystore in Tivoli Federated Identity Manager key service, where the key is stored.
	Enter a signing key for the Identity Provider.
	
	The protocol mandates that a SAML Response that contains the assertion is signed when using the HTTP POST binding.
	
	If you also select to sign any other messages the specified signing key is used to sign them. (Required)
		Keystore name:
		
		Key alias name:
		
		This data is provided in the format of
		`"Keystore Name" _"Alias Name"`
		
		For example,
		`DefaultKeyStore_ testkey`
			`SigningKeyIdentifier`

Table 102. Response file settings for Identity Provider in SAML 2.0 federation (continued)

Configuration item	Description	Your value	CLI Properties or Names

Select Encryption Key
 	
Keystore in Tivoli Federated Identity Manager key service, where the key is stored.
	A public/private key pair used in encryption.
	
	Your partner uses the public key to encrypt data to you.
	
	Use the private key to decrypt data that your partner sends to you.
	
	You must specify the key pair to use.
	
	Note: Before you complete this task, create the key and import it into the appropriate keystore in the Tivoli Federated Identity Manager key service.(Required)
		Keystore name:
		
		Key alias name:
		
		This data is provided in the format of
		`"Keystore Name" _"Alias Name"`
		
		For example,
		`DefaultKeyStore_ testkey`
			EncryptionKeyIdentifier

Single Sign-on	SAML 2.0 supports single sign-on using different profiles, use this setting to enable them accordingly.	True or false. Default: false. You must enable at least one property. For example, set `SsoPostEnabled` to true.	**SsoPostEnabled** **SsoArtifactEnabled** **SsoRedirectEnabled**
Single Logout	To enable single logout, set at least one to true and they can choose which binding and provider that can be used to initiate single logout.	True or false. Default: false. You must enable at least one property to enable the single logout profile for the federation. For example, set `SloIPPostEnabled` to true.	**SloIPArtifactEnabled** **SloIPPostEnabled** **SloIPRedirectEnabled** **SloIPSOAPEnabled** **SloSPArtifactEnabled** **SloSPPostEnabled** **SloSPRedirectEnabled** **SloSPSOAPEnabled**

Table 102. Response file settings for Identity Provider in SAML 2.0 federation (continued)

Configuration item	Description	Your value	CLI Properties or Names
Artifact Resolution Service URL	The Artifact Resolution Service is a SOAP endpoint on the Identity Provider point-of-contact server where artifacts are exchanged for SAML messages. By default, Tivoli Federated Identity Manager configures one SOAP endpoint for the Artifact Resolution Service. You can optionally define additional SOAP endpoints.	Specify the assertion resolution service URL, the URL index. Set to true if you use the endpoint use as the default. Set to false otherwise. For example, `https://idp.example.com/FIM/sps/saml_fed/saml20/soap;0;true`	**ArtifactResolutionServiceList**
Artifact Cache Lifetime (seconds)	The artifact cache lifetime in seconds. Default value: 120 seconds.	Use the default value.	**ArtifactLifetime**
Amount of time before the issue date that an assertion is considered valid	The number of seconds that an assertion is considered valid before its issue date. Default value: 60	Use the default value	**AssertionValidBefore**
Amount of time the assertion is valid after being issued	The number of seconds that an assertion is considered valid after its issue date. Default value: 60	Use the default value	**AssertionValidAfter**
Identity mapping options An XSL transformation (XSLT) file containing mapping rules	The type of identity mapping to use. Use an XSLT file for identity mapping, and have the file ready to use for the federation. (Required)	XSLT File that corresponds to the IP role for SAML 2.0 federations: `/opt/IBM/FIM/examples/mapping_rules/ip_saml_20_email_nameid.xsl`	**MappingRuleFileName**

3. Type the following command in a command prompt to create the Identity Provider federation:

```
wsadmin>$AdminTask manageItfimFederation { -operation create
-fimDomainName fimipdomain -fileId
/downloads/saml20_ip_properties.xml }
```

The following confirmation message shows:

```
FBTADM001I Command completed successfully
```

What to do next

Continue with Configuring a SAML 2.0 Service Provider federation using CLI.

Configuring a SAML 2.0 service provider federation using CLI

Use CLI commands to configure a SAML 2.0 service provider federation by creating a response file and creating a service provider federation.

About this task

This task requires the use of the command **manageItfimFederation**. The **manageItfimFederation** command requires specific parameters to execute operations on a federation. For more information, see the *IBM Tivoli Federated Identity Manager Administration Guide*.

Procedure

1. Create a response file by issuing the following command in the WebSphere **wsadmin** console:

   ```
   wsadmin>$AdminTask manageItfimFederation { -operation createResponseFile
   -fimDomainName fimspdomain -protocol SAML2_0 -role sp -fileId
   /downloads/saml20_sp_properties.xml }
   ```

   The following confirmation message shows:

   ```
   FBTADM001I Command completed successfully
   ```

2. Edit the response file to modify the following values:

Table 103. Response file settings for service provider in SAML 2.0 federation

Configuration item	Description	Your value	CLI Properties or Names
Federation name	The unique name of the federation. (Required)	Any name For example, `saml20sp`	**FedName**
Company name	The name of the company that is associated with the federation. (Required)	Any name For example, `SP Company Name`	**CompanyName**
Company URL	A URL for a website of the company that is associated with the federation. (Required)	URL of the website of your company	**CompanyUrl**
Provider ID	A URL or URN that uniquely identifies the provider. By default Tivoli Federated Identity Manager uses the URL of the point of contact server with the federation name and the protocol name, such as /saml20, appended to it.	URL For example, for a federation named saml_fed: `https://sp.example.com/FIM/sps/saml_fed/saml20`	**ProviderId**
Point of Contact Server URL	The URL of the point of contact server with the federation name and the protocol name, such as /saml20, appended to it. (Required)	A URL For example, for a federation named `saml_fed: https://sp.example.com/FIM/sps/saml_fed/saml20`	**BaseUrl**

Table 103. Response file settings for service provider in SAML 2.0 federation (continued)

Configuration item	Description	Your value	CLI Properties or Names

Select Signing Key
 	
Keystore in Tivoli Federated Identity Manager key service, where the key is stored.
	Enter a signing key for the service provider. If you also select to sign any other messages the specified signing key is used to sign them. (Required)
	
	Note: Before you complete this task, create the key and import it into the appropriate keystore in the Tivoli Federated Identity Manager key service.
		Keystore name:
		
		Key alias name:
		
		This data is provided in the format of
		`"Keystore Name" _"Alias Name"`
		
		For example,
		`DefaultKeyStore_ testkey`
			SigningKeyIdentifier

Single sign-on	The URL to which the Service Provider sends authentication requests.	True or false. Default: false. You must enable at least one property. For example, set `SsoPostEnabled` to true.	**SsoPostEnabled** **SsoArtifactEnabled** **SsoRedirectEnabled**

Select Encryption Key
 	
Keystore in Tivoli Federated Identity Manager key service, where the key is stored.
	A public/private key pair used in encryption. Your partner uses the public key to encrypt data to you.
	
	Use the private key to decrypt data that your partner sends to you.
	
	You must specify the key pair to use.
	
	Note: Before you complete this task, create the key and import it into the appropriate keystore in the Tivoli Federated Identity Manager key service.(Required)
		Keystore name:
		
		Key alias name:
		
		This data is provided in the format of
		`"Keystore Name" _"Alias Name"`
		
		For example,
		`DefaultKeyStore_ testkey`
			EncryptionKeyIdentifier

Single Logout Profile	The URL that the partner contacts to use the Single Logout profile. To enable single logout, set at least one property to true. Then, you can choose which binding and provider to use to initiate single logout.	True or false. Default: false. You must enable at least one property to enable the single logout profile for the federation. For example, set `SloSPPostEnabled` to true.	**SloIPArtifactEnabled** **SloIPPostEnabled** **SloIPRedirectEnabled** **SloIPSOAPEnabled** **SloSPArtifactEnabled** **SloSPPostEnabled** **SloSPRedirectEnabled** **SloSPSOAPEnabled**

Table 103. Response file settings for service provider in SAML 2.0 federation (continued)

Configuration item	Description	Your value	CLI Properties or Names
Artifact Resolution Service list	The Artifact Resolution Service is a SOAP endpoint on the service provider point of contact server where artifacts are exchanged for SAML messages. By default, Tivoli Federated Identity Manager configures one SOAP endpoint for the Artifact Resolution Service. You can optionally define additional SOAP endpoints.	Specify the assertion resolution service URL, the URL index, and set to true if the endpoint is used as the default. Otherwise, set to false. For example, `https://sp.example.com/FIM/sps/saml_fed/saml20/soap;0;true`	**ArtifactResolutionServiceList**
Identity mapping options An XSL transformation (XSLT) file containing mapping rules	The type of identity mapping to use. Use an XSLT file for identity mapping, and have the file ready to use for the federation. (Required)	XSLT File that corresponds to the SP role for SAML 2.0 federations: `/opt/IBM/FIM/examples/mapping_rules/sp_saml_20.xsl`	**MappingRuleFileName**

3. Type the following command in a command prompt to create the Service Provider federation:

```
wsadmin>$AdminTask manageItfimFederation { -operation create
-fimDomainName fimspdomain -fileId
/downloads/saml20_sp_properties.xml }
```

   The following confirmation message shows:

```
FBTADM001I Command completed successfully
```

What to do next

Continue with Importing a SAML 2.0 service provider into the SAML identity provider federation.

Importing a SAML 2.0 Service Provider into the SAML Identity Provider federation

To add a Service Provider to the Identity Provider federation, you must import the Service Provider configuration properties.

Procedure

1. Type the following command in a command prompt to export the Service Provider metadata and obtain most of the environmental information:

```
wsadmin>$AdminTask manageItfimFederation { -operation export
-fimDomainName fimspdomain -federationName saml20sp -fileId
/downloads/saml20_sp_metadata.xml }
```

   The following confirmation message shows:

```
FBTADM001I Command completed successfully
```

2. Create a Service Provider response file by issuing the following command in the WebSphere **wsadmin** console:

```
wsadmin>$AdminTask manageItfimPartner { -operation createResponseFile
-fimDomainName fimipdomain -federationName saml2ip -partnerRole sp -fileId
/downloads/saml20_sp_partner_properties.xml }
```

   The following confirmation message shows:

```
FBTADM001I Command completed successfully
```

3. Edit the response file to modify the following values:

Table 104. Response file settings for Service Provider partner in SAML 2.0 federation

Configuration item	Description	Your value	CLI Properties or Names
Import metadata file	To import a metadata file, you need the file name and its location. (Required)	The fully specified metadata file name. For example: `/downloads/saml20_sp_metadata.xml`	**metadataFileName**
Signature Validation Options	The partner metadata contains the key to use for signature validation. Specify the keystore name and alias name where Tivoli Federated Identity Manager stores the key included on the metadata.	`DefaultTrustedKeyStore`	**signatureKeystoreName**
	The alias name under which the key is stored on the specified keystore.	`spsignkey`	**signatureKeyAlias**
Encryption Options	The partner metadata contains the key to use for encryption. Specify the keystore name and alias name where Tivoli Federated Identity Manager stores the key included on the metadata.	`DefaultTrustedKeyStore`	**encryptionKeystore**
	The alias name under which the key is stored on the specified keystore.	`spenckey`	**encryptionKeyAlias**
Assertion Attribute Types	Specify the attribute types to be added to the assertion generated by the Identity Provider.	* (Default)	**AssertionAttributeTypes**

Table 104. Response file settings for Service Provider partner in SAML 2.0 federation (continued)

Configuration item	Description	Your value	CLI Properties or Names
Partner Server SSL Certificate Validation	The Identity Provider makes a direct connection to the Service Provider for some SAML bindings. Specify the key to use to validate the server ssl certificate. **Note:** Before you complete this task, import the key into the appropriate keystore in the Tivoli Federated Identity Manager key service.	`DefaultTrustedKeystore_spsslcert`	**`ServerCertKeyId`**

4. Type the following command in a command prompt to add the new Service Provider partner to the Identity Provider federation:

```
wsadmin>$AdminTask manageItfimPartner { -operation create
-fimDomainName fimipdomain
-federationName saml20ip -partnerName saml20sp -fileId
/downloads/saml20_sp_partner_properties.xml
-signingKeystorePwd testonly}
-encryptionKeystorePwd testonly }
```

   The following confirmation message shows:

```
FBTADM001I Command completed successfully
```

What to do next

Continue with Importing a SAML 2.0 Identity Provider into the SAML Service Provider federation.

Importing a SAML 2.0 Identity Provider into the SAML service provider federation

To add an Identity Provider to the Service Provider federation, you must import the Identity Provider configuration properties.

Procedure

1. Type the following command in a command prompt to export the Identity Provider metadata and obtain most of the environmental information:

```
wsadmin>$AdminTask manageItfimFederation { -operation export
-fimDomainName fimipdomain -federationName saml20ip -fileId
/downloads/saml20_ip_metadata.xml }
```

   The following confirmation message shows:

```
FBTADM001I Command completed successfully
```

2. Create an Identity Provider response file by issuing the following command in the WebSphere **`wsadmin`** console:

```
wsadmin>$AdminTask manageItfimPartner { -operation createResponseFile
-fimDomainName fimspdomain -federationName saml20sp -partnerRole ip -fileId
/downloads/saml20_ip_partner_properties.xml }
```

The following confirmation message shows:

```
FBTADM001I Command completed successfully
```

3. Edit the response file to modify the following values:

Table 105. Response file settings for Identity Provider partner in SAML 2.0 federation

Configuration item	Description	Your value	CLI Properties or Names
Import metadata file	To import a metadata file, you need the file name and its location. (Required)	The fully specified metadata file name. For example: `/downloads/saml20_ip_metadata.xml`	**metadataFileName**
Signature Validation Options	The partner metadata contains the key to use for signature validation. Specify the keystore name and alias name where Tivoli Federated Identity Manager stores the key included on the metadata.	`DefaultTrustedKeyStore`	**signatureKeystoreName**
	The alias name under which the key is stored on the specified keystore.	`ipsignkey`	**signatureKeyAlias**
Encryption Options	The partner metadata contains the key to use for encryption. Specify the keystore name and alias name where Tivoli Federated Identity Manager stores the key included on the metadata.	`DefaultTrustedKeyStore`	**encryptionKeystore**
	The alias name under which the key is stored on the specified keystore.	`ipenckey`	**encryptionKeyAlias**

Table 105. Response file settings for Identity Provider partner in SAML 2.0 federation (continued)

Configuration item	Description	Your value	CLI Properties or Names
Partner Server SSL Certificate Validation	The Identity Provider makes a direct connection to the Service Provider for some SAML bindings. Specify the key to use to validate the server ssl certificate. **Note:** Before you complete this task, import the key into the appropriate keystore in the Tivoli Federated Identity Manager key service.	`DefaultTrustedKeystore_ipsslcert`	**`ServerCertKeyId`**
Default Target URL	Default target URL where the browser is sent to, when a successful single sign on happens at the Service Provider. This location is only used if a Target URL was not specified on the request.	`https://saml20clisp:444/FIM/fimivt/protected/ivtlanding.jsp`	**`DefaultPostAuthTargetURL`**
Default Single Sign On User	Default user name for single sign on at the Service Provider side.	`guest`	**`AnonymousUserUserName`**

4. Type the following command in a command prompt to add the new Service Provider partner to the Identity Provider federation:

```
wsadmin>$AdminTask manageItfimPartner { -operation create -fimDomainName
fimspdomain -federationName saml20sp -partnerName saml20ip -fileId
/downloads/saml20_ip_partner_properties.xml -signingKeystorePwd testonly}
-encryptionKeystorePwd testonly }
```

   The following confirmation message shows:

```
FBTADM001I Command completed successfully
```

Chapter 21. Planning an Information Card federation

This planning guide reviews the Tivoli Federated Identity Manager implementation of the Information Card standard, and describes how to plan the configuration process. This guide does not provide a comprehensive review of the Information Card standard.

You can use the Information Card system to manage your digital identities from various identity providers. Then, you can and use these digital identities to access various services that accept these digital identities.

Administrators who are not familiar with the standard must review the Information Card documentation in the Microsoft website.

The Tivoli Federated Identity Manager support for Information Card includes deployment of Tivoli Federated Identity Manager in both of the Information Card roles: Managed Identity Provider and Relying Party.

The protocol flow when the user provides an information card to authenticate at a website, resembles the forms-based login flow. However, it requires extra steps.

1. User directs the browser to a protected web page that requires authentication.
2. The site redirects the browser to a login page. In an Information Card-enabled browser, the login page contains an HTML tag that allows the user choose an Information Card to authenticate to the site. When the user selects the tag, the browser starts the *identity selector*.

   Note: An *identity selector* is a browser plug-in that enables the browser to use the Information Card protocol. The plug-ins are sometimes called *identity agents*.
3. The browser support code for Information Cards starts the identity selector. Then, the browser passes to the identity selector the parameter values supplied by the Information Card HTML tag obtained from the website in Step 2.

   The user then selects an Information Card, which represents a digital identity that can be used to authenticate at the site.
4. The identity selector sends the information card to the Tivoli Federated Identity Manager identity provider. The identity provider uses the Tivoli Federated Identity Manager security token service to process the WS-Trust message and WS-Metadata Exchange. Then, it generates a token that contains the user credentials. The identity provider returns the token to the browser.

   Note: IBM deprecated the Tivoli Federated Identity Manager Security Token Service (STS) Client in this release.

   If you use WebSphere 6.X, you can still use the Tivoli Federated Identity Manager Security STS client while Tivoli Federated Identity Manager supports WebSphere 6.X. When Tivoli Federated Identity Manager discontinues its support for WebSphere 6.X, use WebSphere Application Server version 7 Update 11 and later. See WS-Trust client API and WS-Trust Clients for details.
5. The browser forwards the user credentials to the website that protects the requested resource. The site validates the credentials and redirects the browser back to the requested page.

In the protocol flow, the Relying Party and the identity provider do not communicate directly with each other. By default, neither party is aware of the other. The Relying Party does not know which identity provider was selected by the user until the token is received in Step 5. At that time, the Relying Party can learn the identity by examining the Issuer field in the token.

You can use the Information Card to prompt the identity provider to require identification from the Relying Party. However, doing so is not a requirement, and is typically discouraged.

Overview of the Information Card identity provider

Tivoli Federated Identity Manager, when operating as an identity provider, supports the issuing of managed cards, and issues security tokens for managed cards.

The identity provider supports:

- Issuing managed cards

  The issuing of managed cards is triggered when a user authenticates to a Tivoli Federated Identity Manager identity provider and accesses a card download URL. The URL sends the user a template HTML form, requesting user information that is required in order to issue the card. When the user supplies the necessary information, Tivoli Federated Identity Manager issues the card and sends it to the browser of the user. The user can save this card for future use.
- Retrieval of security tokens for managed cards

  This support is provided through the Security Token Service (STS). This component supports two types of SOAP messages from an Information Card identity selector. The SOAP messages are required for an identity selector to obtain a security token for the managed information card of the user.

  Note: An *identity selector* is a browser plug-in. It is sometimes called an *identity agent*.

  Only SAML 1.1 security tokens are supported.

The support includes the following features:

- Issuing of managed cards
- Endpoints for metadata exchange, and processing of WS-Trust messages
- Support for Information Card claims
- A unique federation to contain the identity provider endpoints
- A trust service chain to convert user identity information into a SAML 1.1 token

Note: Information Card federations do not maintain configuration settings as metadata. There is no metadata to export or import between identity providers and relying parties for Information Card deployments.

Issuing of managed cards

Tivoli Federated Identity Manager provides support for identity providers to issue managed cards, and to retrieve security tokens from managed cards that have been issued by other authorities.

Tivoli Federated Identity Manager provides a protected endpoint that permits the downloading of a managed card. When a user, through a browser, accesses the

endpoint, an HTML template file is loaded and returned the user. The user is prompted to supply information that is required in order to issue the managed card.

The required information is:

- User name

  The user name is an arbitrary value that the user assigns to the card.
- The set of claims that the card supports.

  A claim is a Uniform Resource Indicator (URI) that represents qualified attribute names. Tivoli Federated Identity Manager uses the list of claims to determine which information to place into the security token that is generated at runtime, when the managed card is processed. Examples of the information placed in the security token are each claim, and its corresponding value.
- When the federation uses an authentication method called *self-issued credential* or *self-signed SAML assertion*, part of the request is to prompt the user to post a token generated by a self-issued card.

  When the federation uses an authentication method called *username token*, the user does not need to provide this parameter.

Tivoli Federated Identity Manager provides two template HTML pages.

- When the authentication method is username token, the template getcard_ut.html is used.
- When the authentication method is self-issued credential, the template getcard_sss.html is used.

Administrators can modify the template HTML files to best suit the local deployment.

The getcard_* template files contain the following macros, which are replaced with values specific to the request from the user.

@FORMACTION@
: This macro is replaced with the required form action URL, to which the HTML form is posted.

@USERNAME@
: This macro is replaced with the user name, as supplied by either the login name for the Tivoli Access Manager user, or by an authenticated WebSphere user. The Tivoli Access Manager user name is used when WebSEAL is the point of contact server. The WebSphere user name is used when WebSphere is the point of contact server.

  This value can be used pre-populate the card name parameter in the template.

When the user posts the form back to Tivoli Federated Identity Manager, the information is placed into macros in an XML template file called infocard_template.xml. This template file represents the managed card that is returned to the user through the browser.

In most deployments, system administrators do not need to modify the macros in infocard_template.xml. However, the file provides a number of macros that can be modified if needed.

Note: To view a list of macros, see "Replacement macros in the infocard_template XML file" on page 301.

Tivoli Federated Identity Manager support for Information Card includes the SAML 1.1 token type only. The SAML 1.1 token type has two representations:

SAML 1.1
urn:oasis:names:tc:SAML:1.0:assertion

SAML 1.1
http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1

Most managed information cards support both representations. The user does not select the token type. The @SUPPORTED_TOKENS@ macro in infocard_template.xml is defined in the two preceding SAML representations.

Tivoli Federated Identity Manager supports two methods for the identity selector to authenticate the user to the identity provider security token service. Each method supports a different replacement template for the @USERCRED@ macro in the information card template (infocard_template.xml).

The type of authentication is specified by the administrator when the federation is configured. The configuration values for the `authenticationMethod` parameter map to template files as follows:

UsernameToken
Maps to the template file **infocard_usercred_usernametoken.xml**

The template file has one replacement macro:

@USERNAME@
This macro is replaced with the user name. The user name is supplied either by the login name for the Tivoli Access Manager user or an authenticated WebSphere user. The Tivoli Access Manager user name is used when WebSEAL is the point of contact server. The WebSphere user name is used when WebSphere is the point of contact server.

SelfSignedSAML
Maps to the template file **infocard_usercred_selfsignedsaml.xml**.

The template file has one replacement macro:

@PPID@
This macro is replaced with the PPID of the self-issued card that is posted as part of the getcard_sss.html form. This process occurs when the federation uses the SelfSignedSAML authentication method.

Tivoli Federated Identity Manager stores this value as an alias for the current user in the Tivoli Federated Identity Manager alias service. The alias is used to map the self-issued card back to the Tivoli Federated Identity Manager user.

This process occurs when the self-issued card is used at runtime to generate a SAML assertion and authenticate the identity provider security token service.

Identity provider federations

The configuration of Information Card federations differs from the configuration of other single sign-on protocol federations, such as SAML 2.0, Liberty, WS-Federation, or OpenID. The main difference is that the Information Card identity provider does not have to know the security token recipient. The identity provider security token service interacts only with the identity selector. This eliminates the need to configure any properties that contain information about partners.

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

The concept of partner configuration exists in Information Card configurations only as part of the configuration of token modules used by the trust service.

The key properties that define a federation for an identity provider are:

ProtocolID

Tivoli Federated Identity Manager uses a ProtocolID as a unique identifier. The Information Card federation has the succeeding protocolId syntax:

`https://<hostname:port>/FIM/sps/<federation_name>/infocard`

For example:

`https://www.exampleidentitydemo.com/FIM/sps/csip/infocard`

Endpoint for obtaining a managed card

An endpoint for processing HTML interaction with an authenticated user, in order to build and download a managed card.

The URL for the endpoint is based on the ProtocolID. For example:

`https://www.exampleidentitydemo.com/FIM/sps/csip/infocard/getcard.crd`

The Tivoli Federated Identity Manager component (single sign-on protocol service delegate) for the endpoint completes the succeeding tasks:

1. Prompts the user for information required to generate an information card. The information card information includes the card name and the supported claims. When the authentication method is self-issued credential, a personal information card is generated.
2. When the authentication mechanism is self-issued card, also called SelfSignedSAML, the delegate creates and store an alias in the alias service. The alias maps the personal card presented by the user during this process to the user account of the person currently authenticated in the browser session.
3. Generates the managed card from an XML template, with various pieces populated dynamically. The delegate signs the card with the private key of the SSL certificate associated with the point-of-contact server. Then, the delegate sends the card back to the browser.

Endpoint for exchanging metadata

The identity selector uses an endpoint at runtime to exchange metadata. Using an endpoint at runtime to exchange metadata determines the identity provider security token service RST connection and message formatting requirements.

The URL for the metadata endpoint is based on the ProtocolID. For example:

`https://www.exampleidentitydemo.com/FIM/sps/csip/infocard/mex`

The metadata exchange endpoint has a template XML file called metadata_template.xml. This file has some macros available for replacement.

Note: Administrators can use the default macros. You do not have to modify the macros in order to use the template file.

The replacement macros for metadata_template.xml are:

@IPSTS@
The URL of the identity provider security token service endpoint for the federation.

@IPPOLICY@
This value consists of WS-Policy information. The information is dependent upon the type of authentication token that is used to authenticate to the identity provider security token service. The WS-Policy information is read from a template file.

@IPCERTIFICATE@
The base-64 encoded public SSL certificate for the point of contact server.

Each of the supported authentication methods supports a different replacement template for the @IPPOLICY@ macro in the metadata exchange template.

The template files for each authentication method are:

UsernameToken authentication
metadata_policy_usernametoken.xml

SelfSignedSAML authentication
metadata_policy_selfsignedsaml.xml

The metadata_policy_usernametoken.xml and metadata_policy_selfsignedsaml.xml have no replacement macros. The template files consist of different sets of policy, as appropriate for each method. Information Card administrators do not have to modify these files.

An endpoint for receiving WS-Trust messages
The identity provider security token service has an endpoint that receives WS-Trust messages from the identity selector. The Information Card identity provider module processes the incoming request, modifies the Tivoli Federated Identity Manager trust service, and communicates with the trust service to get the token.

Information Card claims

Information Card uses information called *claims* to define attributes that can be required in order to fulfill a user request. An information card contains the Uniform Resource Indicators (URIs) for the set of claims that are supported by its issuer.

An identity selector can use the claims information to determine if an identity card can be useful for signing in to a specific relying party. For example, when a relying party requires the e-mail address claim, and the identity provider associated with a given managed card does not support that claim, the identity provider does not offer the managed card as an option for signing in to that relying party.

The Tivoli Federated Identity Provider managed card provider places no restrictions on the set of claims that can be specified in cards. The templates (getcard_ut.html and getcard_sss.html) contain the full set of the standard supported claims. Administrators can add support for additional claims by modifying the templates.

The Information Card identity agent sends a WS-Trust request to the Tivoli Federated Identity Manager module (delegate) for the single sign-on protocol service. The WS-Trust request contains a claims element (wst:Claims) that contains the set of requested claims.

Figure 17 shows some example claims.

```
<wst:Claims>
 <wsid:ClaimType
  Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"
  xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" />
 <wsid:ClaimType
  Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"
  xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" />
 <wsid:ClaimType
  Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"
  xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" />
 <wsid:ClaimType
  Uri="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/
       privatepersonalidentifier"
  xmlns:wsid="http://schemas.xmlsoap.org/ws/2005/05/identity" />
</wst:Claims>
```

To view the standard set of claims supported by Microsoft, see: http://schemas.xmlsoap.org/ws/2005/05/identity/claims.xsd.

Figure 17. Example claims from a Information Card identity agent

Information Card error pages

The following Page Identifiers are provided:

/infocard/error_get_card.html

Maps to the following page:

`/infocard/error_get_card.html`

Used to show an error in HTML when a user is attempting to download a card.

/infocard/error_get_metadata.html

Maps to the following page:

`/infocard/error_get_metadata.html`

Used to show an error in HTML when an identity selector user is attempting to download metadata using HTTP GET (rather than SOAP over HTTP/POST).

Overview of the Information Card relying party

The role of Relying Party is similar to the role of a *service provider*, as supported by Tivoli Federated Identity Manager for other single sign-on protocols. The Relying Party consists of a login service implemented a single sign-on protocol service component (delegate) and a WS-Trust chain.

The Tivoli Federated Identity Manager implementation supports:

- Reception of SAML 1.x assertion tokens
- The use of login with both self-issued cards and managed cards issued by other identity providers.

In the Information Card model, the Secure Socket Layer (SSL) public key is used to encrypt the token that is sent to the endpoints for the Relying Party. The SSL key is the key of the SSL session established between the browser and the site presenting the Web page (as specified in the embedded OBJECT tags). This means that Tivoli Federated Identity Manager needs access to the SSL keys used by Point of Contact server.

The administrator must configure access to these keys during Tivoli Federated Identity Manager Information Card configuration.

Web sites must use X509v3 certificates with logotypes (also known as Extended Validation certificates) instead of SSL server certificates when providing identification of the enterprise.

The Information Card term **Relying Party** refers to a role that is similar to the **Service Provider** role in other single sign-on protocols that are supported by Tivoli Federated Identity Manager.

As a Relying Party, Tivoli Federated Identity Manager supports both Managed and Self-Issuing Identity Providers.

Tivoli Federated Identity Manager configuration enables administrators to configure support for one or both types of providers.

Prior to completing the Tivoli Federated Identity Manager configuration steps, the relying party administrator can obtain public keys from the identity provider, for use when validating digital signatures on assertions received from that provider.

The Tivoli Federated Identity Manager implementation includes support for:

- User access to the relying party
- Information Card claims
- Federations for processing requests
- Token exchange

User access to a relying party

When a user attempts to access a protected resource at a Web site, and the user has not previously established credentials, a *Point of Contact* Web server typically prompts the user to establish credentials by filling out a login page. The use of Information Card in this scenario is dependent on the prior establishment of the following:

- The user must be using a browser that has been enabled for Information Card. Browsers that support for Information Card have an *identity selector* plug-in installed.
- The login page from the Point of Contact that is protecting the resources at the Web site must be tagged with specific OBJECT tags. The OBJECT tags in the page trigger the browser to start the Information Card interaction.
- The URL that the browser accesses must use the HTTPS protocol.

```
<form method="post" action="/FIM/sps/infocard-fed/infocard/login">
   ...
   <input type="hidden" name="TARGET" value="/TheResource"/>
   <object type="application/x-informationCard" name="xmlToken">
      <param name="requiredClaims"
             value="http://schemas.xmlsoap.org/ws/2005/05/identity/
                    claims/privatepersonalidentifier" />
   </object>
   <input type="submit" value="Login"/>
   ...
</form>
```

Figure 18. Example login format for use by Relying Party

Figure 18 shows sample XML elements in the required login format. The login format requires several important parameters:

Form method action

The value of the `action` parameter must be the URL of the Information Card federation endpoint. The Information Card enabled browser redirects to this endpoint to process the security token received from the Identity Provider.

Note: The administrator specifies this endpoint when configuring Tivoli Federated Identity Manager Information Card.

Input type hidden name

The login form should have a `hidden` element with:

- The `name` parameter set to `TARGET`
- The `value` set to the URL to which the browser is redirected when the login process completes.

There is an alternative way to specify the URL to which the browser is to be redirected. The target can be specified using a query string parameter on the value of the `action` parameter. For example, using the values from Figure 18:

```
action=''FIM/sps/infocard-fed/infocard/login?TARGET=/theResource''
```

When WebSEAL is the Point of Contact server, the `%URL%` macro supported by WebSEAL can be used to specify the target URL.

Object type name

The value of the `name` parameter on the `OBJECT` element must be `xmlToken`.

The browser sends this value to the Relying Party. The Tivoli Federated Identity Manager implementation for Information Card Relying Party uses this parameter to access the security token.

Tivoli Federated Identity Manager as a Relying Party supports the following SAML token types:

- URI supported for all provider types:
  `urn:oasis:names:tc:SAML:1.0:assertion`
- URI supported for self-issuing identity providers only:
  `http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1`

One or more of these type URIs can be specified in the `tokenType` parameter of the `OBJECT` tag.

Relying party federations

Tivoli Federated Identity Manager establishes and uses federations for Information Card in a manner that is similar to, but not identical to, the federations used for other single sign-on protocols. The differences are:

- The Relying Party interaction is part of the authentication process used by the Point of Contact server (for example, WebSEAL) to grants access to protected resources.

  For other single sign-on protocols, the Point of Contact server presents a login page when a protected resource is accessed, and then authenticates the user and produces credentials for the user. For Information Card, Tivoli Federated Identity Manager as a relying party performs the authentication process and produces the credentials for the user.

  The Relying Party becomes aware that a user sign-on is in progress when a security token (assertion) is received at its message endpoint. The Relying Party must then decide whether to accept or reject the security token.
- Unlike a service provider for other single sign-on protocols, the Relying Party does not send messages to the identity provider. The messages are sent by the *Identity Selector*, without the knowledge of the Relying Party
- In a Information Card federation, the identity providers are a set of loosely-federated entities from which the Web site accepts assertion tokens.
- Information Card supports the Self-Issuing Identity provider.

Information Card requires creation of a federation to represent the Relying Party *self*. The term *self* should not to be confused with Self-Issuing Identity provider. The term is used to distinguish the federation originator (creator) from any partners that are subsequently added to the federation. The self entity properties include:

- The login endpoint
- Parameters that indicate the types of tokens that are accepted
- The keystore alias for the private key from the Point of Contact server, for use in Secure Socket Layer (SSL) connections.
- A default mapping rule. The mapping rule can be overridden by a partner's configuration.

The Information Card federation uses the standard Tivoli Federated Identity Manager naming convention for **protocolID**. The syntax is:

`https://<hostname:port>/FIM/sps/<federation name>/infocard`

For example, when the host for the federation endpoints is `rp.example.com`, listening on port `443`, with a federation named `MyInfoCard-rp`, the protocolID is:

`https://rp.example.com:443/FIM/sps/MyInfoCard-rp/infocard`

Partner federations are needed to represent identity providers. There can be only one self-issuing token partner. There can be any number of Managed Identity provider partners. An **any** Identity Provider partner may also be added. This partner can be used for guest account access.

Self-Issuing partner

Tivoli Federated Identity Manager configures a partner with the protocolId set to:

```
http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self
```

This partner is used to process Self-issued cards.

Named Managed Identity Provider Partner

A managed provider partner must have a unique `Issuer URI`. The `Issuer` field of the trust chain mapping is set to the `protocolID` value. When assertions from this provider are signed, a public key alias must be configured for the partner.

The administrator must import the public key into a Tivoli Federated Identity Manager keystore before configuring the federation. The Tivoli Federated Identity Manager key service should be used to import the key.

Any Provider Partner

The `Any` provider allows the configuration of a wildcard Assertions from these providers must use *one* of the following values for`<saml:SubjectConfirmationMethod>` :

```
urn:oasis:names:tc:SAML:1.0:cm:bearer
urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
```

When the assertion is signed, the assertion must include a `<ds:KeyInfo>` element in the signature containing a public key that is to be used for validating the signatures.

Note: This configuration should only be used for guest user access. In this configuration, all users are mapped to a guest account.

Website enablement for Information Card

The Tivoli Federated Identity Manager implementation of the Information Card profile interoperates with the Microsoft CardSpace™ Version 1.0 implementation. Both implementations are based on Information Card Profile Version 1.0. This version is supported in Microsoft Internet Explorer Version 7.

Browsers that support Information Card must:

- recognize special HTML or XHTML tags for starting the Identity Selector,
- pass encoded parameters on to the Identity Selector on the platform, and
- post back the token resulting from the authentication type selected by the user for choice of a digital identity.

Websites that employ Information Card-based authentication must support two pieces of functionality:

- Addition of HTML or XHTML tags to their login page to request an Information Card-based login
- Code to log the user in to the site, using the credentials supplied by the user in the HTTP POST operation

In response to the Information Card-based login, the website typically responds by:

- Writing the same client-side browser cookie as it would when logins occur based on username-password authentication (or other mechanisms)
- Issuing the same browser redirects

Changes to login pages

HTML extensions such as the OBJECT tag are used to signal to the browser when to start the Identity Selector. However, not all HTML extensions are supported by all browsers.

Also, some commonly supported HTML extensions are disabled in browser high security configurations. For example, the OBJECT tag is disabled by high security settings on some browsers, including Internet Explorer.

An alternative to the use of HTML extensions is the use of an XHTML syntax that is not disabled by changing browser security settings. However, not all browsers provide full support for XHTML.

To provide a solution that addresses the range of scenarios, there are two HTML extension formats. Browsers might support one or both extension formats.

OBJECT syntax

Figure 19 shows an example of a page that uses the OBJECT syntax to request that the user login using an Information Card.

```
<html>
<head>
<title>Welcome to Fabrikam</title>
</head>
<body>
<img src='fabrikam.jpg'/>
<form name="ctl00" id="ctl00" method="post"
action="https://www.fabrikam.com/InfoCard-Browser/Main.aspx">
<center>
<img src='infocard.bmp' onClick='ctl00.submit()'/>
<input type="submit" name="InfoCardSignin" value="Log in"
id="InfoCardSignin" />
</center>
<OBJECT type="application/x-informationCard" name="xmlToken">
<PARAM Name="tokenType"
Value="urn:oasis:names:tc:SAML:1.0:assertion">
<PARAM Name="issuer" Value=
"http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self">
<PARAM Name="requiredClaims" Value=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname">
</OBJECT>
</form>
</body>
</html>
```

Figure 19. OBJECT syntax example

Notice the OBJECT of type `application/x-informationCard`. When the user selects a card, the resulting security token is included in the response (POST) as the

xmlToken value of the form. Parameters of the Information Card OBJECT are used to encode the required WSSecurityPolicy information in HTML.

In this example, the relying party is requesting a SAML 1.0 token from a self-issued identity provider, supplying the required claims emailaddress, givenname, and surname.

Note: You can omit the Issuer to indicate that *any* issuer of an Information Card available in the browser for the user is acceptable.

XHTML syntax

The XHTML syntax is as follows:

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:ic>
<head>
<title>Welcome to Fabrikam</title>
</head>
<body>
<img src='fabrikam.jpg'/>
<form name="ctl00" id="ctl00" method="post"
action="https://www.fabrikam.com/InfoCard-Browser/Main.aspx">
<ic:informationCard name='xmlToken'
style='behavior:url(#default#informationCard)'
issuer="http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self"
tokenType="urn:oasis:names:tc:SAML:1.0:assertion">
<ic:add claimType=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"
optional="false" />
<ic:add claimType=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"
optional="false" />
<ic:add claimType=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname"
optional="false" />
</ic:informationCard>
<center>
<input type="submit" name="InfoCardSignin" value="Log in"
id="InfoCardSignin" />
</center>
</form>
</body>
</html>
```

Figure 20. Example of InfoCard XHTML syntax

Identity selector invocation parameters

The parameters to the OBJECT and XHTML Information Card objects are used to encode information in HTML. In cases where an Identity Selector is used in a Web services context, this information would be supplied as WS-SecurityPolicy information through use of WSMetadataExchange.

The following list shows parameters supported by the Information Card standard for Identity Selector invocation.

Note: All parameters are optional. None of them are required.

issuer This parameter specifies the URL of the security token service (STS) from which to obtain a token. When omitted, no specific STS is requested. The

special value `http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self` specifies that the token comes from a self-issued identity provider.

Note: This parameter is not supported by Tivoli Federated Identity Manager.

issuerPolicy
This parameter specifies the URL of an endpoint from which the WS-SecurityPolicy can be retrieved using WS-MetadataExchange. If omitted, the value `<issuer>/mex` is used. This endpoint must use HTTPS.

Note: This parameter is not supported by Tivoli Federated Identity Manager.

tokenType
This parameter specifies the type of the token to be requested from the STS as a URI. You can omit the parameter under the following circumstances:

- when the STS and the website point of contact have either previously agreed what token type is to be provided, or
- if the website is willing to accept *any* token type.

requiredClaims
This parameter specifies the types of claims that must be supplied by the identity. If omitted, there are no required claims. The value of `requiredClaims` is a space-separated list of URIs, each specifying a required claim type.

optionalClaims
This parameter specifies the types of optional claims that might be supplied by the identity. If omitted, there are no optional claims. The value of `optionalClaims` is a space-separated list of URIs, each specifying a claim type that can be optionally submitted.

privacyURL
This parameter specifies the URL of the human-readable privacy policy of the site, if provided.

privacyVersion
This parameter specifies the privacy policy version. The parameter must be a value greater than 0 if a `privacyUrl` is specified. If this value changes, the UI notifies the user and allows them to review the change to the privacy policy.

Example of a WebSEAL login page

The next figure is an example of the WebSEAL login.html that has been modified with the OBJECT tags shown highlighted in **`bold font`**.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<!-- Copyright (C) 2000 Tivoli Systems, Inc. -->
<!-- Copyright (C) 1999 IBM Corporation -->
<!-- Copyright (C) 1998 Dascom, Inc. -->
<!-- All Rights Reserved. -->
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<TITLE>Access Manager for e-business Login</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000">
<B>Access Manager for e-business Login (www-default---2)</B>
<BR>
%ERROR%
<BR><BR>
<!--- DO NOT TRANSLATE OR MODIFY any part of the hidden parameter(s) --->

<!---
  The following block of code provides users with a warning message
  if they do not have cookies configured on their browsers.
  If this environment does not use cookies to maintain login sessions,
  simply remove or comment out the block below.
--->

<!--- BEGIN Cookie check block --->
<!---
<! ..... edited from this example for brevity ....
<!--- END Cookie check block --->

<BR>
   <form name="ctl00" id="ctl00" method="post"
        action="https://example.com:443/FIM/sps/infocard/login">
      <center>
          <input type="submit" name="InfoCardSignin" value="Log in"
          id="InfoCardSignin" />
      </center>
      <OBJECT type="application/x-informationCard" id="oCard" name="xmlToken">
        <PARAM Name="tokenType" Value="urn:oasis:names:tc:SAML:1.0:assertion">
        <PARAM Name="issuer" Value=
            "http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self">
        <PARAM Name="requiredClaims" Value=
"http://schemas.xmlsoap.org/ws/2005/05/identity/
   claims/privatepersonalidentifier">
        <PARAM Name="optionalClaims" Value=
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname
">      </OBJECT>
    </form>
   </BODY>
</HTML>
```

Figure 21. Example WebSEAL login page with OBJECT tags

Configuration requirements for Information Card

Configure the requirements for the Information Card before you can create a federation.

Requirement for WebSphere Version 6.1

Tivoli Federated Identity Manager supports Information Card on WebSphere Application Server 6.1. Information Card is not supported on WebSphere 6.0. Information Card uses the encryption algorithm **rsa-oaep-mgf1p** for key wrap. This algorithm is supported by WebSphere 6.1, but is not available on WebSphere 6.0.

Tivoli Federated Identity Manager requires application of a fix pack for WebSphere Application Server 6.1. See the hardware and software requirements on the Tivoli Federated Identity Manager information center for the required fix pack level.

Updating the cryptography policy for Information Card

The encryption algorithms used by Information Card require strong cryptographic library support. This means that a replacement is needed for the default Java security files local_policy.jar and US_export_policy.jar.

About this task

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

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

Procedure

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

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

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

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

Information Card requirement for alias service

The alias service must be configured if managed cards backed with self-issued-credential authentication are to be used.

Decryption key from point of contact server

Information Card configuration requires the specification of a key for decrypting messages in the federation. Decryption is required.

This means that a decryption key alias must be added to the Tivoli Federated Identity Manager keystore. The key must be the point of contact server's private key. The key must be imported using the Tivoli Federated Identity Manager key service.

This means that a decryption key alias must be added to the Tivoli Federated Identity Manager keystore. The key is from whichever Web site presents, to the Information Card-enabled browser, the HTML page tagged with the required OBJECT tags. The site can be the point of contact server, but does not have to be. The URL that results in the tagged page *must* use SSL.

For example:

`https://pointofcontact.example.com/FIM`

The SSL key used for the URL must be imported into a Tivoli Federated Identity Manager keystore for the Relying Party.

Note: When the SSL key is changed or updated for the Web site or point of contact, the administrator must also update the Tivoli Federated Identity Manager keystore with the new SSL key. This may also include modification of the configuration to update the keystore alias.

Information Card time synchronization requirements

Successful deployment of Information Card is dependent on time synchronization between systems.

The following requirements must be met to achieve a successful deployment:

- When the UsernameToken method of authentication is configured for a federation, then time must be synchronized between the identity provider and the relying party systems.
- When the self-issued credentials method of authentication for a managed card is used, the browser system (which hosts the browser with Information Card functionality) must also be time synchronized.
- When a self-issued card is used to log on to the relying party, the browser system (which hosts the browser with Information Card functionality) must also be time synchronized.

The required time synchronization can be specified by the **clock skew** property for each Information Card federation. You can use the Tivoli Federated Identity Manager administration console to modify this property from the federation partner properties panel.

Identity mapping for Information Card

The Tivoli Federated Identity Manager support for Information Card identity providers uses a trust chain that contains modules to perform the standard actions of validate, map, and issue.

Identity provider

The validate operation is performed on the authentication token sent by the identity selector to represent the user. The token is either a Username token or SAML assertion. The SAML assertion is used with self-issued credentials authentication.

The mapping module can be one of the following items:

- XSLT mapping module
- Tivoli Directory Integrator module
- A custom-developed Java map module

Tivoli Directory Integrator is commonly used as the mapping module with Information Card. In Information Card deployments, a primary goal of the trust chain is to identify claims values and populate them in the security token service universal user. The claims values can come from external data sources, such as an LDAP registry.

The Tivoli Directory Integrator module can, for example, convert LDAP entries for a user into the corresponding claims values, as defined in the schema that Microsoft has specified.

Tivoli Directory Integrator modules can also easily be used to combine claims values from various sources. For example, some claims values might come from an LDAP registry, while others originate from sources such as databases, Java or JavaScript code, or other web services.

The output of the mapping module is used to produce a SAML 1.1 token in *issue* mode.

Relying party

The trust chain for a relying party federation consists of:

- A SAML 1.1 token module, in validate mode.
- The default Map module.
- The IVCred token module, in issue mode.

The federation wizard prompts the administrator to specify identity mapping rules, using XSLT, as appropriate for the deployment. The mapping rules use the attributes of the assertions or the information in the claims to determine the use identity.

The SAML token modules create STSUniversalUser attributes for each attribute in the SAML assertion. The name, namespace, and value for each SAML attribute are used to set the STSUniversalUser/Attribute name, type, and value.

Identity provider configuration worksheet

Tivoli Federated Identity Manager provides a wizard to guide you through the configuration of Information Card federations. The wizard prompts you to supply properties for your deployment.

This worksheet describes the prompts. Use this worksheet to plan your properties, and refer to it when running the wizard.

Federation name
An arbitrary string that you choose to name this federation. For example:

```
infocard-idp
```

Federation role
Select identity provider.

Company name
The wizard requests contact information. The Company Name field is required. This can be any string. Other fields are optional.

Federation protocol
Select Information Card.

Point of contact server
The server that acts as initial point of contact for incoming requests. For example:

```
https://pointofcontact.example.com/FIM
```

Note: For Information Card support, the point of contact server must use Secure Socket Layer (SSL). The URL must specify `https://`.

SSL Endpoint Key Identifier

The configuration wizard asks you to specify a key to use for decryption operations for the federation. The key must be the key used by the point of contact server for SSL operations.

The wizard asks for this key on the **Infocard Configuration Settings** panel. You specify the key by selecting the Keystore and then the Key.

Note: You must import this key from the point of contact server into the Tivoli Federated Identity Manager keystore before configuring the federation.

Keystore
The Tivoli Federated Identity Manager keystore containing the key.

For example, Tivoli Federated Identity Manager supplies a keystore called DefaultKeystore.

Keystore password
Password required to access the specified keystore.

Key to select
The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use.

Authentication option

You will select one authentication option:

- Authentication with a self-issued card
- Authentication with a username and password

Authentication with a self-issued card is the default option.

The choice of authentication option determines the default value for the property **Download card template file**.

Download card template file
This is an HTML template file that prompts you to enter the input parameters needed to issue a managed Information Card. The configuration wizard provides default values. You can use the defaults unless you have modified and renamed the template files.

- When you select Authentication with a self-issued card, the default value is:

  `/infocard/getcard_sss.html`

- When you select Authentication with a username and password, the default value is:

  `/infocard/getcard_ut.html`

Information card template file
This is an HTML template file that comprises the Information Card that is sent back to you. Default file:

`/infocard/infocard_template.xml`

Information card image file
This is the image file to use for the Information Card. It must be located in the directory for the current locale. The default value is identical for both authentication options. Default file:

`/infocard/fim_infocard.gif`

Card expiration
This property specifies the number of days from the issue date for which the information card is valid. The default value is identical for both authentication options. Default value:

`365`

Identity mapping options
You must select one of the following options:

- Use XSL for identity mapping

  Select this option when you want to use an XSLT mapping rule. You must provide the name of a file that supplies identity mapping rules. Tivoli Federated Identity Manager provides a sample identity mapping rules file for Information Card identity providers federations:

  `/installation_directory/examples/ip_infocard.xsl`

- Use Tivoli Directory Integrator for mapping

  Select this option when you have previously configured a Tivoli Directory Integrator assembly line for the identity mapping required for your Information Card federation.

- Use custom mapping module instance

  Select this option when you have written and deployed a custom trust service module for the identity mapping required for your Information Card federation.

Table 106. Worksheet for identity provider federation properties

Property	Your value
Federation name	
Role	Identity Provider
Company Name	
Federation Protocol	Information Card
Point of Contact server	
SSL Endpoint Key Identifier: Keystore	
SSL Endpoint Key Identifier: Keystore password	
SSL Endpoint Key Identifier: Key to select	
Authentication option	
Download card template file	
Information card template file	
Information card image file	Default: /infocard/fim_infocard.gif
Card expiration	Default: 365 days
Identity mapping options	Select one: • Use XSL for identity mapping • Use Tivoli Directory Integrator for mapping • Use custom mapping module instance
Identity mapping rules file	If using XSL for identity mapping, specify the mapping rule file name:
Custom mapping module	If using a custom mapping module, make note of the name of the module:

Relying party configuration worksheet

Tivoli Federated Identity Manager provides a wizard to guide you through the configuration of Information Card federations. The wizard prompts you to supply properties for your deployment.

This worksheet describes the prompts. Use this worksheet to plan your properties, and refer to it when running the wizard.

Federation name
An arbitrary string that you choose to name this federation. For example, `infocard-rp`.

Federation role
Select service provider. This value is required for the relying party.

Company name
The wizard requests contact information. The Company Name field is required. This can be any string. Other fields are optional.

Federation protocol
Select Information Card.

Point of contact server
The server that acts as initial point of contact for incoming requests. For example:

```
https://pointofcontact.example.com/FIM
```

Note: For Information Card support, the point of contact server must use Secure Socket Layer (SSL). The URL must specify `https://`.

Decryption

The configuration wizard asks you to specify a key to use for decryption operations for the federation. The key must be the key used by the point of contact server for SSL operations.

The wizard for asks for this key on the **Decryption** panel. You specify the key by selecting the Keystore and then the Key.

Note: You must import this key from the point of contact server into the Tivoli Federated Identity Manager keystore before configuring the federation.

Keystore
The Tivoli Federated Identity Manager keystore containing the key.

For example, Tivoli Federated Identity Manager supplies a keystore called DefaultKeystore.

Keystore password
Password required to access the specified keystore.

Key to select
The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use.

Standard Partner
The wizard prompts you to select one option:

- Add a partner that can handle any identity provider

  This option is the default.

  Selecting this option results in a partner being automatically added. This partner configuration can accept any Information Card identity provider, including a self-issuing provider.
- Add a partner that can handle the self-issuing identity provider

  Selecting this option results in a partner being automatically added. This Tivoli Federated Identity Manager partner only accepts personal cards issued by the self-issuing provider built into the browser.
- Do not add a standard partner

  Selecting this option results in no standard partners being added. The administrator must explicitly add partners using the Tivoli Federated Identity Manager console Add Partner wizard.

Identity mapping options
You must select one of the following options:

- Use XSL for identity mapping

  Select this option when you want to use an XSLT mapping rule. You must provide the name of a file that supplies identity mapping rules. Tivoli Federated Identity Manager provides a sample identity mapping rules file for Information Card identity providers federations:

  `/installation_directory/examples/rp_infocard.xsl`
- Use Tivoli Directory Integrator for mapping

  Select this option when you have previously configured a Tivoli Directory Integrator assembly line for the identity mapping required for your Information Card federation.
- Use custom mapping module instance

  Select this option when you have written and deployed a custom trust service module for the identity mapping required for your Information Card federation.

Table 107. Worksheet for relying party federation properties

Property	**Your value**
Federation name	
Role	Service Provider
Company Name	
Federation Protocol	Information Card
Point of Contact server	
Decryption: Keystore	
Decryption: Keystore password	
Decryption: Key to select	
Standard Partner	Default option: **Add a partner that can handle any identity provider.** Your option:
Identity mapping options	Select one: • Use XSL for identity mapping • Use Tivoli Directory Integrator for mapping • Use custom mapping module instance
Identity mapping rules file	If using XSL for identity mapping, specify the mapping rule file name:
Custom mapping module	If using a custom mapping module, make note of the name of the module:

Managed partner worksheet

When you create a federation for an identity provider, a partner is automatically created.

After you create a federation for a relying party, you can choose one of several options for configuring a partner. When you choose not to add a standard partner, you can later create a partner for the federation. When you do this, you will need to provide some configuration values.

The Tivoli Federated Identity Manager console provides a wizard to guide you through this process.

Identity Provider Company name
Contact information.

Security Token Issuer
This value is used to set the protocolID and endpoint URL in etc/feds.xml and the Issuer field in the STS chain mapping configuration. For example:

```
https://example.com
```

Maximum allowable clock skew between hosts (seconds)
This is the maximum allowable clock skew between the relying party host and the identity provider host. The clock skew value is used during validation of the assertion's validity period.

The default value is 60 seconds.

Validate signatures on Information Card tokens
You can select this checkbox to specify that incoming security tokens must be signed. When you select this option, you must use the additional configuration properties to specify the public key that is to be used to validate the digital signature.

Type of signature validation key
You must select one of the following:

- Public key from the KeyInfo in the signature of the Information Card token

  You can choose this option if you do not want to distribute and update public keys, and need only to ensure that token integrity is maintained.
- Public key from a keystore

  This public key must have previously been obtained from the managed identity provider and imported into a Tivoli Federated Identity Managerkeystore using the Key Services.

Keystore
The Tivoli Federated Identity Manager keystore containing the key

For example, Tivoli Federated Identity Manager supplies a keystore called DefaultKeystore.

Keystore password
Password required to access the specified keystore.

Key to select
The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use.

Table 108. Worksheet for managed partner configuration properties

Property	Your value
Identity Provider Company name	
Security Token Issuer	

Table 108. Worksheet for managed partner configuration properties (continued)

Property	Your value
Maximum allowable clock skew between hosts (seconds)	
Validate signatures on Information Card tokens	
Type of signature validation key	If validating signatures, select one: • Public key from the KeyInfo in the signature of the Information Card token • Public key from a keystore
Keystore	*When using **Public key from a keystore***:
Keystore password	
Key to select	

Chapter 22. Configuring an Information Card federation

Verifying Information Card dependencies

Verify that the requirements in creating an Information Card federation have been considered.

Before you begin

Before you use the federation creation wizard, ensure that the Information Card dependences have been met.

Procedure

1. Verify that you are installing on WebSphere Application Server 6.1. Older versions are not supported. See "Requirement for WebSphere Version 6.1" on page 286.
2. Verify that you have the correct encryption libraries. See "Updating the cryptography policy for Information Card" on page 286.
3. Review whether you need to configure the alias service. See "Information Card requirement for alias service" on page 287.
4. Ensure that you have imported the encryption key for the point of contact server This key must be imported into the Tivoli Federated Identity Manager Key service.

Configuring an Infocard federation

To configure a Infocard single sign-on federations, you must create the federation, add your partner to your federation, and provide your partner with configuration information from your new federation.

Before you begin

Ensure that you have prepared configuration information before using the wizard to create the federation. The planning activities are described in a series of topics in this guide. See Chapter 4, "Overview of configuration tasks for federated single sign-on," on page 35.

About this task

To use the federation wizard to create and configure an Infocard federation, complete the steps in this procedure:

Procedure

1. Log on to the Integrated Solutions Console.
2. Click **Tivoli Federated Identity Manager → Configure Federated Single Sign-on → Federations**. The Current Domain and Federations portlets show. The Federations portlet shows several action buttons.
3. Click **Create**. The Federation Wizard starts. The wizard presents a series of configuration panels.

4. Use your completed worksheet to provide values at each panel. Supply the necessary values. You can view the online help for information about specific fields.
   a. The first series of panels requests settings for the federation name, role, protocol, and point of contact server.
   b. Next, the Infocard configuration panel requests the values needed for an Infocard identity provider or relying party.
   c. The last series of panels requests settings for the identity mapping configuration.

   When you finish entering configuration settings, the Summary panel opens.
5. Click **Next** to proceed to the next panel. If you need to go back to adjust a configuration setting, click **Back**.
6. Verify that the configuration settings are correct .
7. Click **Finish**. The Create Federation Complete portlet opens.

Configuring WebSEAL as a point of contact server for an Information Card federation

When you plan to use WebSEAL as the point of contact server, you must configure it for the Information Card federation.

Before you begin

The 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.

These instructions assume that the WebSEAL point of contact profile has been activated.

About this task

The Create Federation Complete portlet provides a button that you can use to obtain the Tivoli Federated Identity Manager configuration utility tool. You must obtain the tool and run it. To configure WebSEAL as the point of contact server, complete the steps in this procedure:

Procedure

1. After creating the federation, click **Load configuration changes to Tivoli Federated Identity Manager runtime** to reload your changes.

   Note: The management console gives you the option of adding a partner now, but for this initial configuration of the federation other tasks are completed first.
2. Click **Done** to return to the Federations panel.
3. Click **Download Tivoli Access Manager Configuration Tool**.
4. Save the configuration tool to the file system on the computer that hosts the WebSEAL server.
5. Run the configuration tool from a command line. The syntax is:

```
java -jar /download_dir/tfimcfg.jar -action tamconfig
-cfgfile webseald-instance_name.conf
```

Note: If Federal Information Processing Standards (FIPS) is enabled in your environment, the secure socket connection factory must be specified. For example:

```
java -jar /download_dir/tfimcfg.jar -action tamconfig
-cfgfile webseald-instance_name.conf -sslfactory TLS
```

You must know the Tivoli Access Manager administration user (default: sec_master) and administration user password. The utility configures endpoints on the WebSEAL server, creates a WebSEAL junction, attaches the appropriate ACLs, and enables the necessary authentication methods.

Example

For example, when you have placed tfimcfg.jar in `/tmp`, and the WebSEAL instance name is `default`, the command is:

```
java -jar /tmp/tfimcfg.jar -action tamconfig -cfgfile webseald-default
```

For more information, see Appendix A, "tfimcfg reference," on page 753.

Configuring WebSphere as a point of contact server

Tivoli Federated Identity Manager is configured by default to use Tivoli Access Manager WebSEAL as the default point of contact server. To configure WebSphere as your point of contact server, you must make a configuration change.

Procedure

1. Log on to the administration console.
2. Click **Tivoli Federated Identity Manager** > **Manage Configuration** > **Point of Contact**.
3. Select **WebSphere**.
4. Click **Make Active**.

Results

The WebSphere server is now configured to be the point of contact server.

Specifying a persona index

A persona index is a collection of several sets of attributes, available to a user on an identity provider. The user can specify attributes that describe a persona. For example, a user might have a work persona containing work email address and telephone, and a home persona containing personal email address and telephone. These personas might be called *work* and *home*.

When a user downloads a managed card, the user might want to associate that managed card with a particular persona. Doing so helps the identity provider determine which set of persona attributes to use to populate the token when a single sign-on token is requested for the card.

The use of personas is enabled by the use of an optional form field parameter called `userdata`. This parameter can be included in getcard_ut.html and getcard_sss.html template pages.

This input field is not included in the shipped template files, but is supported.

When this parameter is provided, a replacement macro called @USERDATA@ can be populated in the infocard_template.html. The @USERDATA@ macro is used in the CardId part of the infocard_template.html file.

The default infocard_template.html file contains the following macro pattern for CardId:

```
<InformationCardReference>
  <CardId>@IPSTS@/@UUID@</CardId>
  <CardVersion>1</CardVersion>
</InformationCardReference>
```

When administrators want to use @USERDATA@, a suggested macro pattern is:

```
<InformationCardReference>
  <CardId>@IPSTS@/@UUID@/@USERDATA@</CardId>
  <CardVersion>1</CardVersion>
</InformationCardReference>
```

The CardId information is part of the RST that the identity selector sends to Tivoli Federated Identity Manager when requesting a single sign-on token. Information mapping rules can read and differentiate which persona to use when you have the CardId information.

Using this pattern, a user would be prompted for their persona index when downloading a managed card, and the card is tied to a particular persona. A user can download different cards for each persona they have specified at the identity provider. The mapping rule in the STS trust chain can read the CardID, and the persona index. It can also populate the runtime identity token with attributes from the correct persona.

Chapter 23. Information Card reference

Replacement macros in the infocard_template XML file

Provides a list of the replacement macros and their uses in the infocard_template XML file.

The replacement macros for infocard_template.xml are:

@IPSTS@
The Uniform Resource Locator (URL) of the identity provider endpoint for the federation.

@IPMEX@
The Uniform Resource Locator (URL) of the identity provider Metadata exchange endpoint for the managed card. Note that the URL is specific to the authentication type used.

@UUID@
This macro is replaced with a randomly generated universal user identifier (UUID). This value ensures that the Card identity is unique.

@USERDATA@
This macro is not included in the default file. You can add this macro to the CardId container when you want to specify attributes. This macro is useful when users in your deployment have multiple personas. The users can provide attributes that identify the persona to be used.

@CARDNAME@
The card name that the user specified in the response posting to the form getcard_ut.html or getcard_sss.html.

@CARDIMAGE@
A Multi-purpose Internet Email Extension (MIME) encoded image file that is displayed to the user by the identity selector. There is one image file for each federation.

@ISSUETIME@
The time the card is issued. The time is calculated at runtime.

@EXPIRETIME@
The time the card expires. The time is calculated by adding card the value of the card *lifetime* to the issue time.

@IPCERTIFICATE@
This is the base64-encoded public certificate configured for the federation. It should also be the public certificate of the SSL endpoint for the point of contact server.

@USERCRED@
This is a piece of metadata about the type of credential that is used by the identity selector to authenticate the user to the identity provider (security token service) endpoint. The metatdata comes from another template file, depending on the type of authentication used.

Tivoli Federated Identity Manager support for Information Card includes support for two forms of authentication:

- Username token

The metadata for the user credential is loaded from the template file inforcard_usercred_usernametoken.xml.

- Self-issued credential

  The metadata for the user credential is loaded from the template file infocard_usercred_selfsignedsaml.xml.

@SUPPORTED_TOKENS@
Tivoli Federated Identity Manager support for Information Card includes the SAML 1.1 token type only. There are two default representations.

@SUPPORTED_CLAIMS@
The set of claims supported by this card. These values come from the form posted by the user in getcard_*.html. The values must be presented in the XML format dictated by the Information Card specifications.

Information Card claims

Provides a list of claim types with the URI and description for each.

The claims types are summarized here for convenience, but users should consult the official list in the referenced schema.

Note: Information Card support in Tivoli Federated Identity Manager is not limited to this set of claims.

First Name
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname

Preferred name or first name of a subject. RFC 2256 uses `givenName` states: "This attribute is used to hold the part of the name of a person which is not their surname nor middle name."

Last Name
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname

Surname or family name of a subject. RFC 2256 uses `sn` and states: "This is the X.500 surname attribute which contains the family name of a person."

Email Address
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress

Preferred address for the `To:` field of email to be sent to the subject, usually of the form `<user>@<domain>`.

The term `mail` is used by inetOrgPerson using RFC1274, which states: "This attribute type specifies an electronic mailbox attribute following the syntax specified in RFC 822."

Street Address
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/streetaddress

Street address component of address information for the subject.

RFC 2256 uses the term `street`, and states: "This attribute contains the physical address of the object to which the entry corresponds, such as an address for package delivery." Its content is arbitrary, but typically given as a PO Box number or apartment or house number followed by a street name. For example, 303 Mulberry St.

Locality Name or City
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/locality

Locality component of the address information for a subject. RFC 2256 uses the term `l`, and states: "This attribute contains the name of a locality, such as a city, county or other geographic region." For example, Austin.

State or Province
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/stateorprovince

Abbreviation for state or province name of the address information for a subject. RFC 2256 uses the term `st`, and states: "This attribute contains the full name of a state or province. The values should be coordinated on a national level and if well-known shortcuts exist, like the two-letter state abbreviations in the US, these abbreviations are preferred over longer full names."

For example, the abbreviation `TX` is used to indicate the state of Texas.

Postal code
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/postalcode

Postal code or zip code component of the address information for a subject. X.500(2001) uses the term `postalCode`, and states: "The postal code attribute type specifies the postal code of the named object. If this attribute value is present, it will be part of the object's postal address - zip code in USA, postal code for other countries."

Country
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/country

Country of a subject. RFC 2256 uses the term `c` and states: "This attribute contains a two-letter ISO 3166 country code."

Telephone Number
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/homephone

Primary or home telephone number of a subject. The term `homePhone` is used in inetOrgPerson using RFC 1274, which states: "This attribute type specifies a home telephone number associated with a person."

Attribute values should follow the agreed format for international telephone numbers. For example, +99 99 999 9999.

Secondary or Work Telephone Number
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/otherphone

Secondary or work telephone number of a subject. X.500(2001) uses the term `telephoneNumber` and states: "This attribute type specifies an office/campus telephone number associated with a person."

Attribute values should follow the agreed format for international telephone numbers. For example, +99 99 999 9999.

Mobile Telephone Number
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/mobilephone

Mobile telephone number of a subject. The term `mobile` is used by inetOrgPerson using RFC 1274, which states: “This attribute type specifies a mobile telephone number associated with a person.”

Attribute values should follow the agreed format for international telephone numbers. For example, +99 99 999 9999.

Date of Birth
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/dateofbirth

The date of birth of a subject in a form allowed by the `xs:date` data type.

Gender
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/gender

Gender of a subject. The value must be one of the following string values:

0 Unspecified

1 Male

2 Female

Use of these values allows them to be language neutral.

Private Personal Identifier
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier

A private personal identifier (PPID) that identifies the subject to a relying party. The word **private** means that the subject identifier is specific to a given relying party and therefore is known only to (or *private to*) that relying party. The PPID of a subject at one relying party cannot be correlated with the PPID for the same subject at another relying party.

Web Page
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/webpage

The Web page of a subject expressed as a URL.

Federation properties for identity providers

When you create an Information Card federation for an identity provider, the configuration wizard automatically assigns default values to some properties.

You cannot modify the federation properties for identity providers during the initial configuration. However, you can modify federation properties after the initial configuration completes.

Federation identification

Federation name
An arbitrary string that you choose to name this federation.

For example, for a managed identity provider:

```
infocard-idp
```

Company name
The wizard requests contact information. The only field that is required in the Company name. The value can be any string.

Single sign-on properties

Provider ID
A unique identifier that identifies the provider to its partner provider. The value consists of the protocol and host name of the identity provider URL. Optionally it can include a port number. For example, for a federation named infocard_fed:

```
https://idp.example.com/sps/infocard_fed/infocard
```

Download Card Endpoint
Endpoint to create and download a managed card. The extension of the file name must be .crd. The default value is:

Provider_ID/getCard.crd

Metadata Exchange Endpoint
The endpoint used by identity selectors to request metadata about the Security Token Service (STS) of the identity provider. The default value is:

Provider_ID/mex

Security Token Service Endpoint
The endpoint used by identity selectors to request security tokens for a user as part of an Information Card authentication. The default value is:

Provider_ID/sts

Alias Management Endpoint
The endpoint used to manage the association, or link between a self-issued card and the Tivoli Federated Identity Manager account of the user. The link is established when a user downloads a managed card using the **self-issued card** authentication mechanism. This endpoint might be used to review and delete that link. The default value is:

Provider_ID/alias

This property is not used when you have selected the authentication option for user name and password.

Authentication option

You can change the authentication option to one of the following options:

- Authentication with a self-issued card
- Authentication with a username and password

Download card template file
This property is an HTML template file that prompts the user to enter the necessary input parameters to issue a managed Information Card.

- When you selected Authentication with a self-issued card, the default value was:

  /infocard/getcard_sss.html
- When you selected Authentication with a username and password, the default value was:

  /infocard/getcard_ut.html

Information card template file
This property is an HTML template file that comprises the Information Card that is sent back to you. Default file:

/infocard/infocard_template.xml

The default value is the same for both authentication options.

Information card image file
This property is the image file to use for the Information Card. It must be located in the directory for the current locale. The default value is identical for both authentication options. Default file:

/infocard/fim_infocard.gif

The default value is the same for both authentication options.

Metadata Card Template
The name of the file to use as the template for the metadata or the Information Card. The default file is:

/infocard/metadata_template.xml.

Self Signed SAML Credentials Metadata Policy
The name of the policy file to use for the self-signed SAML credentials metadata. The default file is:

```
/infocard/metadata_policy_selfsignedsaml.xml
```

This field is only displayed when you select **Authenticate with a self-issued card**.

Username Credentials Metadata Policy
The name of the policy file to use for the user name credentials metadata. The default file is:

```
/infocard/metadata_policy_usernametoken.xml
```

This field is only displayed when you select **Authenticate with a username and password**.

Card expiration
This property specifies the number of days from the issue date for which the Information Card is valid. The default value is identical for both authentication options. Default value:

```
365
```

SSL Endpoint key identifier

Note: The SSL Endpoint key identifier is the key that you must import from the point of contact server into the Tivoli Federated Identity Manager keystore before configuring the federations.

Keystore
The Tivoli Federated Identity Manager keystore containing the key.

For example, Tivoli Federated Identity Manager supplies a keystore called DefaultKeystore.

Keystore password
Password required to access the specified keystore.

List Keys
The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use.

Information Card signing key identifier

The public and private key pair that is used to sign newly issued Information Cards.

Keystore
The Tivoli Federated Identity Manager keystore containing the key.

For example, Tivoli Federated Identity Manager supplies a keystore called DefaultKeystore.

Keystore password
Password required to access the specified keystore.

List Keys
The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use.

Token module properties

When the Information Card federation is initially configured, the trust chain is automatically built and configured. The trust chain contains trust modules that require configuration. The properties in this section can be changed.

Enable one-time assertion use enforcement
Use the assertion only one time and do not cache it for future use. This property is enabled by default.

This property is used only with self-issued card authentication.

Skip password validation
Do not perform password validation for the Username token. The default is unchecked, which means that password validation occurs.

This property is used only with username and password authentication.

Amount of time before the issue date that an assertion is considered valid (seconds)
Default: 60 seconds. There is no minimum or maximum value enforced.

Amount of time the assertion is valid after being issued (seconds)
Default: 60 seconds. There is no minimum or maximum value enforced.

Identity mapping properties

The identity mapping properties are the same as all other protocols supported by Tivoli Federated Identity Manager.

Identity Mapping Module Instance
This value reflects your choice at initial configuration time.

Change Identity Mapping Module Instance
Invokes the Identity Mapping Options panel. The Identity Mapping Options panel enables you to select an XSL transformation, Tivoli Directory Integrator, or a custom mapping module instance.

Modify Current® Properties
Invokes another panel where you can modify properties:

- When the federation uses an XSL transformation, select this option to open the Identity Mapping Rule panel. You can modify or delete the identity mapping rule in this panel.
- When the federation uses a custom mapping module, select this option to open a panel where you can view or modify the custom mapping instance properties.

Federation properties for relying party

Provides a list of values and their descriptions for the federation properties for a relying party.

Federation identification

Federation name
An arbitrary string that you choose to name this federation.

For example, for a relying provider:

```
infocard-rp
```

Company name
The wizard requests contact information. The only field that is required in the Company name. This can be any string.

Single sign-on properties

Provider ID
A unique identifier that identifies the provider to its partner provider. The value consists of the protocol and host name of the identity provider URL. Optionally it can include a port number. For example, for a federation named infocard_fed:

```
https://rp.example.com/sps/infocard_fed/infocard
```

Authentication URL
The URL to which the user sends authentication requests. This value cannot be changed on the Properties panel. For example, for a federation named infocard_fed, the authentication URL would be:

```
https://idp.example.com/sps/infocard_fed/infocard/login
```

Decryption key properties

The key to use for decrypting incoming tokens. Note that this must be the same key that is used for SSL by the point of contact server (for example, WebSEAL).

Keystore
The Tivoli Federated Identity Manager keystore containing the key.

For example, Tivoli Federated Identity Manager supplies a keystore called DefaultKeystore.

Keystore password
Password required to access the specified keystore.

List Keys
The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use.

Identity mapping properties

The identity mapping properties are the same as all other protocols supported by Tivoli Federated Identity Manager.

Identity Mapping Module Instance
This value reflects your choice at initial configuration time.

Change Identity Mapping Module Instance
Invokes the Identity Mapping Options panel. The Identity Mapping Options panel enables you to select an XSL transformation, Tivoli Directory Integrator, or a custom mapping module instance.

Modify Current Properties
Invokes another panel that enables you to modify properties:

- When the federation uses an XSL transformation, this button invokes the Identity Mapping Rule panel. This panel enables you to modify or delete the identity mapping rule.
- When the federation uses a custom mapping module, this button invokes a panel that enables you to view or modify the custom mapping instance properties.

Properties for identity provider partners for relying party federations

Provides a list of property values and their descriptions for identity provider partners for relying party federations.

Federation identification

Member of federation name
The federation to which this partner has been added. You cannot modify this property.

For example, the identity provider is now a partner of the relying provider federation:

```
infocard-rp
```

Partner role
Identity provider. You cannot modify this property.

Status The Partner properties page displays a partner status of Enabled or Disabled. Partners must be enabled (activated) before they can participate in a federation.

- When partner status is Disabled, click Enable to activate the partner.
- When partner status is Enabled, click Disabled to deactivate the partner.

Identity provider company name
The name of the partner company. This can be any string. The space character is allowed. This field is required.

Company URL
The URL of the partner company. This field is optional. For example:

```
http://www.example.com
```

Contact person
Optional contact information for the administrator. You can use the Other information field if necessary.

Token properties

Security Token Issuer
Specify the identity provider's unique issuer Uniform Resource Identifier (URI). This value must be used in the saml:Issuer element of the saml:Assertion. Following is an example:

```
https://example.com
```

You can enter an asterisk (*) to indicate that any identity provider is acceptable.

Maximum allowable clock skew between hosts (seconds)
Specify an integer value indicating the maximum amount of allowable clock skew, in seconds, between the Relying Party host and the Identity Provider host. You must specify a minimum value of zero seconds for this field. The default value is 60. This field is only available when the federation uses the Authenticate with a self-issued card authenticate option.

Signature validation key properties

Validate signatures on Infocard tokens
When checked, indicates that you must sign the Information Card tokens

and then indicate what type of public key to use to validate the digital signature. Clear the check box to turn off signature validation. This check box is selected by default.

Type of signature validation key

- **Public key from the KeyInfo in the signature of the Information Card token**

  Select to use the public key from the KeyInfo in the signature of the Information Card token. This is the default selection.

- **Public key from a keystore**

  Select to use a public key from a keystore. If you select this option, you must select the keystore and key.

Keystore

The Tivoli Federated Identity Manager keystore containing the key.

For example, Tivoli Federated Identity Manager supplies a keystore called DefaultKeystore.

Keystore password

Password required to access the specified keystore.

List Keys

The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use.

Identity mapping properties

The identity mapping properties are the same as all other protocols supported by Tivoli Federated Identity Manager.

Identity Mapping Module Instance

This value reflects your choice at initial configuration time.

Change Identity Mapping Module Instance

Invokes the Identity Mapping Options panel. The Identity Mapping Options panel enables you to select an XSL transformation, Tivoli Directory Integrator, or a custom mapping module instance.

Modify Current Properties

Invokes another panel that enables you to modify properties:

- When the federation uses an XSL transformation, this button invokes the Identity Mapping Rule panel. This panel enables you to modify or delete the identity mapping rule.
- When the federation uses a custom mapping module, this button invokes a panel that enables you to view or modify the custom mapping instance properties.

Properties for relying party partners for identity provider federations

Provides a list of property values and their descriptions for relying party partners for identity provider federations.

Federation identification

Member of federation name

The federation to which this partner has been added. You cannot modify this property.

For example, the relying party is now a partner of the identity provider federation:

```
infocard-idp
```

Partner role
Service provider (Relying party). You cannot modify this property.

Status The Partner properties page displays a partner status of Enabled or Disabled. Partners must be enabled (activated) before they can participate in a federation. You cannot modify this property because this property applies to all relying parties.

Service provider company name
This value indicates that this partner configuration is used for all partners.

For example, for an identity provider federation named infocard-idp, the default value is:

```
All Relying Parties for infocard-idp
```

Company URL
The URL of the partner company. This field is optional. For example:

```
http://www.example.com
```

Contact person
Optional contact information for the administrator. You can use the Other information field if necessary.

Infocard global partner settings

Maximum allowable clock skew between hosts (seconds)
Specify an integer value indicating the maximum amount of allowable clock skew, in seconds, between the Relying Party host and the Identity Provider host. You must specify a minimum value of zero seconds for this field. The default value is 60. This field is only available when the federation uses the Authenticate with a self-issued card authenticate option.

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

Keystore
The Tivoli Federated Identity Manager keystore containing the key.

For example, Tivoli Federated Identity Manager supplies a keystore called DefaultKeystore.

Keystore password
Password required to access the specified keystore.

List Keys
The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use.

Token properties

Include the following attribute types
Specify the types of attributes to include in the assertion. The asterisk (*), which is the default setting, indicates that all of the attribute types that are specified in the identity mapping file or by the custom mapping module will be included in the assertion. To specify one or more attribute types individually, type each attribute type in the box. Use && to separate multiple attribute types.

Include the InclusiveNamespaces element in the canonicalization of the assertion during signature creation
Select to use the InclusiveNamespaces element in the canonicalization of the assertion during signature creation. The default is unchecked.

Include the X509 Certificate data in the KeyInfo element of the signature
Select to use the X509 Certificate data in the KeyInfo element of the signature. The default is checked.

Include the public key data in the KeyInfo element of the signature
Select to use the public key data (X509 public RSA/DSA key) in the KeyInfo element of the signature. The default is checked. KeyInfo element contains information about the key that is needed to validate the signature.

Identity mapping properties

The identity mapping properties are the same as all other protocols supported by Tivoli Federated Identity Manager.

Identity Mapping Module Instance
This value reflects your choice at initial configuration time.

Change Identity Mapping Module Instance
Invokes the Identity Mapping Options panel. The Identity Mapping Options panel enables you to select an XSL transformation, Tivoli Directory Integrator, or a custom mapping module instance.

Modify Current Properties
Invokes another panel that enables you to modify properties:

- When the federation uses an XSL transformation, this button invokes the Identity Mapping Rule panel. This panel enables you to modify or delete the identity mapping rule.
- When the federation uses a custom mapping module, this button invokes a panel that enables you to view or modify the custom mapping instance properties.

Chapter 24. OpenID planning overview

Tivoli Federated Identity Manager supports single sign-on through use of the OpenID protocol.

This overview describes the Tivoli Federated Identity Manager implementation of OpenID. The information in the overview enables an administrator to deploy and configure single sign-on federations.

The OpenID specifications refer to an *OpenID Provider or Identity Provider* as the party who asserts that a user owns a particular identity URL. A Relying Party or *Consumer* is referred to as the party who receives that information from the identity provider. In Tivoli Federated Identity Manager, the term *identity provider* is a direct match for the OpenID concept of *OpenID Provider or Identity Provider*. The OpenID *Consumer* fits well into the Tivoli Federated Identity Manager concept of service provider.

Tivoli Federated Identity Manager support for OpenID authentication allows for all the OpenID message modes:

associate
A mode for establishing a shared secret with the consumer.

checkid_immediate
A mode for performing a non-blocking check to see if a user owns the claimed identifier URL.

checkid_setup
A mode for performing a check to see if a user owns the claimed identifier URL. The check can optionally include interaction with the user.

check_authentication
A mode for determining if a message signature is valid. This mode is typically used for dumb or stateless consumers.

Note: For a complete description of the OpenID specifications, see the Open ID Web site:

http://www.openid.net

OpenID 1.1 and 2.0 support

Both OpenID 1.1 and OpenID 2.0 are supported.

OpenID ID URLs

An OpenID Identity URL is a digital identity designed to be used to authenticate users and grant access to services.

Identity URL with a WebSEAL point of contact

Use the following example values to build an Identity URL with a WebSEAL point of contact:

- An identity provider federation called `openidfedip`

- A Tivoli Federated Identity Manager server where the point of contact server is WebSEAL, with the hostname `webseal.example.com`.
- A user identity (in this case a Tivoli Access Manager user) of `john`.

The OpenID Identity URL can be any URL that meets the following requirements:

- Be resolvable to your Web site. For our example, the URL must either:
  - Start with `http(s)://webseal.example.com`
  - Or, if you are using DNS wildcard entries and a site certificate for `*.example.com`, it could be a value like `http(s)://john.example.com`
- It must contain an identifier that is unique to the user. Typically this identifier is your user identity at the identity provider, however it can be a generated alias for privacy reasons.
- It must match a regular expression that you configure for your OpenID identity provider federation.
- The OpenID identity provider endpoint must be discoverable using either Yadis or HTML discovery from your identity URL as described in the OpenID specifications.

Identity URL with a WebSphere point of contact

Use the following examples values to build the Identity URL with a WebSphere point of contact:

- An identity provider federation called `openidfedip`
- A Tivoli Federated Identity Manager server where the point of contact server is WebSphere, with the hostname `poc.example.com`
- A user identity of `john`

The same requirements apply to the URL as discussed for the first example. Figure 22 shows a sample code when a WebSphere point of contact server is deployed and HTML discovery is used.

```
<html>
<head>
<link rel="openid.server"
href="https://poc.example.com/sps/openidfedip/openid/sso">
<link rel="openid2.provider"
href="https://poc.example.com/sps/openidfedip/openid/sso">
</head>
...
</html>
```

Figure 22. Example code for returning a pointer to your OpenID server from your identity URL using HTML discovery

Note: You can also use Yadis discovery for returning a pointer to your OpenID server from your identity URL.

Example identity URL

When you configure a federation for OpenID, set a regular expression for identity URLs. An easy way to ensure that you can return a link to your OpenID server endpoint from the page returned from your identity URL is to:

1. Ensure that identity URL page is an unprotected page.
2. Embed the OpenID server link in the login form for the point of contact server.

The point of contact server is typically WebSEAL or WebSphere.

This method has the limitation that there can be only one OpenID identity provider federation on the computer. Normally, this restriction causes no problems, and matches the typical deployment of OpenID.

Examples:

- For example, when the configured regular expression is:

```
http://webseal.example.com/@ID@
```

an example identity URL is:

```
http://webseal.example.com/john
```

This simple method of configuration requires no interaction with the user to establish the identity URL. Tivoli Federated Identity Manager determines if a user owns this identity URL by:

1. Replacing the `@ID@` macro in the configured regular expression with the Tivoli Federated Identity Manager username, and
2. Verifying that the identity URL claimed by the user in the single sign-on request is an exact match.

- Another deployment example is one that where the deployment:
  - Uses a site certificate with `CN=*.example.com`
  - Uses a DNS wildcard entry which maps `*.example.com` to a site that is protected by WebSEAL.
  - Allows user to have either `http` or `https` OpenID URLs.

  For this example, the following identity URLs are valid:
  - `john.example.com`
  - `http://john.example.com`
  - `https://john.example.com`

  Note:
  - When the protocol is not specified, as in the first example, http is used.

    The regular expression that is configured for this federation would include the wildcard hostname and multi-protocol support. For example:

    ```
    http[s]?://@ID@.example.com
    ```

    The `@ID@` macro maps to a user name.
  - In some application environments, you might want to use a trailing slash in the patterns for the identity URLs:

    ```
    webseal.example.com/john/
    ```

    Some applications add a trailing slash (/) when normalizing user entry. A mismatch occurs when a trailing slash is added by the application but not specified for the identity URL. Access is not granted.

    In these environments, ensure that the configured regular expression includes the trailing slash. For example:

    ```
    http://webseal.example.com/@ID@/
    ```

Private Personal Identifier Generator

In some authentication scenarios, you might want to maintain the privacy of the user by hiding their identity from the relying party. In addition, you might also want the same user to log in to two different relying parties using different claimed identifiers.

The Private Personal Identifier (PPID) Generator creates the identifier. The relying party is prevented from colliding user identities by using different claimed identifiers.

This type of authentication scenario is called *directed identity*. Directed identity requires the user to initiate login at the relying party using a shared identity provider identifier. For example `https://example.ibm.com`

Depending on the configuration, the OpenID Provider generates an identifier for the user of a specific relying-party. A Private Personal Identifier (PPID) Generator creates the identifier. The OpenID Provider generates a separate identifier for each relying party to which the same user authenticates.

Creating different claimed identifiers prevents information sharing between relying-parties. This feature also effectively protects the identity of the user.

Using the Private Personal Identifier Generator feature requires the OpenID Provider to advertise its server endpoint information using an Extensible Resource Descriptor Sequence (XRDS) document. The XRDS document is required.

A user can only log in to a relying party using an identity provider identifier that is discoverable through the XRDS document. The XRDS document is the only way for the relying party to differentiate between the claimed identifier of a user and an identity provider identifier.

A plug-in provides a Private Personal Identifier Generator in the identity provider implementation. The plug-in provides several standard generator implementations. An administrator can also use the plug-in to write and integrate a custom IDGenerator. This feature determines how to generate the identity of a claimed identifier for a particular user at a particular relying party.

When an identity provider identifier is used at the relying party to initiate authentication, the identity provider is responsible for generating the claimed identifier for the user. Tivoli Federated Identity Manager generates a claimed identifier using a simple configured pattern URL. The URL must contain the `@ID@` macro. The value of the `@ID@` is generated by the PPID generator.

For example, the default configuration is:

```
https://myidp.com/@ID@
```

The following IDGenerators can be used to replace the `@ID@` macro of the identity URL:

- Username ID Generator
- Hash ID Generator
- Alias service ID Generator

Username ID Generator

When the Username ID Generator is used, a username is returned as the `@ID@` portion of the expression for identity URLs.

For example, when the identity URL expression is:

```
http://webseal.example.com/@ID@
```

an example identity URL is:

```
http://webseal.example.com/john
```

This setting is the default behavior of Tivoli Federated Identity Manager.

Hash ID Generator

The Hash ID Generator replaces the `@ID@` value with a sha256 hash value. This hash value is a combination of the current federation ID, the username, and the relying party trust root.

The benefit of hash mode is that the username is not exposed to each site used for OpenID single sign-on. The hash value is fast to generate with no external lookups. This hiding of the account name helps to protect the user from malicious hackers who are intent on identity theft. Exposing the account name provides a staring point for phishing attacks or for locking a user from an account.

For example, when the configured regular expression is:

```
http://webseal.example.com/@ID@
```

an example identity URL is:

```
http://webseal.example.com/
3d0f1d5e9a3a617771608b390b5c7fc1601a3839f161060cbad8e93b98f034c2
```

Alias service ID Generator

The Alias Service implementation automatically assigns a randomly generated UUID for the `@ID@` value.

On first use, a UUID is generated and stored in the alias service. The lookup key for the UUID is based on the username, the current federation ID, and the relying party trust root. On subsequent uses, the same UUID is retrieved from the alias service. This method ensures that a consistent identifier is used for the user at that particular relying party.

Like the hash mode, the username is not exposed to each site used for OpenID single sign-on.

For example, when the configured regular expression is:

```
http://webseal.example.com/@ID@
```

an example identity URL is: `http://webseal.example.com/c84911b2-0124-14f0-991a-a5a8f0e6f99d`

Avoiding reuse of user identities for identity URLs

OpenID identity URLs must never be reused. Once a URL has been assigned to an individual user, it must never be reassigned to another user. This specification is important because any consumer Web site to which the original user has authenticated can still have an account associated with the URL.

The requirement to ensure that OpenID identity URLs are never reused must be enforced by the deployment environment. Tivoli Federated Identity Manager cannot check for reuse. The provisioning of user names must follow a process that ensures that each URL is allocated only once.

Identity provider federations

OpenID identity provider federations share similarities with other single sign-on federations supported by Tivoli Federated Identity Manager. However, the concepts of *federation* and *partners* are applied differently.

A key difference is that an OpenID identity provider does not need to know about the consuming party in advance. Shared secret negotiation is part of the protocol, and no pre-configuration of keys or partners is necessary.

In OpenID, the user is involved in the decision on whether to trust particular consuming partners. The decision is made by examining the *trust_root* URL on the consent-to-authenticate page. This means that the concept of partner service providers is unnecessary.

Note: In OpenID 2.0, the *trust_root* is called a *realm*.

The federation configuration contains some partner configuration properties, but these properties are used by the token modules for the security token service.

The OpenID federation naming follows the standard Tivoli Federated Identity Manager naming convention for a unique identifier or `protocolID`. The syntax is:

`https://<hostname:port>/FIM/sps/<federation_name>/openid`

For example:

`https://www.example.com/FIM/sps/openidfedip/openid`

Single sign-on endpoint

The single sign-on endpoint is the OpenID server URL. It supports requests from the consumer and from the browser, when redirected by the consumer. This URL requires unauthenticated access, in order that queries from anonymous consumer clients can be made for the following message modes:

- associate
- checkid_immediate
- check_authentication

When a checkid_setup request is received, and the user has not previously trusted the consumer, this URL also supplies the *consent-to-authenticate* prompt.

This endpoint returns authentication results to consuming sites.

Example endpoint:

```
https://webseald.example.com/FIM/sps/openidfedip/openid/sso
```

Authentication endpoint

When a user has not previously logged into an identity provider, the single sign-on endpoint redirects the browser to this authentication endpoint. The user is then authenticated. This endpoint is required during checkid_setup operations when the user has not already authenticated to the identity provider, and single sign-on is initiated from a consumer.

When authentication succeeds, the end point typically redirects the user back to the single sign-on endpoint for further processing. The redirect is provided as a query-string parameter. The syntax is:

```
<protocolID>/authn?return=<url>
```

For example, as one continuous string:

```
https://webseald.example.com/FIM/sps/openidfedip/openid/authn?return=
      https://webseald.example.com/FIM/sps/openidfedip/sso
```

Site management endpoint

When the identity provider receives a checkid_setup message, the identity provider asks the user for permission (consent) to provide the consumer authentication and attribute information for the user.

The identity provider uses a page template, and a browser cookie for that user to remember the user preferences. The identity provider must be able to retrieve the saved preferences in order to successfully answer messages in the checkid_immediate message mode, and to automate single sign-on responses for checkid_setup mode.

Tivoli Federated Identity Manager saves user preferences through use of a trusted sites manager extension point. The extension point uses a pluggable interface, which enables administrators to replace the default extension implementation with a custom implementation that, for example, supports a server-side storage model. Another purpose for this extension point is that a custom implementation might be used to auto-consent all trust decisions in a closed authentication environments.

The identity provider uses the trusted site manager to manage trusted and untrusted consumer sites. When the site manager asks the user to provide consent to authenticate, the user can specify policy for the specified consumer, as follows:

- Always Allow
- Allow Once
- Deny Once
- Always Deny

The user can later use the site manager to access and modify the saved preferences. Users can optionally remove a permanently trusted site or untrusted site from the list. When a user does this, the user is prompted with consent-to-authenticate on the next attempted single sign-on to that consumer.

The trusted site manager also remembers any optional attributes requested by a service provider (if the user has allowed the attributes to be shared).

The endpoint completes the following tasks:

1. Uses an HTML template to prompt the user with their set of permanently remembered trusted and untrusted sites.
2. Allow the user to remove sites from the permanent list.

The syntax for the endpoint URL is:

```
<protocolID>/sites
```

For example:

```
https://webseald.example.com/FIM/sps/openidfedip/openid/sites
```

Identity provider trust chains

In the OpenID model, the consumer can require that specific attributes be provided for each user identity. Tivoli Federated Identity Manager uses a trust service chain on the identity provider to obtain the attributes and place them in a simple XML token.

When a user contacts the identity provider by presenting an OpenID identity URL, the identity provider verifies the user identity. When the identity provider is operating in either checkid_immediate or checkid_setup mode, the trust service is started to acquire and populate the attribute data. In addition, the trust service is used to validate that any requested Provider Authentication Policy Extension (PAPE) authentication requirements have been met.

The identity provider uses a chain of trust service modules primarily to enable the retrieval of required and optional attribute values. The trust chain follows the standard module flow:

1. Validate

   The validate operation is performed on an IVCred token that is generated from the authentication credential for the user.
2. Map

   The mapping module can be any of the supported module types. When attribute data for the user can be extracted from the IVCred input token, an XSLT mapping rule is often a good option. A Tivoli Directory Integrator mapping module, or a custom Java mapping module, is useful when the attribute data must be obtained from an external source.
3. Issue

   The issue operation produces a Security Token Service Universal User (STSUU) token. The token supplies the set of required and optional attributes to the single sign-on protocol service, along with validated PAPE authentication information. This operation enables the service to generate an OpenID login response, or prompts again for authentication if needed to satisfy additional requested PAPE policies.

In order for the mapping module to populate required and optional attributes, it must know the list of required and optional attributes. The list of required and optional attributes are sent to the trust service in claims. The requested PAPE information is also available to the mapping rule in claims information.

The list of claims can also contain user preference data. For example, a persona index can be posted in the consent-to-authenticate form. The index is retrieved from the trusted consumers management extension point, and included in the claims.

```
<fimopenid:OpenIDClaims
    xmlns:fimopenid="urn:ibm:names:ITFIM:openid"
    xmlns:fimpape="urn:ibm:names:ITFIM:openid:PAPE"
    xmlns:fimqs="urn:ibm:names:ITFIM:queryservice"
    ClaimedId="http://specs.openid.net/auth/2.0/identifier_select"
    DiscoveredIdentifier="https://www.myidp.ibm.com/FIM/op/
85da8845-0127-1a04-9a9f-dca9f50a9649"
    IdentityURL="http://specs.openid.net/auth/2.0/identifier_select"
    IsOPIdentifierLogin="true"
    IsRPReturnToValidated="false"
    OPLocalId="http://specs.openid.net/auth/2.0/
identifier_select"
    OpenIDServerURL="https://www.myidp.ibm.com/
FIM/sps/openididp/openid/sso"
    PolicyURL="http://www.ibm.com"
    ReauthCount="0"
    ReturnTo="https://www.myrp.ibm.com/sps/myrp/openid/
loginreturn?nonce=uuid85d96a6f-0127-1f6e-bafb-c3b7deb3ed5d"
    TrustRoot="https://www.myrp.ibm.com/"
    Userdata=""
    Version="http://specs.openid.net/auth/2.0">
  <fimopenid:PrincipalName>shane</fimopenid:PrincipalName>
  <fimqs:RequestedAttributes>
    <fimqs:Attribute name="openid.sreg.email" optional="false" />
    <fimqs:Attribute name="openid.sreg.nickname" optional="true" />
    <fimqs:Attribute name="openid.sreg.fullname" optional="true" />
  </fimqs:RequestedAttributes>
  <fimpape:OpenIDPAPEClaims>
    <fimpape:Attribute name="openid.pape.preferred_auth_levels">
      <fimpape:Value>urn:ibm:names:ITFIM:5.1:accessmanager</fimpape:Value>
    </fimpape:Attribute>
    <fimpape:Attribute name="openid.pape.preferred_auth_policies">
      <fimpape:Value>http://schemas.xmlsoap.org/ws/2005/05/
identity/claims/privatepersonalidentifier</fimpape:Value>
      <fimpape:Value>http://www.idmanagement.gov/schema/2009/05/
icam/openid-trust-level1.pdf</fimpape:Value>
    </fimpape:Attribute>
  </fimpape:OpenIDPAPEClaims>
</fimopenid:OpenIDClaims>
```

Figure 23. Example claims during the identity provider invocation of the trust service

Figure 23 shows an example of claims being passed into the trust service. Note the optional claims in the `RequestedAttributes` list:

- `openid.sreg.email`
- `openid.sreg.nickname`
- `openid.sreg.fullname`

When attributes are required, the optional value is set to `false`.

In the example, notice that the property `userdata` is an empty string. This empty string indicates that there was no optional data defined during the consent-to-authenticate. This data (and the reading of it in the mapping module of the chain) is where persona-specific attribute retrieval can be accomplished for a user.

Handling large amounts of user attribute data

OpenID authentication works with URL redirects. When an authentication response from the identity provider must contain more than 2 kb of user registry

data, Tivoli Federated Identity Manager automatically switches to POST messages. This behavior supports the OpenID 2.0 specification, which allows for auto-posting POST transactions for indirect messages.

Note: The automatic switch to POST messages is not supported in OpenID 1.1 deployments.

Relying Party Discovery

Using Relying-Party (RP) discovery, OpenID Providers can detect and verify the `return_to` addresses of realms that support OpenID.

Relying-Party discovery is performed when an OpenID Provider receives a solicited single sign-on request. Relying-Party then performs the discovery process on the URL specified in the `openid.realm` parameter of the sign-on message. Relying parties must publish their `return_to` URL in XRDS.

An administrator can configure the identity provider properties panel to enforce successful Relying-party discovery. The macro in the consent.html page makes it possible for the identity provider to indicate if the Relying-Party discovery has not been done yet. For more information about consent to authenticate page see, .

This specification enables OpenID Providers to verify authentication requests and ensure that responses are redirected to valid `return_to` endpoints.

If discovery cannot verify the `return_to` URL on the Relying Party realm Tivoli Federated Identity Manager either shows an error or warning, depending on the configuration.

The IsRPReturnToValidated claim attribute tells the mapping rule if the `return_to` URL validation occurred. Tivoli Federated Identity Manager adds this attribute to the OpenIDClaims element passed to the security token service. It enables a mapping rule to detect when Relying-Party discovery fails and performs an appropriate action. The value for this claims attribute can be `true` or `false`.

Authentication modes

OpenID support two authentication modes: checkid_immediate, and checkid_setup

The checkid_immediate authentication mode is typically used in rich client environments where a widget does the following tasks:

- Determine whether a browser user owns a particular claimed OpenID URL
- Avoid having the browser interact with the user

To initiate a checkid_immediate request from the consumer to an identity provider, add this input parameter in the login form:

```
<input type=''hidden'' name=''openid.mode'' value=''checkid_immediate''>
```

The Tivoli Federated Identity Manager single sign-on protocol URL endpoint initiates the login from the consumer.

- When the response from the identity provider is a successful assertion that the user owns the identity URL, Tivoli Federated Identity Manager performs a security token service token exchange, and login to the point of contact server. This behavior is the same as for checkid_setup.

- When the response from the identity provider is a failed assertion, an HTML page template is loaded from the page factory. The replacement macro in the page is populated with the value of the `open.user_setup_url` parameter that was returned from the identity provider.

The checkid_setup authentication mode allows the identity provider to interact with the user, to request authentication or self-registration before returning a result to the consumer. When no authentication mode is specified in the login form, checkid_setup is the default mode.

Since checkid_setup is the default mode, it is not necessary to specify the mode in the login form. However, the consumer can specifically request this mode. See the following code to request this mode:

```
<input type=''hidden'' name=''openid.mode'' value=''checkid_setup''>
```

Tivoli Federated Identity Manager support for checkid_setup is a federated single sign-on flow with redirect to the identity provider for authentication, including user interaction for approval of sign-on. The result of the authentication flow is the return of signed response attributes to the consumer. When the digital signature is validated, the attributes are built into a Security Token Service Universal User (STSUU) token and sent to the trust service for exchange for an IVCred credential. The credential is then used for the login.

Consumer federations

The Tivoli Federated Identity Manager OpenID *consumer* plays a role like a *service provider* in other single sign-on protocols.

The OpenID consumer uses a Tivoli Federated Identity Manager federation that has some similarities to, but significant differences from, the federations for other single sign-on protocols.

In particular, there is no need to directly associate identity provider partners with OpenID consumer federations. The key exchange and association with particular identity providers is controlled by the OpenID identity URL, as determined at runtime. Partners are not added and configured for an OpenID service provider federation.

The Tivoli Federated Identity Manager federation entity for the consumer contains:

- A login endpoint
- A login return endpoint
- A trust root URL (known as a *realm* in OpenID 2.0)
- Parameters indicating the type of map module in the trust chain
- Any associated configuration parameters
- User-agent policy controlling the allowed range of IP addresses, networks, and (or) hostname patterns for OpenID identity URLs and OpenID server endpoints.

The syntax for the OpenID federation protocolID is:

```
https://<hostname:port>/FIM/sps/<federation name>/openid
```

For example:

```
https://webseald.example.com/FIM/sps/openidfedsp/openid
```

The login endpoint

The Tivoli Federated Identity Manager consumer supports a login URL. The login URL receives the POST of the initial login form and initiates a checkid_setup or checkid_immediate.

Using the previous example federation protocolID, the endpoint is:

```
https://webseald.example.com/FIM/sps/openidfedsp/openid/login
```

Note: An example endpoint for deployments with WebSphere as point of contact server is:

```
https://poc.example.com/sps/openidfedsp/openid/login
```

The single sign-on delegate at the endpoint completes the following tasks:

1. Determines from the incoming login form the OpenID identity URL plus any extension parameters.
2. Determine the canonical form of the identity URL, in accordance with the applicable OpenID authentication specification. Yadis and HTML discovery are supported.
3. Retrieve the final identity URL, including delegates, for the user. Determine the OpenID server for the user.
4. When an association does not exist with an identity provider, establish one.
5. Build a checkid_setup or checkid_immediate request to the OpenID server and redirect the browser to the identity provider.

The login return endpoint

Tivoli Federated Identity Manager support a login return URL. The browser is redirected by the identity provider to this URL after single sign-on processing is complete. This endpoint is passed as the openid.return_to parameter during the single sign-on request.

For example, this endpoint would be:

```
https://webseald.example.com/FIM/sps/openidfedsp/openid/loginreturn
```

The single sign-on delegate at this endpoint processes responses from checkid_setup and checkid_immediate. The delegate processes these responses, and any check_authentication requests or association handle invalidation that might occur as a result.

- When a response is returned with a successfully validated signature, the trust service uses the parameters in the response. The parameters are used to build a Security Token Service Universal User (STSUU) token. The delegate uses the trust service to exchange the STSUU token for an IVCred credential. The credential is then used for Tivoli Federated Identity Manager authentication.
- When the response is returned with an unsuccessful response, an error page is displayed.

The trust root or realm URL

The Tivoli Federated Identity Manager consumer also supplies a *trust root* or *realm* URL. This URL serves as the basis for trust shown to the user at the identity provider.

Tivoli Federated Identity Manager reads the trust root URL from configuration properties. This property is initially generated by entries done by the administrator to combine the following values:

- Protocol

  For example, `https`.
- Hostname

  Host name for the point of contact server
- Port

  Optional. Specified only when not the standard port.
- A forward slash (/)

For example:

`https://webseald.example.com/`

OpenID login

The Tivoli Federated Identity Manager consumer presents a login form to request the OpenID URL from the user. The form can use either POST or GET methods to the Tivoli Federated Identity Manager consumer login endpoint. The included parameters can contain more than the URL if required.

Tivoli Federated Identity Manager supports:

- OpenID 1.1 authentication specifications
- OpenID 2.0 authentication specifications
- OpenID Simple Registration Extension 1.0
- OpenID Simple Registration Extension 1.1
- OpenID Attribute Exchange Extension
- Provider Authentication Policy Extension 1.0

Note: The method for login by the Tivoli Federated Identity Manager consumer is the same when accessing either a Tivoli Federated Identity Manager identity provider, or another identity provider.

For example, consider the following deployment scenario:

- WebSEAL as the point of contact for a host called `www.example.com`
- An OpenID consumer federation called `openidfedsp`

Figure 24 shows a sample login form for this example.

```
<html>
  <form method="post"
  action="https://www.example.com/FIM/sps/openidfedsp/openid/login">
    <img src="login-bg.gif" /> 
    <input type="text" name="openid_identifier" /> 
    <input type="submit" value="Login" />
  </form>
</html>
```

Figure 24. Simple OpenID login form

The Tivoli Federated Identity Manager service provider completes the following steps:

1. Reads the `openid_identifier` parameter
2. Performs the authentication flow specified for OpenID Authentication 2.0
3. Performs an External Authentication Interface (EAI) login to WebSEAL

After a successful checkid_immediate or checkid_setup response, the Tivoli Federated Identity Manager consumer calls the trust service to perform any required attribute or user identity manipulation.

During the login process, the consumer can request attributes from the identity provider by specifying additional parameters in the login form. The parameters must correspond to the parameter names described in the OpenID Simple Registration Extension 1.0. You can also use other supported specifications such as Simple Registration Extension 1.1, Attribute Exchange 1.0 and Private Personal Identifier Generator 1.0.

For example, Figure 25 shows a login form that accomplishes the following requirements using Simple Registration Extension:

- Requires the e-mail address from the identity provider
- Requires the date of birth from the identity provider
- Optionally requests the full name for the user
- Provides a policy URL that links to a page that describes a privacy policy

```
<html>
  <form method="post"
  action="https://www.example.com/FIM/sps/openidfedsp/openid/login">
    <input type="hidden" name="openid.sreg.required"
     value="email,dob" />
    <input type="hidden" name="openid.sreg.optional"
     value="fullname" />
    <input type="hidden" name="openid.sreg.policy_url"
value="http://www.example.com/privacy_policy.html" />
    <img src="login-bg.gif" /> 
    <input type="text" name="openid_identifier" /> 
    <input type="submit" value="Login" />
  </form>
</html>
```

Figure 25. OpenID login form with registry extension parameters

When these parameters are present in the login request, Tivoli Federated Identity Manager sends them to the identity provider. This action is done during checkid_immediate and checkid_setup requests.

The parameters do not have to be hidden, and do not have to be a comma-separated list.

The parameters can consist of multi-valued attributes. The use of multi-valued attributes enables the server to present the user with radio buttons, list boxes, or other multi-valued widgets in the HTML. Tivoli Federated Identity Manager treats each value as a comma-separated list. Multiples values consisting of one entry only (each) are allowed.

You can implement login with automatic redirection to a specified URL. When WebSEAL is the point of contact server, the rules for processing EAI authentication apply. You can include an optional `TARGET` parameter in the login form, to redirect the user after successful authentication.

Template pages

The Tivoli Federated Identity Manager consumer uses several template HTML pages when processing authentication requests and errors:

- When the consumer processes a checkid_immediate request, and the identity provider cannot determine if the OpenID URL for the user is valid, the consumer returns a template file.

  See "Template page returned for checkid_immediate" on page 364.
- The consumer uses a template file as part of supporting POST transport for large indirect messages. The Tivoli Federated Identity Manager consumer supports POST transport for large indirect messages. The support uses a template file.

  See "Template page for OpenID 2.0 indirect post" on page 363.
- When a checkid_immediate or checkid_setup request results in an error, the consumer uses a template file to return an error.

  See "Template page returned for server error" on page 365.
- When an error occurs on the consumer that halts processing, the consumer uses a template file to return the error.

  See "Template page for OpenID error" on page 362.

Consumer trust chains

During the login process, Tivoli Federated Identity Manager handles attribute and identity mapping. When a checkid_immediate or checkid_setup response comes back from the identity provider and reports a successful assertion, Tivoli Federated Identity Manager builds all the attributes and PAPE response data returned from the identity provider into a Security Token Service Universal User (STSUU) token, and uses the trust service to exchange that token for an IVCred credential.

The trust chain consists of:

- An STSUU Token in validate mode

  The token contains the OpenID identity URL plus any extension parameters including user attributes. The STSUU token is built by the OpenID login return delegate when it has verified the signature on a login response from the identity provider.
- A mapping module

  The consumer can use the map module to do any necessary identity and attribute mapping.

  The type of mapping module to use for the consumer federation is set when the federation is configured. The standard map module types are supported:

  – XSLT or Javascript mapping rules
  – Tivoli Directory Integrator mapping module
  – Custom mapping modules.

  In many cases, the use of scripted mapping rules is sufficient, since typically there are no external attributes to retrieve.

  The Tivoli Federated Identity Manager product distribution includes sample scripted mapping rules and a sample Tivoli Directory Integrator assembly line (mapping module).
- A IVCred credential in issue mode

Account linking

An important consumer scenario for OpenID is to perform account linking. For example, when a user has authenticated directly to a website which is also an OpenID consumer, the website might enable the user to link their account with an OpenID. By performing an OpenID login while the user is already logged in to the website, the consuming site can associate that OpenID with the current logged in account.

To support this scenario, Tivoli Federated Identity Manager sends claims in a WS-Trust call to the security token service. The call includes the current logged in user name when an authenticated session exists.

Figure 26 on page 329 shows an example of the claims format sent to the trust service. This is available to map module implementors as part of the STSUU.

```
<fimopenid:OpenIDClaims
xmlns:fimopenid="urn:ibm:names:ITFIM:openid"
xmlns:fimpape="urn:ibm:names:ITFIM:openid:PAPE"
ClaimedId="https://www.myidp.ibm.com/FIM/op/
85da8845-0127-1a04-9a9f-dca9f50a9649"
IdentityURL="https://www.myidp.ibm.com/FIM/op"
IsOPIdentifierLogin="true"
IsRPReturnToValidated="false"
NormalizedIdentityURL="https://www.myidp.ibm.com/FIM/op/
85da8845-0127-1a04-9a9f-dca9f50a9649"
OPLocalId="https://www.myidp.ibm.com/FIM/op/
85da8845-0127-1a04-9a9f-dca9f50a9649"
OpenIDServerURL="https://www.myidp.ibm.com/FIM/
sps/openididp/openid/sso"
ReauthCount="0"
ReturnTo="https://www.myrp.ibm.com/sps/myrp/openid/
loginreturn?nonce=uuid85e85b2a-0127-1286-99d4-d5e72a774a5f"
Signed="openid.op_endpoint,openid.return_to,
openid.response_nonce,openid.assoc_handle,
openid.claimed_id,openid.identity,
openid.sreg.dob,openid.sreg.gender,
openid.sreg.email,openid.sreg.language,
openid.sreg.timezone,openid.sreg.fullname,
openid.sreg.postcode,openid.sreg.country,
openid.sreg.nickname,openid.ns.sreg,
openid.ns.pape,openid.pape.auth_time,
openid.pape.auth_policies,openid.pape.auth_level.ns1,
openid.pape.auth_level.ns.ns1"
Target="https://www.myrp.ibm.com/fimivt/protected/ivtlanding.jsp"
Version="http://specs.openid.net/auth/2.0">
<fimpape:OpenIDPAPEClaims>
<fimpape:Attribute name="satisfied_auth_age">
<fimpape:Value>true</fimpape:Value>
</fimpape:Attribute>
<fimpape:Attribute name="openid.pape.preferred_auth_levels">
<fimpape:Value>urn:ibm:names:ITFIM:5.1:accessmanager
</fimpape:Value>
</fimpape:Attribute>
<fimpape:Attribute name="satisfied_auth_policies">
<fimpape:Value>true</fimpape:Value>
</fimpape:Attribute>
<fimpape:Attribute name="openid.pape.preferred_auth_policies">
<fimpape:Value>http://www.idmanagement.gov/schema/
2009/05/icam/openid-trust-level1.pdf</fimpape:Value>
<fimpape:Value>http://schemas.xmlsoap.org/ws/2005
/05/identity/claims/privatepersonalidentifier</fimpape:Value>
</fimpape:Attribute>
</fimpape:OpenIDPAPEClaims>
</fimopenid:OpenIDClaims>
```

Figure 26. OpenID claims during a Consumer WS-Trust call

The PrincipalName attribute in the claims, when present, contains the Tivoli Federated Identity Manager user name of the currently authenticated user. This enables trust chains, through the use of mapping rules, to automatically associate a particular OpenID with an existing account.

The example contains other claims which are parameters from the single sign-on response from the OpenID server. The Identity URL attribute is what the user presented in the login form as the identity URL. The NormalizedIdentityURL attribute is the canonical form of the identity URL that results from normalization that occurs as part of the discovery process.

The STSUU token sent with the request to the trust service contains attributes for each of the listed components of the `Signed` set of attributes, plus any other query string parameters.

Figure 27 shows the STSUU generated from the WS-Trust call shown in Figure 26 on page 329.

```
<?xml version="1.0" encoding="UTF-8" ?>
<stsuuser:STSUniversalUser xmlns:stsuuser="urn:ibm:names:ITFIM:1.0:stsuuser">
<stsuuser:Principal><stsuuser:Attribute name="name"><stsuuser:Value>https://www.myidp.ibm.com/FIM/
op/85da8845-0127-1a04-9a9f-dca9f50a9649</stsuuser:Value></stsuuser:Attribute>
</stsuuser:Principal><stsuuser:AttributeList><stsuuser:Attribute name="openid.identity">
<stsuuser:Value>https://www.myidp.ibm.com/FIM/op/85da8845-0127-1a04-9a9f-dca9f50a9649</stsuuser:Value>
</stsuuser:Attribute></stsuuser:AttributeList><stsuuser:RequestSecurityToken />
<stsuuser:ContextAttributes><stsuuser:Attribute name="openid.op_endpoint">
<stsuuser:Value>https://www.myidp.ibm.com/FIM/sps/openididp/openid/sso</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.sreg.email">
<stsuuser:Value>jsmith@ibm.com</stsuuser:Value></stsuuser:Attribute>
<stsuuser:Attribute name="openid.sig"><stsuuser:Value>NuKNV1ypZC16d3og6HbvjbCedPVjhRbWAWZ9Gq6g1DU=
</stsuuser:Value></stsuuser:Attribute>
<stsuuser:Attribute name="openid.pape.auth_level.ns1"><stsuuser:Value>1</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.claimed_id">
<stsuuser:Value>https://www.myidp.ibm.com/FIM/op/85da8845-0127-1a04-9a9f-dca9f50a9649</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.ns">
<stsuuser:Value>http://specs.openid.net/auth/2.0</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.sreg.language">
<stsuuser:Value>en</stsuuser:Value></stsuuser:Attribute>
<stsuuser:Attribute name="openid.sreg.fullname"><stsuuser:Value>John Smith</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="nonce">
<stsuuser:Value>uuid85e85b2a-0127-1286-99d4-d5e72a774a5f</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.pape.auth_time">
<stsuuser:Value>2010-03-22T12:47:41Z</stsuuser:Value> </stsuuser:Attribute>
<stsuuser:Attribute name="openid.return_to"><stsuuser:Value>https://www.myrp.ibm.com/sps/myrp/
openid/loginreturn?nonce=uuid85e85b2a-0127-1286-99d4-d5e72a774a5f</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.signed"><stsuuser:Value>
op_endpoint,return_to,response_nonce,assoc_handle,claimed_id,identity,sreg.dob,
sreg.gender,sreg.email,sreg.language,sreg.timezone,sreg.fullname,
sreg.postcode,sreg.country,sreg.nickname,ns.sreg,ns.pape,pape.auth_time,
pape.auth_policies,pape.auth_level.ns1,pape.auth_level.ns.ns1</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.sreg.nickname">
<stsuuser:Value>Smithy</stsuuser:Value></stsuuser:Attribute>
<stsuuser:Attribute name="openid.identity">
<stsuuser:Value>https://www.myidp.ibm.com/FIM/
op/85da8845-0127-1a04-9a9f-dca9f50a9649</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.ns.sreg">
<stsuuser:Value>http://openid.net/extensions/sreg/1.1</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.pape.auth_level.ns.ns1">
<stsuuser:Value>urn:ibm:names:ITFIM:5.1:accessmanager</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.sreg.dob">
<stsuuser:Value>1980-12-25</stsuuser:Value></stsuuser:Attribute>
<stsuuser:Attribute name="openid.sreg.postcode"><stsuuser:Value>99999</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.assoc_handle">
<stsuuser:Value>uuid85ca8353-0127-1776-9b7b-c75a4586c507</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.sreg.country">
<stsuuser:Value>AU</stsuuser:Value></stsuuser:Attribute>
<stsuuser:Attribute name="openid.pape.auth_policies">
<stsuuser:Value>http://schemas.xmlsoap.org/ws/2005/05/
identity/claims/privatepersonalidentifier http://www.idmanagement.gov/schema/2009/05/icam/
openid-trust-level1.pdf</stsuuser:Value></stsuuser:Attribute>
<stsuuser:Attribute name="openid.mode"><stsuuser:Value>id_res</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.sreg.timezone">
<stsuuser:Value>Australia/Brisbane</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.ns.pape">
<stsuuser:Value>http://specs.openid.net/extensions/pape/1.0</stsuuser:Value>
</stsuuser:Attribute><stsuuser:Attribute name="openid.sreg.gender">
<stsuuser:Value>M</stsuuser:Value> </stsuuser:Attribute>
<stsuuser:Attribute name="openid.response_nonce">
<stsuuser:Value>2010-03-22T12:48:08Zuuid85ea7849-0127-1515-b2b5-e9223d6c6970
</stsuuser:Value></stsuuser:Attribute></stsuuser:ContextAttributes>
<stsuuser:AdditionalAttributeStatement />
</stsuuser:STSUniversalUser>
```

Figure 27. Example STSUU during trust service request at the OpenID Consumer

User agent policy

The user agent can be configured to restrict the set of locations that it tried to access. This setting is done to prevent malicious users attempting to make the user-agent connect to internal resources.

The Consumer uses a *user agent* (HTTP client) to connect directly to OpenID identity URLs, and to the OpenID server URLs they reference. The Provider also uses the same type of user agent policy configuration for relying-party discovery operations.

The restrictions are managed through the use of a static connection policy configuration and a customizable dynamic endpoint authorization module. The dynamic endpoint module can be used to further restrict access to endpoints at runtime.

Each Tivoli Federated Identity Manager federation has one global policy setting. The setting defines default behavior when the host is not explicitly found in the allow or deny access lists. This setting either permits or denies access to URLs as a default behavior. In addition to the global policy, administrators can create custom dynamic endpoint access plug-ins. These plug-ins can check an online list of trusted or untrusted endpoints. Administrators can add the custom plug-ins to the list of dynamic endpoint access authorization modules.

Static connection policy

With a static connection policy, an administrator can list allowed and denied hosts. Depending on the selected default behavior, the administrator also can specify a list of hosts in the allow or deny list.

When the default behavior selected is **deny**, only hosts in the **allow** lists can be accessed by the user agent. This setting is restrictive. Every OpenID identity URL and server for which you want to allow access must be covered in the allow lists. When the default behavior set to **deny**, any *deny access lists* are not useful. By default all hosts are denied unless they are explicitly included in an allow list.

When the default behavior selected is **allow**, the host is contacted, unless it is included in a *deny list*. This setting is more liberal and generally enables users to log in from any legitimate OpenID server on the Internet. However, when the default setting selected is **allow**, the deny lists must be carefully configured.

Tivoli Federated Identity Manager supports the following types of lists:

Allow lists:

- A user-configurable list of hostname regular expressions
- A user-configurable list of IP address netmasks (IPv4 and IPv6)

Deny lists:

- A user-configurable list of hostname regular expressions
- A user-configurable list of IP address netmasks (IPv4 and IPv6)
- A built-in list of default-deny hostname regular expressions
- A built-in list of default-deny IP address netmasks

Note: The allow lists take precedence over the deny lists.

The access lists for host names follow the standard Java Regular Expression syntax as defined by the Pattern class. The list uses regular expressions to match the host names.

The built-in deny lists cannot be changed by users. However, the list can be overridden to allow certain entries by adding the hostname regular expressions or netmasks to the user-configurable allowed lists.

Figure 28 shows the default-deny host names. The default values provide protection against attacks that try to access arbitrary URLs on the local system.

```
.*\.localdomain
localhost
```

Figure 28. Default-deny hostname regular expressions

Figure 29 shows the default-deny IP address netmasks. These netmasks include various non-routable IPv4 and IPv6 addresses. This list can be overridden by adding the networks that you want to allow connection to the allowed list of IP netmasks.

```
0.0.0.0/8
10.0.0.0/8
127.0.0.0/8
169.254.0.0/16
172.16.0.0/12
192.168.0.0/16
255.255.255.255
::/128
::1/128
::/96
fc00::/7
fe80::/10
ff00::/8
```

Figure 29. Default-deny IP address netmasks

Dynamic endpoint access plug-in

A dynamic endpoint access plug-in is a custom module. An administrator can create the custom dynamic endpoint access plug-in to check external lists of trusted and untrusted hosts.

When an administrator selects a custom **dynamic endpoint access plug-in** in the dynamic endpoint access authorization module, the software checks specified endpoints. The specified endpoints are checked to determine if they can be trusted. You can use this setting with the allow or deny access list. However, if you set the dynamic endpoint authorization to the default access approval, the software uses only endpoints in the allow or deny lists.

Example – allowing any Internet OpenID server, deny access to 9.x.x.x intranet

- To configure this environment, the default access policy show be **allow**
- The allowed hosts list is ignored.
- The denied hosts list is ignored.

- The denied IP address netmask is 9.0.0.0/8.

  Multiple netmasks can be added if there is more than one intranet, and IPv6 equivalents must be added if the network supports both IPv4 and IPv6.

Example – only allow OpenID login from example1 and example2 companies

- To configure this environment, the default access policy show be **deny**
- The list of denied hosts and IP address netmasks are ignored.
- The allowed-hosts regular expression list is:

  ```
  .*\.example1\.com,openid\.example2\.com,openidserver\.example2\.com
  ```

Example1 OpenIDs look like `john.example1.com` and the OpenID server that the identity URLs resolves to is:

```
https://www.example1.com/openidProcessing.action
```

For example 2, OpenIDs look like `openid.example2.com/<example2_screenname>`, and this resolves to an HTML page which points to the OpenID server:

```
https://api.screenname.exmple2.com/auth/openidServer
```

The need for this URL is why both these host names appear in the list.

Example - allow any hostname containing .ibm.com string

This example shows the settings to allow a user to access any hostname containing the .ibm.com string.

- To configure this environment, select a custom plug-in.
- The allowed hosts list is checked.
- The denied hosts list is checked.
- The custom dynamic endpoint plug-in is:

```
package com.tivoli.am.fim.demo.ibmaccessapproval;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

import com.tivoli.am.fim.useragent.AccessApproval;

public class IBMAccessApproval implements AccessApproval {

 final static String CLASS = IBMAccessApproval.class.getName();

 final static Logger _log = Logger.getLogger(CLASS);

 public IBMAccessApproval() {
 }

 public boolean canAccess(Map ctx) {
  String methodName = "canAccess";
  _log.entering(CLASS, methodName, new Object[] { ctx });
  boolean result = false;
  boolean finestLoggable = _log.isLoggable(Level.FINEST);
  try {
   String endpoint = (String) ctx.get(AccessApproval.CTX_ENDPOINT);
   String fedname = (String) ctx
     .get(AccessApproval.CTX_FEDERATION_NAME);
   String fedid = (String) ctx
```

```
    .get(AccessApproval.CTX_FEDERATION_ID);

    if (finestLoggable) {
     _log.logp(Level.FINEST, CLASS, methodName, "Fedname: "
       + fedname + " Fedid: " + fedid + " Endpoint: " + endpoint);
    }

    try {
     URL u = new URL(endpoint);
     String hostname = u.getHost();
     if (hostname != null && hostname.indexOf(".ibm.com") > 0) {
      result = true;
     }
    } catch (MalformedURLException e) {
     e.printStackTrace();
    }

   } finally {
    _log.exiting(CLASS, methodName, "" + result);
   }
   return result;
  }
}
```

OpenID Extensions

OpenID Simple Registration Extension

During the login process, the consumer can request attributes from identity providers by specifying additional parameters in the login form. The parameters must correspond to the parameter names described in the OpenID Simple Registration Extension 1.0 or Attribute Exchange Extension 1.0, whichever is applicable. Simple Registration Extension (SREG) is an extension to the OpenID Authentication protocol and supports a simple list of common user registration information.

For more information, see OpenID documentation at: http://openid.net/specs/openid-simple-registration-extension-1_0.html

```
<form name="openidLoginForm" method="post"
action="https://sp.example.com/FIM/sps/openidsp/openid/login">
<input name="openid.mode" type="hidden"
value="checkid_setup">
<input name="openid.sreg.required" type="hidden"
value="email">
<input name="openid.sreg.optional" type="hidden"
value="fullname,dob">
<input name="openid.sreg.policy_url" type="hidden"
value="https://sp.example.com/privacy_policy.html">
<input name="TARGET" type="hidden"
value="https://sp.example.com/myapp">
<input name="openid_identifier" type="text">
<input value="OpenID Login" type="submit">
</form>
```

Figure 30. Sample Simple Registration Extension

OpenID Attribute Exchange Extension

Identity providers can use OpenID extensions to obtain and communicate user attributes to consumers.

The attribute exchange extension provides identity providers the ability to communicate user attributes to consumers.

The Attribute Exchange Extension (AX) protocol can be extended to accommodate varying types of attributes and multi-valued attributes. The attributes are identified by a unique URI and typically correspond to personal identity information. For more information see OpenID documentation at: http://openid.net/specs/openid-attribute-exchange-1_0.html

Attribute Exchange Extension provides strict compatibility with OpenID 2.0. You can use either or both extensions simultaneously. Use Attribute Exchange Extension unless you need to be compatible with older OpenID 1.1 implementations that only support SREG.

As an administrator, you can add a set of parameters to the OpenID login form posted to the login endpoint.

The example shows a login form with the following requirements:

- Requires the e-mail address from the identity provider
- Optionally requests for the full name, date of birth, friends and groups.

```
<form name="openidLoginForm" method="post"
action="https://sp.example.com/FIM/sps/openidsp/openid/login">
<input name="openid.mode" type="hidden" value="checkid_setup">
<input name="openid.ax.required" type="hidden" value="axemail">
<input name="openid.ax.if_available" type="hidden"
value="axfullname,axdob,axfriends,axgroups">
<input name="openid.ax.type.axemail" type="hidden"
value="http://axschema.org/contact/email">
<input name="openid.ax.type.axfullname" type="hidden"
value="http://axschema.org/namePerson">
<input name="openid.ax.type.axdob" type="hidden"
value="http://axschema.org/birthDate">
<input name="openid.ax.type.axfriends" type="hidden"
value="http://example.com/myschema/friends">
<input name="openid.ax.count.axfriends" type="hidden"
value="5">
<input name="openid.ax.type.axgroups" type="hidden"
value="http://example.com/myschema/groups">
<input name="openid.ax.count.axgroups" type="hidden"
value="unlimited">
<input name="TARGET" type="hidden"
value="https://sp.example.com/myapp">
<input name="openid_identifier" type="text">
<input value="OpenID Login" type="submit">
</form>
```

Figure 31. Sample Attribute Exchange Extension

Note: If no explicit count is requested for an attribute exchange parameter, the default max count value is 1.

Tivoli Federated Identity Manager sends parameters to the identity provider during `checkid_immediate` and `checkid_setup` requests. The fetch messages sent with the request retrieves the personal identity attributes of the user. For additional information about fetch messages see the OpenID documentation: http://openid.net/specs/openid-attribute-exchange-1_0.html#fetch

Attribute Exchange Extension fetch requests parameters

The Attribute Exchange Extension supports an information model that combines a subject identifier, an attribute type identifier, a count, and a value. Including additional parameters attaches the Attribute Exchange Extension fetch request on a standard authentication request. To enable the consumer to retrieve information from the identity provider, specify the following form field parameters in the login form.

openid.ax.required
Fetches required attributes from the identity provider. The value is a list of aliases, which are labels that represent individual attributes at the identity provider. Bind each alias to a URI that identifies the attribute in a separate `openid.ax.type.`*`alias`* parameter. *(Optional)*

openid.ax.if_available
Fetches an attribute that is available from the identity provider. The value has the same requirements as `openid.ax.required`. *(Optional)*

Note: You must specify either `openid.ax.required` or `openid.ax.if_available` in the request. Each requested attribute alias must have an associated `openid.ax.type.`*`alias`* parameter.

openid.ax.type.*alias*
Binds the alias to a URI that defines the meaning of the attribute. You must specify a parameter for each alias specified in either `openid.ax.required` or `openid.ax.if_available`. *(Optional)*

Many typical attributes already have defined type URIs at http://www.axschema.org/types/

openid.ax.sendalways
Includes OpenID Attribute Exchange Extension information in authentication requests to the identity provider. The consumer runtime sends Attribute Exchange Extension request information if the identity provider advertises Attribute Exchange Extension support with XRDS. The default value is false. *(Optional)*

Attribute Exchange Extension fetch response parameters

After granting access to an identity provider, a fetch response message supplies the information in the fetch request parameters. The following optional fetch response parameters specify the retrieved personal attributes from the identity provider.

openid.ax.type.*alias*
Specifies the URI type for the fetched attribute identified by alias. *(Optional)*

openid.ax.count.*alias*
Returns the number of values specified for the attribute that corresponds to alias. If you do not specify a specific value, it returns only one value.

openid.ax.value.*alias*
Assigns a value specified for the attribute that corresponds to alias. *(Optional)*

openid.ax.value.*alias***.number**
Assigns a value specified for the attribute that corresponds to alias. This parameter is required if `openid.ax.count.`*`alias`* is sent and at least one

value is configured for the associated attribute. There should be a separate parameter for each value for the alias, with incrementing numbers.

OpenID Provider Authentication Policy Extension

Use the administration console to configure the OpenID Provider Authentication Policy Extension.

When a user initiates authentication from a relying party with an OpenID identifier, the relying party requests the identity provider to authenticate the user.

The OpenID Provider Authentication Policy Extension (PAPE) is a mechanism where the relying party can:

- request identity providers to use specific authentication policies when authenticating a user.
- require an identity provider to inform the relying party of the authentication policies used during authentication.
- require an identity provider to communicate the levels of authentication used as defined in sets of requested custom assurance levels.

Depending on your role in the federation, certain parameters are available in the configuration properties panel of the administration console.

Note: PAPE settings can only be configured AFTER creating a federation. Use the Federation Properties panel to specify the configuration settings.

Relying party PAPE implementation

Use the relying party configuration properties panel to enable PAPE. Once enabled, the specified PAPE attributes are sent to the identity provider in the authentication request. When a relying party sends the authentication request with the specified PAPE attributes, the identity provider sends a response. The response indicates which requirements have been met and which have not. Based on the response, the relying party can determine whether to authenticate a user.

Specify the following parameters in the relying party configuration properties panel:

Enforcement Mode

- Strict

  Specifies that a user is not authenticated if the PAPE requirements are not met.
- Lenient

  Specifies that a user is authenticated even if the PAPE requirements are not met. The mapping rule used in the federation accesses the response information. The response indicates which requirements have been met and which have not been met. This setting allows the author of the mapping rule to decide whether to log in the user based on that information. The mapping rule provides a more restricted authorization.

Authentication policies

Specifies a set of authentication policy URIs. The URIs represent authentication policies to be satisfied by the identity provider when authenticating a user. If multiple policies are requested, the identity provider must satisfy as many of them as it can. The identity provider then indicates which authentication policies were satisfied in the response.

Maximum authentication age
Specifies the length of time in which the user must have been authenticated. If this time has expired, the identity provider must re-authenticate the user.

Preferred assurance level
Specifies an ordered list of preferred assurance level namespace URIs. The assurance level namespace values determine the level of trust placed in the authentication of the user. Relying parties request information about these assurance level namespaces from the identity provider.

Identity provider PAPE implementation

The identity provider configuration properties panel specifies the conditions for when a user must authenticate.

Note: If you plan to use WebSEAL cookie management with OpenID PAPE implementation, ensure that the list of managed cookies does not include the WebSphere session cookie. See "Configuring WebSEAL to manage cookies" on page 532

Specify the following parameters in the identity provider configuration properties panel:

Force authentication on any requested PAPE maximum authentication age
This parameter specifies that a user must always authenticate. If selected, the Maximum authentication age allowable clock skew field is disabled.

Maximum authentication age allowable clock skew
When a maximum authentication age is requested by a service provider during single sign-on, the identity provider mapping rule must return the last authentication time of the user. This parameter is used to account for clock skew between:

- the last authentication time returned by the identity provider mapping rule
- the clock of the identity provider

Typically the skew time is a small number, but can account for differences between the point of contact machine and the runtime machine.

Identity provider configuration worksheet

Tivoli Federated Identity Manager provides a wizard to guide you through the configuration of OpenID federations. The wizard prompts you to supply properties for your deployment. This worksheet describes the properties.

Use this worksheet to plan your properties, and refer to it when running the wizard.

Federation name
The name can be any character string. For example, `openid-idp`. This field is required.

Federation role
Your role is *identity provider*.

Company name
The name of the company that is creating this federation. The value can be any string. You can also use the space character along with the other characters. This field is required.

Federation protocol
OpenID.

Point of contact server
The URL address of the server that acts as initial point of contact for incoming requests. The address consists of a protocol specification, the server host name, and (optionally) a port number. When WebSEAL is the point of contact server, the WebSEAL junction is specified.

Example value:

```
https://webseald.example.com/FIM
```

Note: For OpenID support, the point of contact server must use Secure Socket Layer (SSL). The URL must specify `https://`.

Association expiration (seconds)
Specifies the lifetime of the association handle. This identity provider controls this value. Enter a positive number. The default value is 3600 seconds.

Response nonce expiration time (seconds)
Indicates how many seconds a relying party that is operating without an established association has, before they must perform the `check_authentication` request. If set to a positive number, this feature prevents replay of `check_authentication`. This restriction applies to customers with Relying Parties that cannot create or store associations. The default value is 30 seconds.

ID Generator
Specifies which ID generator creates a value that replaces the `@ID@` of an identity URL. Different ID generators create different values for `@ID@`.

OpenID Identity URL pattern

Represents the regular expression on which identity URLs are matched for the federation. Tivoli Federated Identity Manager replaces the `@ID@` part. The default value is the URL for the single sign-on protocol host name provided by the installation wizard.

For example, if you specified the following point of contact server in the wizard:

```
https://webseald.example.com/FIM
```

the default Identity URL pattern is:

```
https://webseald.example.com/@ID@
```

User setup URL
Specifies the URL that is sent in response to a `checkid_immediate` request from a consumer. The URL is used when the identity provider is unable to determine if a user owns a particular identity URL.

The default URL is the point of contact server URL you specified on the Point of Contact Server panel.
For example, when you have previously specified, in the wizard, a point of contact server:

```
https://webseald.example.com/FIM
```

the default user setup URL is:

```
https://webseald.example.com/
```

Trusted Sites Manager
Selects the implementation class for a trusted sites manager. The implementation persists data concerning consent-to-authenticate decisions made by a user during OpenID authentications.

Support OP Identifier
Specifies if the `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`.

You can use a configuration option to enable or disable support for OP identifier.

To enable the configuration option, you must change `<was_config_root >/itfim/<i><tfim_domain></i>/etc/feds.xml` on the identity provider to add this parameter:

```
<fc:EntityProperty name="OPENID.IPSupportOPIdentifier"

<fim:Value>true</fim:Value>

</fc:EntityProperty>
```

If the value is missing, the default value is false.

If the `OPENID.IPSupportOPIdentifier` is true, then this additional parameter must also be included:

```
<fc:EntityProperty name="OPENID.IPGeneratedClaimedIDPattern">

<fim:Value>see_below</fim:Value>

</fc:EntityProperty>
```

Use a string template for the value. It contains the `@ID@` pattern that is replaced with the user name of the user.

Note: The parameter is similar to the existing Identity Pattern parameter that is used to verify regular claimed identifier logins (not OP-identifier logins). The exception is that the `OPENID.IPGeneratedClaimedIDPattern` parameter is not a regular expression. It is just a template that must include the replacement macro `@ID@`. For example the value can be:

```
https://myidp.com/@ID@
```

OP Generated Claimed Identifier Pattern
Specifies a valid URL that must contain the `@ID@` string. It enables a relying party to initiate single sign-on with a claimed identifier set to `identifier_select`.

The default URL is derived from the point of contact server URL specified during federation configuration.

For example, if the point of contact URL was specified as:

```
https://webseal.example.com/FIM
```

the default OP Generated Claimed Identifier Pattern is:

```
https://webseal.example.com/@ID@
```

Relying-Party Discovery Options
Provides two options:

Perform RP Discovery
Specifies whether to attempt relying party discovery. Not selecting this option disables all other options for relying party discovery.

Require successful RP Discovery
Specifies if Tivoli Federated Identity Manager halts with an error when it cannot complete relying party discovery for the identity provider. This option applies only if you enable Perform RP Discovery.

RP Discovery cache expiration time
Determines how many seconds to cache information discovered about relying parties in seconds. If you enter less than zero, information is never cached.

Permitted OpenID Server Protocols
Specifies the allowed protocols for the OpenID servers to which the user agent permits connection. You can choose either or both values. For best practice, the parameter is typically is set to HTTPS only.

Choose one or both of the following:

- HTTPS
- HTTP

HTTP Connection Timeout
Specifies how many seconds before a timeout occurs during communications with the HTTP client. Enter a positive number. If you enter zero (0), the software uses the Java defaults for URLConnection objects. The default value is 30 seconds.

Keystore
Specifies the keystore used to validate the certificates of SSL endpoints during communications for relying-party discovery. This keystore must contain the certified authority signer certificates of all relying-parties for which relying-party discovery is to be performed.

User Agent Connection Policy
Specifies policy for connections by the user agent. You must select one of the following options.

- Allow access to OpenID hosts by default

  The host is contacted, unless it is included in the deny list. This setting is more liberal and generally enables users to log in from any legitimate OpenID server on the Internet.
- Deny access to OpenID hosts by default

  Only hosts in the allow lists can be accessed by the user agent. This setting is restrictive, and every OpenID identity URL and server for which you want to allow access must be covered in the allow lists.

To review the policy choices, see "User agent policy" on page 331.

Allowed Hostname Regular Expressions
Specifies a list of regular expressions that identify host names to which the user agent can request access. Enter one string per line.

For example:

```
.*\.ibm\.com
```

The value is optional.

Allowed IP Address / Netmasks

Specifies IP addresses or netmasks to which the user agent can request access. User regular expressions, and enter one string per line. Enter one string per line.

For example:

```
10.1.1.0/24
192.168.0.10
```

This value is optional.

Dynamic Endpoint Access Authorization Module

Specifies a list of custom dynamic endpoint access plug-ins. The plug-ins can check external lists of trusted and untrusted hosts. This setting is used in addition to configuration in the User Agent Connection Policy. If set to the default access approval, configuration settings specified under User Agent Connection Policy are used.

Denied Hostname Regular Expressions

Specifies the host names to which the user agent cannot request access. User regular expressions, and enter one string per line.

- When the User Agent Connection Policy is set to Deny access to OpenID hosts by default, this property is not used.
- When the User Agent Connection Policy is set to Allow access to OpenID hosts by default, use of this property is optional.

For example:

```
.*\.example\.com
.*\.example2\.com
```

Denied IP Address / Netmasks

Specifies a list of regular expressions that identify IP addresses or netmasks to which the user agent cannot request access. Enter one string per line.

- When the User Agent Connection Policy is set to Deny access to OpenID hosts by default, this property is not used.
- When the User Agent Connection Policy is set to Allow access to OpenID hosts by default, use of this property is optional.

For example:

```
11.12.13.0/24
192.168.0.10
```

Identity mapping options

Select one of the following options:

- **Use XSLT or Javascript mapping rules for identity mapping**

  Select this option when you create an XSLT or a Javascript mapping rule that supplies identity mapping rules.

  Tivoli Federated Identity Manager provides a sample identity mapping rules file for OpenID identity provider federations:

  `/installation_directory/examples/ip_openid.xsl`

- **Use Tivoli Directory Integrator for mapping**

  Select this option when you have a Tivoli Directory Integrator assembly line for the identity mapping required for your OpenID federation.

- **Use custom mapping module instance**

  Select this option when you have a custom trust service module for the identity mapping required for your OpenID federation.

Table 109. Worksheet for federation identification properties

Property to specify	**Your value**
Federation name	
Role	identity provider
Company name	
Federation Protocol	OpenID
Point of Contact server	
Association expiration time (seconds)	Default: 3600 seconds
Response nonce expiration (seconds)	Default: 30 seconds
ID Generator (generates value of @ID@)	
OpenID Identity URL Pattern	
User Setup URL	
Support OP Identifier	OP Generated Claimed Identifier Pattern
OP Generated Claimed Identifier Pattern	
Perform RP Discovery	
Require successful RP Discovery	
RP Discovery cache expiration time	
Permitted OpenID Server Protocols	
HTTP Connection Timeout	
Keystore	
User Agent Connection Policy	
Allowed Hostname Regular Expressions	
Allowed IP Addresses / Netmasks	
Dynamic Endpoint Access Authorization module	
Denied Hostname Regular Expressions	
Denied IP Addresses / Netmasks	
Identity mapping options	Select one: • Use XSLT or JavaScript for identity mapping • Use Tivoli Directory Integrator for mapping • Use custom mapping module instance

Table 109. Worksheet for federation identification properties (continued)

Property to specify	Your value
Identity mapping rules file	If using XSLT or JavaScript for identity mapping, specify the mapping rule file name:
Custom mapping module	If using a custom mapping module, make note of the name of the module:

Consumer configuration worksheet

Tivoli Federated Identity Manager provides a wizard to guide you through the configuration of OpenID federations.

The wizard prompts you to supply properties for your deployment. This worksheet describes the properties.

Use this worksheet to plan your properties, and refer to it when running the wizard.

Federation name
An arbitrary string that you choose to name this federation. For example, `openid-consumer`.

Federation role
Your role is *service provider*. You must select *service provider* when configuring the consumer role.

Company name
A string value for the Company name. You can optionally provide additional contact information.

Federation protocol
OpenID.

Point of contact server
The URL address of the server that acts as initial point of contact for incoming requests. The address consists of a protocol specification, the server hostname, and (optionally) a port number. When WebSEAL is the point of contact server, the WebSEAL junction is specified. Example value:

```
https://webseald.example.com/FIM
```

Note: For OpenID support, the point of contact server must use Secure Socket Layer (SSL). The URL must specify `https://`.

Advertised trust root
This value is a URL that is the *root* of the trust URLs for the federation. It defaults to the base URL for the federation, where the base URL is the path to the single sign-on protocol service hostname. When the port is not the default number, the port value is also included. The value must end in a forward slash (/).

This value must be a URL parent of the login return delegate endpoint. It is used as the openid.trust_root parameter in the single sign-on request that is sent to the identity provider.

For example, when you have previously specified, in the wizard, a point of contact server:

`https://webseald.example2.com/FIM`

the default authentication trust root is:

`https://webseald.example2.com/`

Enable Yadis Protocol
Specifies whether to perform the Yadis discovery. For best practices, choose enabled.

Enable XRI Identifiers
Specifies whether to resolve URL or XRI-based claimed identifiers. If you do not select an option, the software uses only URL-based claimed identifiers.

XRI Proxies
Specifies a list of URLs for resolving XRI identifiers. The URL must contain the `@XRI@` macro.

Discovered information expiration
Specifies how long the cache stores the discovered information. If you do not enter a positive number, the cache is disabled and discovery is performed at every login.

Response nonce skew time
Specifies a value in seconds used for validating the response nonce from OpenID 2.0 identity providers. Validation is only performed if this skew is a positive number. Validation is performed by taking the time of the response nonce and the configured response nonce skew.

If the number of seconds is outside this range, the authentication response is rejected. If the number of seconds is within this range, a response nonce cache is checked. The check ensures that the authentication response is not a replay.

When validation is successful, the response nonce is added to the response nonce cache for as long as it would be within the skew period. The response nonce is added to the nonce cache to ensure that future authentication responses are not replays.

Permitted OpenID Server Protocols
This value represents the set of allowed protocols for the OpenID servers to which the user agent permits connection. For best practice, the parameter is typically is set to HTTPS only.

Choose one or both of the following:

- HTTPS
- HTTP

HTTPS is the default. You must select at least one protocol.

HTTP Connection Timeout
This value specified the communications timeout for the HTTP client. The value must be a valid positive integer. The maximum value is the maximum integer value. A value of zero (0) means to use the Java defaults for URLConnection objects. The default value is 30 seconds.

Keystore
This value is the name of a keystore that has previously been configured in

the Tivoli Federated Identity Manager key service. The keystore must hold certificate authority signer certificates only.

The HTTP client for the consumer uses this keystore when communicating with SSL-enabled identity providers. The keystore is used to determine if the host that is to be connected to can be trusted. This check occurs when processing `associate` and `check_authentication` messages.

Default:

`DefaultTrustedKeyStore`

User Agent Connection Policy

This value specifies policy for connections by the user agent. You must select one of the following options.

- Allow access to OpenID hosts by default
- Deny access to OpenID hosts by default

To review the policy choices, see "User agent policy" on page 331.

Allowed Hostname Regular Expressions

A list of regular expressions that specify host names to which the user agent can request access. Enter one string per line. For example:

```
.*\.ibm\.com
```

This value is optional.

Allowed IP Addresses / Netmasks

A list of regular expressions that specify IP addresses or netmasks to which the user agent can request access. Enter one string per line. For example:

```
10.1.1.0/24
192.168.0.10
```

This value is optional.

Denied Hostname Regular Expressions

A list of regular expressions that specify host names to which the user agent cannot request access. Enter one string per line. For example:

```
.*\.example\.com
.*\.example2\.com
```

- When the User Agent Connection Policy is set to **Deny access to OpenID hosts by default**, this property is not used.
- When the User Agent Connection Policy is set to **Allow access to OpenID hosts by default**, use of this property is optional.

Denied IP Addresses / Netmasks

A list of regular expressions that specify IP addresses or netmasks to which the user agent cannot request access. Enter one string per line. For example:

```
11.12.13.0/24
192.168.0.10
```

- When the User Agent Connection Policy is set to **Deny access to OpenID hosts by default**, this property is not used.
- When the User Agent Connection Policy is set to **Allow access to OpenID hosts by default**, use of this property is optional.

Dynamic Endpoint Access Authorization Module

Specifies a list of custom dynamic endpoint access plug-ins. The plug-ins can check external lists of trusted and untrusted hosts. This setting is used

in addition to configuration in the User Agent Connection Policy. If set to the default access approval, configuration settings specified under User Agent Connection Policy are used.

Identity mapping options

You are asked to select one of the following options:

- Use XSLT or Javascript mapping rules for identity mapping

  Select this option when you have created an XSLTfile or a Javascript mapping rule that supplies identity mapping rules.

  Tivoli Federated Identity Manager provides a sample identity mapping rules file for OpenID consumer federations:

  `/installation_directory/examples/sp_openid.xsl`
- Use Tivoli Directory Integrator for mapping

  Select this option when you have previously configured a Tivoli Directory Integrator assembly line for the identity mapping required for your OpenID federation.
- Use custom mapping module instance

  Select this option when you have written and deployed a custom trust service module for the identity mapping required for your OpenID federation.

Table 110. Configuration properties for OpenID consumer

Property	**Your value**
Federation name	
Role	service provider
Company name	
Federation Protocol	OpenID
Point of Contact server URL	
Advertised trust root	
Enable Yadis Protocol	
Enable XRI Identifiers	
XRI Proxies	
Discovered information expiration	
Response nonce skew time	
Permitted OpenID Server Protocols	`HTTPS` or `HTTP` or both
HTTP Connection Timeout (seconds)	Default: 30 seconds
Keystore	
User Agent Connection Policy	
Allowed Hostname Regular Expressions	
Allowed IP Addresses / Netmasks	

Table 110. Configuration properties for OpenID consumer (continued)

Property	Your value
Denied Hostname Regular Expressions	
Denied IP Addresses / Netmasks	
Dynamic Endpoint Access Authorization module	
Identity mapping options	Select one: • Use XSL for identity mapping • Use Tivoli Directory Integrator for mapping • Use custom mapping module instance
Identity mapping rules file	If using XSL for identity mapping, specify the mapping rule file name:
Custom mapping module	If using a custom mapping module, make note of the name of the module:

Chapter 25. Configuring OpenID

Configure an OpenID federation by creating the federation, setting up the point of contact server, and updating the information on the login pages.

Verifying OpenID dependencies

Verify that the requirements in creating an OpenID federation have been considered.

Before you begin

Before you use the Federation Creation wizard, ensure that the OpenID dependences have been met. Complete the required planning activities by reviewing the overview material in the planning section.

Procedure

1. Determine your strategy for identity mapping.
   - If you use a mapping rules file, ensure that the XSLT or Javascript mapping rule has been written to match the requirements of your deployment.
   - If you use a Tivoli Directory Integrator assembly line, ensure that the assembly line has been constructed.
   - When using a custom mapping module, ensure that the module has been written and tested.
2. Ensure that you have established the user agent policy for both the consumer and identity provider.
3. Complete the worksheet for the federation. Complete one of the following:
   - "Identity provider configuration worksheet" on page 338
   - "Consumer configuration worksheet" on page 344

Configuring an OpenID federation

Use the Federation wizard to create and configure an OpenID federation.

Before you begin

Ensure that you have prepared configuration information before using the wizard to create the federation.

About this task

To use the Federation wizard to create and configure an OpenID federation, complete the steps in this procedure:

Procedure

1. Log on to the Integrated Solutions Console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations**. The Current Domain and Federations portlets open.
3. Click **Create**. The Federation wizard starts. The wizard presents a series of configuration panels.

4. Use your completed worksheet to provide values at each panel.
5. Supply the necessary values.
   a. The first series of panels requests settings for the federation name, role, protocol, and point of contact server.
   b. Next, the OpenID configuration panel requests the values needed for an OpenID identity provider or consumer.
   c. The last series of panels requests settings for the identity mapping configuration.

   When you finish entering configuration settings, the Summary panel opens.
6. Click **Next** to proceed to the next panel. If you need to go back to adjust a configuration setting, click **Back**. You can view the online help for information about specific fields.
7. Verify that the configuration settings are correct.
8. Click **Finish**. The Create Federation Complete portlet opens.

Configuring performance improvement for OpenID

Use OpenID parameters to improve performance in white-list scenario.

About this task

Single sign-on scenarios where a relying-party includes an OpenID identity provider in a whitelist can use two parameters to improve performance. The two parameters are:

- `OPENID.DiscoveredInformationExpirationSeconds` - the time for which an OpenID relying party caches discovery information retrieved from an OpenID provider-Identifier. This parameter can save one direct communications round trip for every login after the first login has occurred.

  Discovery is not performed more than once within the expiration period for a given OpenID provider-Identifier.

  In environments where a common well-known OpenID provider is used, this parameter can improve authentication performance.
- `OPENID.SkipClaimedIdDiscovery` - when set to `true`, the relying party does not perform discovery on a claimed identifier returned from an OpenID provider during an OpenID provider-Identifier login. An example scenario is when authentications use `identifier_select`.

  For security reasons, use this parameter only when trusted, white-listed OpenID providers are the only providers which you can use with the relying party.

  Enabling this option saves one direct communications round trip for every log in. The parameter is typically enabled within an intranet environment where a common OpenID provider was used by many remote relying parties.

  The parameter is often used with `OPENID.DiscoveredInformationExpirationSeconds`.

Procedure

1. Edit the `<was_config_root>/itfim/<tfim_domain>/etc/feds.xml` file.
2. Add the configuration parameters to the *Self* configuration parameters for the OpenID relying party federation.
3. Insert text similar to:

```
<fc:EntityProperty name="OPENID.DiscoveredInformationExpirationSeconds">
<fim:Value>604800</fim:Value>
```

```
</fc:EntityProperty>
<fc:EntityProperty name="OPENID.SkipClaimedIdDiscovery">
<fim:Value>true</fim:Value>
</fc:EntityProperty>
```

A good reference for the location to add the configuration parameters is to search for the existing parameter `OPENID.AuthenticationMode` and add the configuration parameters next to that.

Configuring a WebSEAL point of contact server for an Open ID federation

When you plan to use WebSEAL as the point of contact server, you must configure it for the OpenID federation.

Before you begin

The 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.

These instructions assume that the WebSEAL point of contact profile has been activated.

About this task

The Create Federation Complete portlet provides a button that you can use to obtain the Tivoli Federated Identity Manager configuration utility tool. You must obtain the tool and run it. To configure WebSEAL as the point of contact server, complete the steps in this procedure:

Procedure

1. After creating the federation, click **Load configuration changes to Tivoli Federated Identity Manager runtime** to reload your changes.

   Note: The management console gives you the option of adding a partner now, but for this initial configuration of the federation other tasks are completed first.
2. Click **Done** to return to the Federations panel.
3. Click **Download Tivoli Access Manager Configuration Tool**.
4. Save the configuration tool to the file system on the computer that hosts the WebSEAL server.
5. Run the configuration tool from a command line. The syntax is:

   ```
   java -jar /download_dir/tfimcfg.jar -action tamconfig
   -cfgfile webseald-instance_name.conf
   ```

   Note: If Federal Information Processing Standards (FIPS) is enabled in your environment, the secure socket connection factory must be specified. For example:

   ```
   java -jar /download_dir/tfimcfg.jar -action tamconfig
   -cfgfile webseald-instance_name.conf -sslfactory TLS
   ```

You must know the Tivoli Access Manager administration user (default: sec_master) and administration user password. The utility configures endpoints on the WebSEAL server, creates a WebSEAL junction, attaches the appropriate ACLs, and enables the necessary authentication methods.

Example

For example, when you have placed tfimcfg.jar in `/tmp`, and the WebSEAL instance name is `default`, the command (as one continuous line) is:

```
java -jar /tmp/tfimcfg.jar -action tamconfig
-cfgfile /<fully_qualified_path>/webseald-default
```

For more information, see Appendix A, "tfimcfg reference," on page 753.

Configuring WebSphere as a point of contact server

Tivoli Federated Identity Manager is configured by default to use WebSphere Application Server WebSEAL as the point of contact server. To configure WebSphere as your point of contact server, you must make a configuration change.

Procedure

1. Log on to the administration console.
2. Click **Tivoli Federated Identity Manager** > **Manage Configuration** > **Point of Contact**.
3. Select **WebSphere**.
4. Click **Make Active**.

Results

The WebSphere server is now configured to be the point of contact server.

Configuring login pages

As part of configuring a point of contact server, you should configure the information on login pages.

- Consumers must provide a login form for presentation to the end user.

  Administrators who use WebSEAL as a point of contact server can choose to modify the default WebSEAL login.html page.
- Identity providers need to provide discovery information using Yadis or HTML discovery. The discovery information is provided at the OpenID identity URL of the user, or the identity provider identifier URL or both.

Chapter 26. OpenID reference

This section contains a list of references for OpenID. It discusses supported algorithms and transports, and the template pages that are used in a single-sign on flow.

Supported algorithms and transports

Tivoli Federated Identity Manager supports the OpenID specifications for the session type of the shared-secret session (association):

- OpenID 1.1
  - clear text
  - DH-SHA1

  For security reasons the Tivoli Federated Identity Manager service provider (consumer) support of OpenID 1.1 will only request session types of DH-SHA1.
- OpenID 2.0
  - DH-SHA256
  - DH-SHA1
  - no-encryption

  The Tivoli Federated Identity Manager consumer tries DH-SHA256 by default.

  When an identity provider returns an error indicating that a requested session type is unsupported, the identity provider may state which session types are supported. In this case, the Tivoli Federated Identity Manager consumer tries the suggested session type.

  Note: The Tivoli Federated Identity Manager consumer attempts to use no-encryption only when the OpenID server is an SSL endpoint.

The identity provider endpoints that are used by consumers to access OpenID should be configured to use SSL.

In most deployments, non-protected endpoints (for example, HTTP instead of HTTPS) are used for resolution of the identity URL for a user. The following URLs, which are returned as HTML header links, should use SSL:

- openid.server
- openid2.provider

The consumer endpoints should be HTTPS (SSL).

Template page for advertising an OpenID server

The OpenID authentication specifications states that when an identity provider single sign-on URL should return a notification whenever it receives with an HTTP GET request that has no parameters (as specified by the OpenID 1.1 specification). The page to be returned is required to have the following text:

```
This is an OpenID server endpoint. For more information, see http://openid.net/
```

Tivoli Federated Identity Manager provides the file openid_server.html. The file does not have any replaceable macros.

Administrators can use this page without modifications, but in some cases might want to modify the HTML style to match their specific deployment environment.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
  <head>
    <title>OpenID Server</title>
  </head>
  <body>
    This is an OpenID server endpoint. For more information,
see <a href="http://openid.net/">http://openid.net/</a>
  </body>
</html>
```

Figure 32. Template file openid_server.html

This template is used on the identity provider only.

Template page for consent to authenticate

Use the template page for consent to authenticate at the identity provider to determine and store user consent information about authentication permissions to a particular consumer. It is also used to indicate which optional attributes to share with the consumer.

During an OpenID checkid_setup operation, the user is redirected to the Identity Provider to validate they are logged in. At this time, the identity provider asks the user for permission to provide authentication and attribute information to the consuming site. The Tivoli Federated Identity Manager identity provider provides an HTML template page called consent.html.

Tivoli Federated Identity Manager retains knowledge of decisions about whether a user trusts a particular consuming site, in the form of the `trust_root` or `realm`. This saved knowledge enables Tivoli Federated Identity Manager to not have to prompt the user every time a user logs on to the same consumer.

The consent page shows the list of attributes that the single sign-on request (from the consumer) has indicated as *required* or *optional*. Since these lists can be of indeterminate length, the template supports multiple copies of stanzas, repeated once for each attribute in either list. The support for repeated stanzas is provided through the simple registration extension specification.

Administrators can use this page without modifications, but in some cases might want to modify the HTML style to match their specific deployment environment.

This template file provides several replacement macros:

@OPENID_TRUSTURL@
: This macro is replaced with the `openid.trust_root` parameter in the checkid_setup request.

@OPENID_POLICYURL@
: This macro is replaced with the `openid.sreg.policy_url` parameter in the checkid_setup request when the URL exists. When the URL does not exists, the value is an empty string.

@OPENID_IDENTITYURL@
This macro is replaced with the `openid.identity` parameter in the checkid_setup request.

@OPENID_SSOURL@
This macro is replaced with the endpoint of the OpenID server delegate (endpoint) on the identity provider. This value is used for the FORM action parameter to post the results of the consent form back to the OpenID server.

@OPENID_RETURN_TO_VALIDATED@
This macro is replaced with true or false to notify the user if return_to URL validation has been performed as part of Relying-Party discovery.

@REQUIRED_ATTRIBUTE@
A multi-valued macro that belongs inside a `[RPT requiredAttrs]` repeatable replacement list. The values show the list of required attributes from the service provider, as specified for the simple registration extension. This macro is replaced for each value contained within the `openid.sreg.required` parameter in the request with the string `openid.sreg.` prepended.

@OPTIONAL_ATTRIBUTE@
A multi-valued macro that belongs inside a `[RPT optionalAttrs]` repeatable replacement list. The values show the list of optional attributes from the service provider, as specified for the simple registration extension. This macro is replaced for each value contained within the `openid.sreg.optional` parameter in the request with the string `openid.sreg.` prepended.

Optional attributes require special consideration. The identity provider allows users to specify individually which of the optional attributes they can send to a specified consumer. The user preferences are denoted by the true or false parameters for each optional attribute, as specified in the form contained in the HTML page for consent-to-authenticate. To enable this feature, the parameter name *must* begin with the prefix `optattr_` and end with the full name of the optional attribute.

For example:

```
optattr_openid.sreg.email=true&optattr_openid.sreg.nickname=false
```

The following figure shows an example of the handling of optional attributes.

```
 The following optional attributes have been requested. Please select
 which attributes you are prepared to send, or select
 "All Optional Attributes":<br /><br />
          <input id="chk_all_optional_attributes" type="checkbox"
           checked="checked" name="all_optional_attributes"
           onClick="allOptionalAttributes()" />
          <label for="chk_all_optional_attributes">All Optional Attributes
          </label><br /><br />
      [RPT optionalAttrs]
               <input id="chk_@OPTIONAL_ATTRIBUTE@" type="checkbox"
name="optattr_@OPTIONAL_ATTRIBUTE@" onClick="oneOptionalAttribute()" />
               <label for="chk_@OPTIONAL_ATTRIBUTE">@OPTIONAL_ATTRIBUTE@
               </label><br />
      [ERPT optionalAttrs]
```

Figure 33. Handling consent of individual optional attributes

Note: The check box input parameter in the form builds the name using the `optattr_` prefix, and the name of the optional attribute. For each optional attribute in the request from the service provider, the code processing this form at the identity provider looks for a parameter like `optattr_<attributename>`. The identity provider then treats the value as `true` or `false`. A value of true indicates consent of the optional attribute. When a parameter does not exist in the posted form, consent is false.

One possible deployment scenario might be the development of a *persona portal* for individual users. Users can have different personas created when they use a *persona portal*. Each persona can have different attribute sets managed in an external data store. This function enables the user to associate a particular persona with a particular OpenID consumer. Doing so gives user the flexibility to select persona attributes when the user logs in to the specified consumer.

For example, you can use the identity provider to dynamically create, name, and populate sets of attributes for each persona.

This scenario is supported by the use of an optional FORM parameter called `userdata`. The `userdata` can be a menu list that where the user can select the persona from which the attributes are populated.

When `userdata` is included in the input form, its URL-encoded string value is included in the claims sent to the security token service during identity mapping.

The following code sample shows the example HTML template file consent.html.

This template is used on the identity provider only.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <title>OpenID Consent-to-Authenticate</title>
<script type="text/javascript">

// when "All optional attributes" is selected,
uncheck any checked individual optional attributes
function allOptionalAttributes() {
    var theForm = document.forms[0];
    for (i = 0; i < theForm.elements.length; i++) {
        if (theForm.elements[i].type == "checkbox") {
```

```
            var cbName = theForm.elements[i].name;
            if (cbName.indexOf("optattr_") == 0) {
                theForm.elements[i].checked = false;
            }
        }
    }
}

// when an individual optional attribute is selected, be sure to
uncheck "All optional attributes"
function oneOptionalAttribute() {
    document.forms[0].all_optional_attributes.checked = false;
}

// utility function to show a section
function showDiv(f) {
  if (f.style) {
    f.style.display='block';
  }
}

</script>
  </head>
  <body>
    This consuming site has asked for an OpenID login from you:
 <b>@OPENID_TRUSTURL@</b>
    <p />
    The consuming site's policy can be found at: <b>@OPENID_POLICYURL@</b>
    <p />


<script type="text/javascript">
    //
    // RP-discovery information
    //
    var txtWarningReturnTo = "WARNING: The return_to URL for the site has not
been successfully validated using relying-party discovery";
    var returntoValidated = @OPENID_RETURN_TO_VALIDATED@;
    if (!returntoValidated) {
        document.write(txtWarningReturnTo);
    }
</script>


    <p />
    Your identity URL is: <b>@OPENID_IDENTITYURL@</b>
<script type="text/javascript">
    //
    // Display claimed identifier if different from identity URL
(e.g. if delegation was being used)
    //
    var txtClaimedID = "Your claimed identifier is: ";
    var identityurl = "@OPENID_IDENTITYURL@";
    var claimedid = "@OPENID_CLAIMEDID@";
    if (claimedid != identityurl) {
        document.write("<p />");
        document.write(txtClaimedID);
        document.write("<b>");
        document.write(claimedid);
        document.write("</b>");
    }
</script>


    <p />
<script type="text/javascript">
    //
```

```
    // PAPE information
    //
    var txtMaxAuthnAge = "Requested Maximum Authentication Age (seconds): ";
    var txtRequestedAuthnPolicies = "Requested Authentication Policies";
    var txtRequestedAssuranceLevels = "Requested Assurance Levels";

    var nopii = false;
    var maxAuthenticationAge = @MAXIMUM_AUTHENTICATION_AGE@;
    if ( maxAuthenticationAge >= 0) {
        document.write("<p/>" + txtMaxAuthnAge + maxAuthenticationAge);
    }

    var strAuthPolicies = "";
    [RPT authenticationPolicies]
        strAuthPolicies += "@REQUESTED_AUTHENTICATION_POLICY@"+",";
    [ERPT authenticationPolicies]
    if (strAuthPolicies.length > 0) {
        // strip last comma and split into array
        strAuthPolicies =
strAuthPolicies.substring(0,strAuthPolicies.lastIndexOf(","));
        var authPolicies = strAuthPolicies.split(",");
        document.write("<p/>");
        document.write("<table border>");
        document.write("<tr><th>" + txtRequestedAuthnPolicies + "</th></tr>");
        for (var i = 0; i < authPolicies.length; i++) {
            document.write("<tr><td>"+authPolicies[i]+"</td></tr>");

            // check if this is the nopii policy
            if (authPolicies[i] ==
"http://www.idmanagement.gov/schema/2009/05/icam/no-pii.pdf") {
                nopii = true;
            }
        }
        document.write("</table>");
    }

    var strAssuranceLevels = "";
    [RPT assuranceLevels]
        strAssuranceLevels += "@REQUESTED_ASSURANCE_LEVEL@"+",";
    [ERPT assuranceLevels]

    if (strAssuranceLevels.length > 0) {
        // strip last comma and split into array
        strAssuranceLevels =
strAssuranceLevels.substring(0,strAssuranceLevels.lastIndexOf(","));
        var assuranceLevels = strAssuranceLevels.split(",");
        document.write("<p/>");
        document.write("<table border>");
        document.write("<tr><th>" + txtRequestedAssuranceLevels + "</th></tr>");
        for (var i = 0; i < assuranceLevels.length; i++) {
            document.write("<tr><td>"+assuranceLevels[i]+"</td></tr>");
        }
        document.write("</table>");
    }

</script>


    <form action="@OPENID_SSOURL@" method="post">
      <input type="hidden" name="openid.mode" value="consent_to_authenticate" />
      <div id="DIV_ATTRIBUTES" name="DIV_ATTRIBUTES" style="display: none;">
        The following required attributes have been requested:<br />
        <ul>
        [RPT requiredAttrs]
            <li>@REQUIRED_ATTRIBUTE@</li>
        [ERPT requiredAttrs]
        </ul>
```

```
        <p />
        The following optional attributes have been requested. Please select
which attributes you are prepared to send, or select
"All Optional Attributes":<br /><br />
            <input id="chk_all_optional_attributes" type="checkbox"
checked="checked" name="all_optional_attributes"
onClick="allOptionalAttributes()" />
            <label for="chk_all_optional_attributes">
All Optional Attributes</label><br /><br />
        [RPT optionalAttrs]
                <input id="chk_@OPTIONAL_ATTRIBUTE@" type="checkbox"
name="optattr_@OPTIONAL_ATTRIBUTE@" onClick="oneOptionalAttribute()" />
                <label for="chk_@OPTIONAL_ATTRIBUTE@"
>@OPTIONAL_ATTRIBUTE@</label><br />
        [ERPT optionalAttrs]
      </div>
      <p />
      Do you wish to authenticate to this site, sending all
required attributes and
the selected optional attributes?
      <div>
        <input id="rd_permit_forever" type="radio"
name="consent" value="permit_forever"
checked="checked" /><label for="rd_permit_forever">
Allow Authentication forever
(add to my trusted sites)</label><br/>
        <input id="rd_permit_once" type="radio"
name="consent" value="permit_once" />
<label for="rd_permit_once">Allow Authentication this time only</label><br />
        <input id="rd_deny_once" type="radio" name="consent"
value="deny_once" />
<label for="rd_deny_once">Do not authenticate to this
site this time only</label><br />
        <input id="rd_deny_forever" type="radio" name="consent"
value="deny_forever" />
<label for="rd_deny_forever">Do not ever authenticate to this site
(add to my untrusted sites)</label><br />
      </div>
      <p /><label for="tx_userdata">User data or persona information:</label>
<input id="tx_userdata" type="text" name="userdata" />
      <p /><input type="submit" name="submit" value="Submit" />
    </form>
<script type="text/javascript">
  //
  // if the nopii policy was requested, leave the attribute information hidden
(as we shouldn't send it), otherwise show it
  //
  if (!nopii) {
    showDiv(document.getElementById("DIV_ATTRIBUTES"));
  }
</script>

  </body>
</html>
```

Template HTML page for trusted site management

This page is used at the identity provider. The HTML page is used to manage the persisted set of trusted or untrusted sites. The user establishes the sites through the consent.html page during single sign-on operations.

OpenID identity provider functionality includes the ability to store and retrieve certain user preference attributes including:

- Whether or not a particular consumer site, as identified by `trust_root` value, is trusted. The trust values can be `once`, `never`, or `always`.
- The list of optional attributes that can be sent to a particular trusted consumer.
- Any optional user preference data that the identity provider might choose to use when building an attribute set for a single sign-on request to a consumer. For example, the optional detail could include a persona index.

  Tivoli Federated Identity Manager provides a mechanism that stores the attributes in persistent cookies on the browser.

The Tivoli Federated Identity Manager server includes a page template and supporting code. The page template and supporting code use the interface for storing and retrieving information about trusted consumers. Users can use the page template to display and manage this list.

The template file is sitemanager.html.

Administrators can use this page without modifications, but in some cases might want to modify the HTML style to match their specific deployment environment.

The template has the following replacement macros:

@USERNAME@
This macro is replaced with the Tivoli Federated Identity Manager user name.

@SITE_NAME@
This macro is a multi-valued and is used inside either a `[RPT trustedSites]` or `[RPT untrustedSites]` repeatable replacement list. The macro is used to display information about sites that are configured for one of the following states:

- trusted forever
- denied forever

This macro displays the `trust_root` URL for the trusted or untrusted site.

@REQUIRED_ATTRIBUTES@
This macro is multi-valued and is used inside a `[RPT trustedSites]` repeatable replacement list. The macro is used to display a comma-separated list of the specific set of required attributes that the user must send to the consumer.

@OPENID_SITEMANAGERURL@
This macro is replaced with the URL endpoint of the site manager delegate which is used to process the `remove` action on trusted sites.

@ALL_OPTIONAL_ATTRIBUTES@
This macro is multi-valued and is used inside a `[RPT trustedSites]` repeatable replacement list to for the trusted site. The macro us used to indicate if the user is prepared to send all requested optional attributes to that consumer. Supported values are `true` or `false`.

@LISTED_OPTIONAL_ATTRIBUTES@
This macro is multi-valued and is used inside a `[RPT trustedSites]` repeatable replacement list. The macro is used to display a comma-separated list of the specific set of optional attributes the user is prepared to send to that consumer. This value is a non-empty string when `@ALL_OPTIONAL_ATTRIBUTES@` is `false` for the trusted site. When `@ALL_OPTIONAL_ATTRIBUTES@` is `true`, this value is an empty string.

@USERDATA@

This macro is multi-valued and is used inside a [RPT trustedSites] repeatable replacement list. The macro is used to display optional user data. The data can be specified by a user when the user processed the consent-to-authenticate page, as part of choosing to permanently trust the site. When no user data is specified, the value of the macro is the empty string

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8"/>
  <title>OpenID Site Manager</title>
 </head>
 <body>
  OpenID Site Manager</titleb>@USERNAME@</b>
  <p/>
  Trusted Sites<br/>
  <table border="1">
   <tr><td>Site</td><td>Required Attributes</td><td>All Optional
Attributes?</td><td>Permitted Optional Attributes</td><td>User
Data</td><td>Action</td></tr>
[RPT trustedSites]
   <tr>
    <td>@SITE_NAME@</td>
    <td>@REQUIRED_ATTRIBUTES@</td>
    <td>@ALL_OPTIONAL_ATTRIBUTES@</td>
    <td>@LISTED_OPTIONAL_ATTRIBUTES@</td>
    <td>@USERDATA@</td>
    <td><a href="@OPENID_SITEMANAGERURL@?action=
remove&site=@SITE_NAME@">Remove</a></td>
   </tr>
 [ERPT trustedSites]
 </table>
 <p/>
 Untrusted Sites<br/>
 <table border="1">
 <tr><td>Site</td><td>Action</td></tr>
 [RPT untrustedSites]
 <tr>
  <td>@SITE_NAME@</td>
  <td>@OPENID_SITEMANAGERURL@?
action=remove&site=@SITE_NAME@">Remove</a></td>
 </tr>
[ERPT untrustedSites]
 </table>
</body>
</html>
```

Figure 34. Template HTML file sitemanager.html

This template is used on the identity provider only.

Template page for OpenID error

Tivoli Federated Identity Manager uses a generic error page template to show the detailed error text information under the following circumstances:

- An error halts processing on the identity provider or the consumer.
- The error is not returned.

For example:

- On an identity provider this page is used when processing the trusted sites page or when a single sign-on request lacks a valid `return_to` URL.
- On a consumer, this page is used when bad parameters are returned in the login page.

The template page is error.html.

Administrators can use this page without modifications, but in some cases might want to modify the HTML style to match their specific deployment environment.

The following replacement macros are supported:

@REQ_ADDR@
: This macro is replaced with the URL of the currently called delegate endpoint.

@TIMESTAMP@
: This macro is replaced with the current time in UTC.

@DETAIL@
: This macro is replaced with the native language support (NLS) text of the error message associated with the error.

@EXCEPTION_STACK@
: This macro is replaced with the stack trace of any exception that caused the error.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
    <head>
        <title>An OpenID error has occurred</title>
    </head>
    <body style="background-color:#ffffff">
        <div>
            <h2 style="color:#ff8800">An error has occurred</h2>
            <div id="infoDiv" style="background-color:#ffffff;color:#000000">
                <em>@REQ_ADDR@</em> <br />
                <em>@TIMESTAMP@</em> <br />
            </div>
            <br />
<div id="detailDiv" style="background-color:#999999; border-style:solid;
border-width:1px; border-color:#000000">
                <h4>Error details</h4>
                @DETAIL@
            </div>
            <br />
            <div id="stackDiv" style="background-color:#999999;
border-style:solid; border-width:1px; border-color:#000000">
                <h4>Stack trace</h4>
                @EXCEPTION_STACK@
            </div>
        </div>
    </body>
</html>
```

Figure 35. Template HTML file error.html

This template is used on both the identity provider and consumer.

Template page for OpenID 2.0 indirect post

OpenID 2.0 specifies that HTTP POST requests can be used instead of HTTP redirects, to send indirect messages between the identity provider and relying party (consumer). The messages are sent to the browser and then redirected to the target.

Tivoli Federated Identity Manager automatically switches messages into a self-posting FORM using HTTP POST (rather than a 302 redirect) when the following conditions are true:

- OpenID 2.0 is being used
- The message size exceeds 2K bytes

When POST is used, a page is loaded with a self-posting FORM (rather than a 302 redirect) containing the same parameters that would otherwise have been passed on the query string.

The template file is indirect_post.html.

Administrators can use this page without modifications, but in some cases might want to modify the HTML style to match their specific deployment environment.

The file supports the following replacement macros:

@OPENID_PARTNER_URL@
This macro is replaced with the URL of the destination partner. This is used for the FORM action parameter.

@PARAM_NAME@ / @PARAM_VALUE@
These are multi-valued macros that are used inside a [RPT formFields] repeatable replacement list. They are used for parameters to pass to the recipient.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <title>OpenID Message</title>
  </head>
  <body>
    <form method="post" name="openid_message" action="@OPENID_PARTNER_URL@">
      [RPT formFields]
        <input type="hidden" name="@PARAM_NAME@" value="@PARAM_VALUE@" />
      [ERPT formFields]
      <noscript>
      <button type="submit">Send OpenID Message</button>
<!-- included for requestors that do not support javascript -->
      </noscript>
    </form>
    <script type="text/javascript">
        var signOnText = 'Sending OpenID message...';
        document.write(signOnText);
        setTimeout('document.forms[0].submit()', 0);
    </script>
  </body>
</html>
```

Figure 36. Template file indirect_post.html

This template is used on the identity provider only.

Template page returned for checkid_immediate

When a checkid_immediate request is initiated by the Tivoli Federated Identity Manager service provider, and the identity provider returns a status that it cannot determine whether the user owns the URL, the identity provider also returns one of the following attributes:

- openid.user_setup_url
  For OpenID 1.1
- openid.mode=user_setup_needed
  For OpenID 2.0

When the Tivoli Federated Identity Manager consumer receives this type of reply, it returns a page template file.

The page template file is immediate.html.

Administrators can use this page without modifications, but in some cases might want to modify the HTML style to match their specific deployment environment.

The file has the following replacement macro:

@OPENID_USER_SETUP_URL@
This macro is replaced with the URL returned in the openid.user_setup_url parameter of an identity provider response to the checkid_immediate request. When the request is an OpenID 2.0 request, this parameter can be the empty string.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
  <head>
    <title>Results from checkid_immediate</title>
  </head>
  <body>
    <script type="text/javascript">
        var setup_url = "@OPENID_USER_SETUP_URL@";
        if (setup_url) {
            document.write('<a href="');
            document.write(setup_url);
            document.write('">Please click here to complete identity
            provider requirements</a>');
        } else {
            document.write('Unable to proceed as authentication is required at
        the OpenID Identity Provider.');
        }
    </script>
  </body>
</html>
```

Figure 37. Template page immediate.html

This template is used by the consumer only.

Template page returned for server error

The identity provider server returns `openid.mode` set to `error` and specifies the error text in `openid.error` under the following circumstances:

- When the Tivoli Federated Identity Manager consumer sends a checkid_immediate or checkid_setup request
- When the checkid_immediate or checkid_setup request results in an error

In this case, the consumer returns the server_error.html page.

Administrators can use this page without modifications, but in some cases might want to modify the HTML style to match their specific deployment environment.

The template page supports the following replacement macros:

@OPENID_SERVER@
This macro is replaced with the OpenID server URL that the consumer was communicating with when the error occurred.

@OPENID_ERROR@
The text from `openid.error`.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
  <head>
    <title>OpenID Error Returned By Server</title>
  </head>
  <body>
    The OpenID Server: @OPENID_SERVER@ has returned
    the following error text:<br />
    @OPENID_ERROR@
  </body>
</html>
```

Figure 38. Template file server_error.html

This template is used on the consumer only.

Chapter 27. OAuth planning overview

Tivoli Federated Identity Manager supports OAuth 1.0 and OAuth 2.0 protocols. The implementation of OAuth in Tivoli Federated Identity Manager strictly follows the OAuth standards.

Note: Every mention of the OAuth 1.0 protocol in the guide refers to the RFC5849 version.

You must be familiar with the OAuth specification before implementing a single sign-on federation. You must be prepared with the following requirements for your OAuth 2.0 implementation:

- Information you are required to provide to your business partners
- Information your partner must provide to you

OAuth Concepts

This topic introduces the main concepts of OAuth 1.0 and OAuth 2.0.

OAuth is an HTTP-based authorization protocol. It gives third-party applications scoped access to a protected resource on behalf of the resource owner. It gives scoped access by creating an approval interaction between the resource owner, client, and the resource server. It gives users the ability to share their private resources between sites without providing user names and passwords. Private resources can be anything, but common examples include photos, videos, contact lists, and so on.

For a complete description of the OAuth specifications, see the OAuth website: http://www.oauth.net.

The following concepts are common for both OAuth 1.0 and OAuth 2.0.

Resource owner
An entity capable of authorizing access to a protected resource. When the resource owner is a person, it is called an *end user*.

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 after the resource owner grants it authorization. OAuth 2.0 introduces two types of clients: confidential and public. Confidential clients are registered with a client secret, while public clients are not.

OAuth server
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. The server issues an access token to the OAuth client after it successfully does the following actions:

- Authenticates the resource owner.
- Validates a request or an authorization grant.
- Obtains resource owner authorization.

An authorization server can also be the resource server. Tivoli Federated Identity Manager takes the role of these two servers.

Access token
A string that represents authorization granted to the OAuth client by the resource owner. This string represents specific scopes and durations of access. It is granted by the resource owner and enforced by the OAuth server.

Protected resource
A restricted resource that can be accessed from the OAuth server using authenticated requests.

Additional concepts are introduced for the OAuth 2.0 protocol. These new concepts are as follows:

Resource server
The server that hosts the protected resources. It can use access tokens to accept and respond to protected resource requests. The resource server might be the same server as the authorization server.

Authorization grant
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 code
A code that the Authorization server generates when the resource owner authorizes a request.

Refresh token
A string that is used to obtain a new access token.

A refresh token is optionally issued by the authorization server to the OAuth client together with an access token. The OAuth client can use the refresh token to request another access token based on the same authorization, without involving the resource owner again.

OAuth endpoints

Endpoints provide OAuth clients the ability to communicate with the OAuth server or authorization server within a federation.

All endpoints can be accessed through URLs. The syntax of the URLs is specific to the purpose of the access.

If you are responsible for installing, configuring, or maintaining a federation in Tivoli Federated Identity Manager, you might find it helpful to be familiar with these endpoints and URLs.

OAuth 1.0 federations

The OAuth 1.0 federation naming follows the standard Tivoli Federated Identity Manager naming convention for a unique identifier or `protocolID`. The syntax is:

`https://<hostname:port>/FIM/sps/<federation_name>/oauth10`

For example:

`https://server.oauth.com/FIM/sps/MySocialNetwork/oauth10`

The following table describes the endpoints that are used in an OAuth 1.0 federation.

Table 111. OAuth 1.0 endpoint definitions and URLs

Endpoint name	Description	Example
Clients manager endpoint	A URL for resource owners to manage their trusted clients. The resource owner can use the clients manager endpoint to access and modify the list of clients that have been authorized to access the protected resource. The trusted clients manager shows the client name and permitted scope of an authorized client. **Note:** The list does not show clients that have been disabled or deleted from the federation. The resource owner can optionally remove trusted client information from the list. In doing so, the resource owner is prompted for consent to authorize the next time the OAuth client attempts to access the protected resource.	`https://server.oauth.com/FIM/sps/MySocialNetwork/oauth10/clients`
Temporary credential request endpoint	A request URL that the OAuth client uses to obtain a set of temporary credentials.	`https://server.oauth.com/FIM/sps/MySocialNetwork/oauth10/request`
Resource owner authorization endpoint	An authorization URL where the resource owner grants authorization to the OAuth client to access the protected resource.	`https://server.oauth.com/FIM/sps/MySocialNetwork/oauth10/authorize`
Token request endpoint	An access URL where the OAuth client exchanges the set of temporary credentials and verification code for a set of token credentials.	`https://server.oauth.com/FIM/sps/MySocialNetwork/oauth10/access`

OAuth 2.0 federations

The OAuth 2.0 federation naming follows the standard Tivoli Federated Identity Manager naming convention for a unique identifier or `protocolID`. The syntax is:

`https://<hostname:port>/FIM/sps/<federation_name>/oauth20`

For example:

`https://server.oauth.com/FIM/sps/MySocialNetwork/oauth20`

The following table describes the endpoints that are used in an OAuth 2.0 federation.

Note: Not all authorization grant types use all three endpoints in a single OAuth 2.0 flow.

Table 112. OAuth 2.0 endpoint definitions and URLs

Endpoint name	Description	Example
Clients manager endpoint	A URL for resource owners to manage their trusted clients. The resource owner can use the clients manager endpoint to access and modify the list of clients that have been authorized to access the protected resource. The trusted clients manager shows the client name and permitted scope of an authorized client. **Note:** The list does not show clients that have been disabled or deleted from the federation. The resource owner can optionally remove trusted client information from the list. In doing so, the resource owner is prompted for consent to authorize the next time the OAuth client attempts to access the protected resource.	`https://server.oauth.com/FIM/sps/MySocialNetwork/oauth20/clients`
Authorization endpoint	An authorization URL where the resource owner grants authorization to the OAuth client to access the protected resource.	`https://server.oauth.com/FIM/sps/MySocialNetwork/oauth20/authorize`
Token endpoint	A token request URL where the OAuth client exchanges an authorization grant for an access token and an optional refresh token.	`https://server.oauth.com/FIM/sps/MySocialNetwork/oauth20/token`

OAuth 1.0 workflow

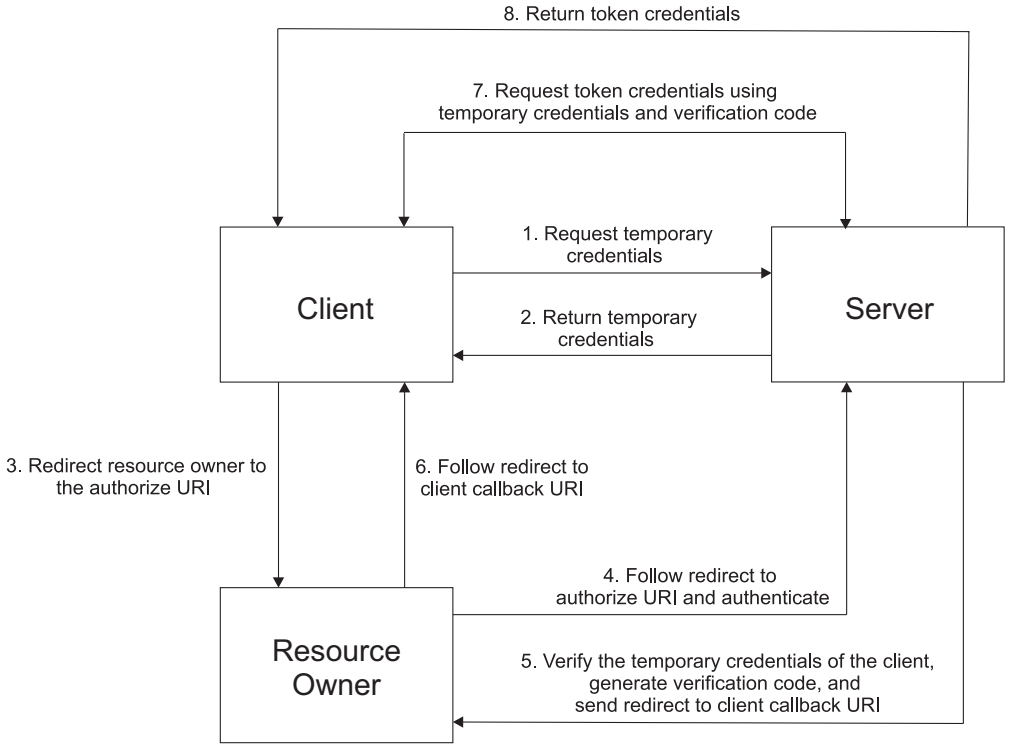
The RFC5849 version of OAuth 1.0, or Open Authorization, is an HTTP-based authorization protocol. OAuth 1.0 support makes it possible for users to share their private resources between sites without providing users and passwords. Private resources can be anything, but common examples include photos, videos, and contact lists.

The OAuth 1.0 function of the Tivoli Federated Identity Manager can be configured through the following methods:

- Tivoli Federated Identity Manager console
- Command-line interface

OAuth 1.0 workflow

An OAuth server issues tokens to OAuth clients. OAuth clients can access resources on behalf of the resource owner using tokens that have scope, lifetimes, and other attributes.



The OAuth 1.0 protocol runtime workflow diagram involves the following steps:

1. The OAuth client requests a set of temporary credentials from the OAuth server to start the authentication process. Temporary credentials distinguish individual OAuth client requests to the OAuth server.
2. The OAuth server validates the request and returns a set of temporary credentials to the OAuth client.
3. The OAuth client redirects the resource owner to the authorized URI to obtain the approval to access the protected resource.
4. The resource owner authenticates with the OAuth server using its client credentials and authorizes the request from the OAuth client.
5. The OAuth server validates the temporary credentials and after the resource owner authorizes the OAuth client, a verification code is generated.
6. The resource owner is redirected to the callback URI provided by the OAuth client in the previous request.
7. The OAuth client requests the access token using the temporary credentials and verification code.
8. The OAuth server validates the request and returns an access token to the OAuth client to access the protected resource.

About two-legged OAuth

Use two-legged OAuth to implement a delegation of authority in the client.

Two-legged OAuth is also called a *Signed Fetch*. In the two-legged OAuth scenario, the OAuth client uses the client secret to sign the request and directly access the protected resource. The OAuth server trusts the OAuth client to provide data without asking the resource owner for authorization.

Security Token Service interface for two-legged OAuth flow

The Security Token Service interface processes requests differently for a two-legged OAuth flow.

This section provides the following information about the two-legged OAuth scenario:

- The behavior of Tivoli Federated Identity Manager Security Token Service.
- The significance of the WebSphere Trust Association Interceptor enforcement point.

When an OAuth client accesses a protected resource that uses two-legged OAuth, no OAuth token or authorizing resource owner is associated with the request. The OAuth client signs the request with its client credentials, proving that the request came from that client. This method is similar to a traditional basic authentication, except that the request is digitally signed rather than containing the client credentials in clear text.

The two-legged OAuth flow follows this process:

1. An OAuth enforcement point receives a two-legged OAuth request.
2. The enforcement point contacts the Security Token Service for validation.
3. The Security Token Service then processes the two-legged OAuth request sent from the enforcement point, and validates the request signature.
4. The Security Token Service returns a `username` response attribute as part of the validation response. This attribute is set to the value of the client identifier, which is also known as the client key.

   Note: This method is different from a traditional three-legged OAuth flow, where the `username` attribute is set to the user name of the resource owner. The user is the same entity who authorized the access of the OAuth client to the protected resource.

The OAuth enforcement point can use the returned `username` response attribute for audit, authentication, or downstream protected resource application.

Tivoli Federated Identity Manager provides sample OAuth mapping rules that detect a two-legged OAuth scenario. When detected, the mapping rule changes the value of the returned `username` attribute from the client identifier to the predefined value `me_guest`.

You can modify the mapping rule to either leave the `username` attribute as the client identifier or map it to something else. This change becomes important only when your enforcement point or downstream protected resource application rely on the value of the returned `username` attribute for special processing.

The Trust Association Interceptor enforcement point requires and relies on the value of the returned `username` attribute. The enforcement point performs a WebSphere authentication as the `username` that is returned from the Security Token Service. The WebSphere Application Server user registry must contain a user that matches the `username` attribute returned by the custom mapping rule in the Security Token Service.

If you are using the example mapping rule as is, you must create a user in the WebSphere user registry called `me_guest`. This step is only needed for two-legged OAuth with the Trust Association Interceptor enforcement point.

OAuth 2.0 workflow

The OAuth 2.0 support in Tivoli Federated Identity Manager provides four different ways for an OAuth client to obtain access the protected resource.

The OAuth 2.0 function of the Tivoli Federated Identity Manager can be configured through the following methods:

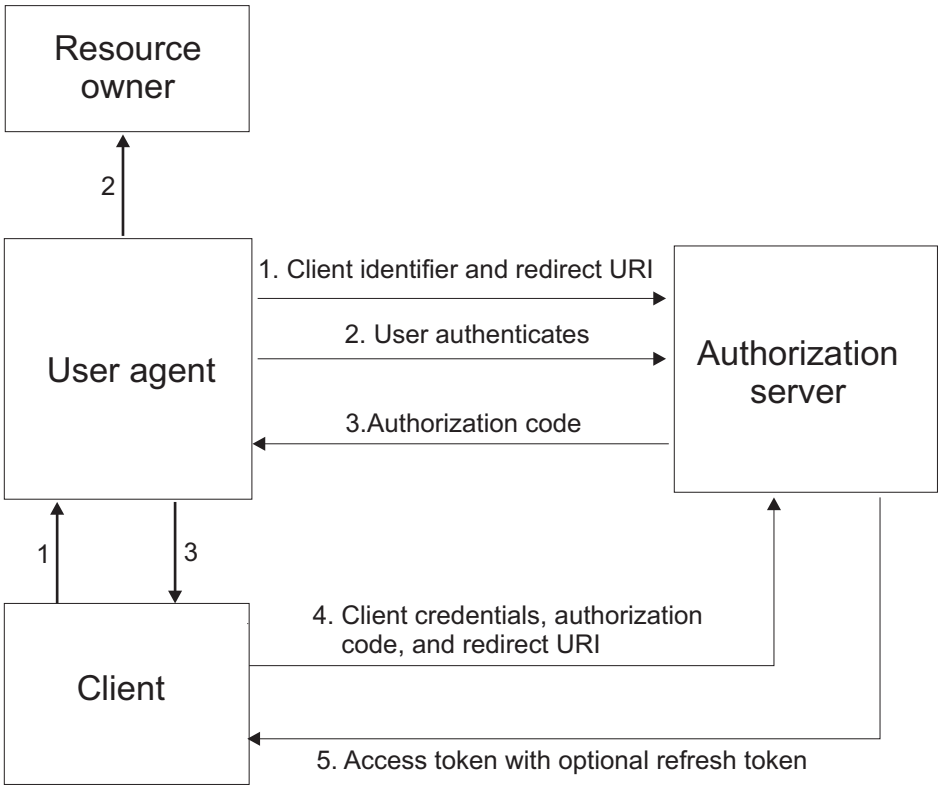
- Tivoli Federated Identity Manager console
- Command-line interface

OAuth 2.0 workflow

Tivoli Federated Identity Manager supports the following OAuth 2.0 workflows.

Authorization code flow

The authorization code grant type is suitable for OAuth clients that can keep their client credentials confidential when authenticating with the authorization server. For example, a client implemented on a secure server. As a redirection-based flow, the OAuth client must be able to interact with the user agent of the resource owner. It also must be able to receive incoming requests through redirection from the authorization server.



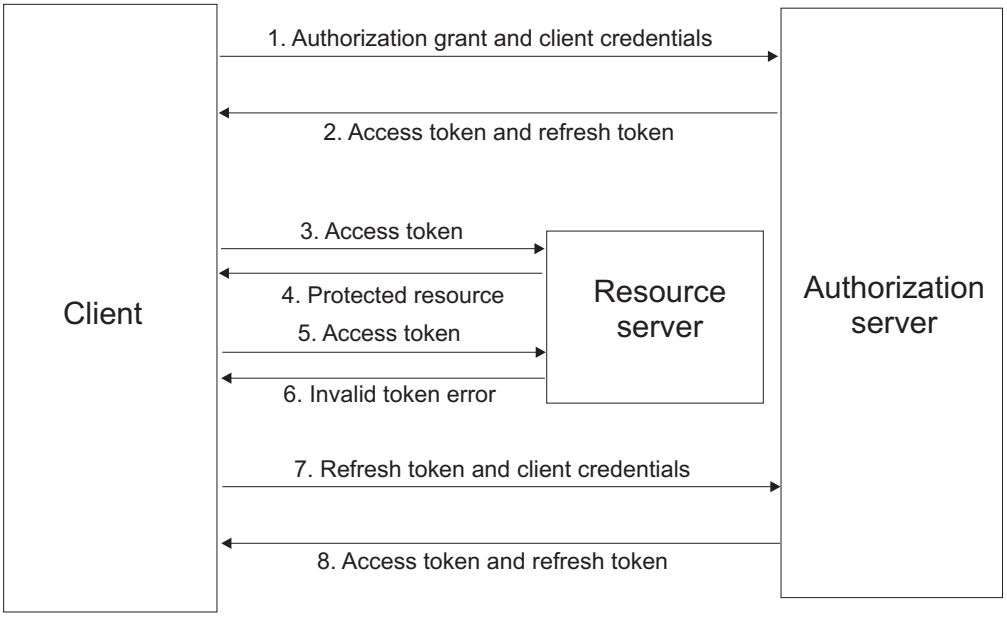
The authorization code workflow diagram involves the following steps:

1. The OAuth client initiates the flow when it directs the user agent of the resource owner to the authorization endpoint. The OAuth client includes its client identifier, requested scope, local state, and a redirection URI. The authorization server sends the user agent back to the redirection URI after access is granted or denied.

2. The authorization server authenticates the resource owner through the user agent and establishes whether the resource owner grants or denies the access request.
3. If the resource owner grants access, the OAuth client uses the redirection URI provided earlier to redirect the user agent back to the OAuth client. The redirection URI includes an authorization code and any local state previously provided by the OAuth client.
4. The OAuth client requests an access token from the authorization server through the token endpoint. The OAuth client authenticates with its client credentials and includes the authorization code received in the previous step. The OAuth client also includes the redirection URI used to obtain the authorization code for verification.
5. The authorization server validates the client credentials and the authorization code. The server also ensures that the redirection URI received matches the URI used to redirect the client in Step 3. If valid, the authorization server responds back with an access token.

The authorization server can be the same server as the resource server or a separate entity. A single authorization server can issue access tokens accepted by multiple resource servers.

Authorization code flow with refresh token



The authorization code workflow with refresh token diagram involves the following steps:

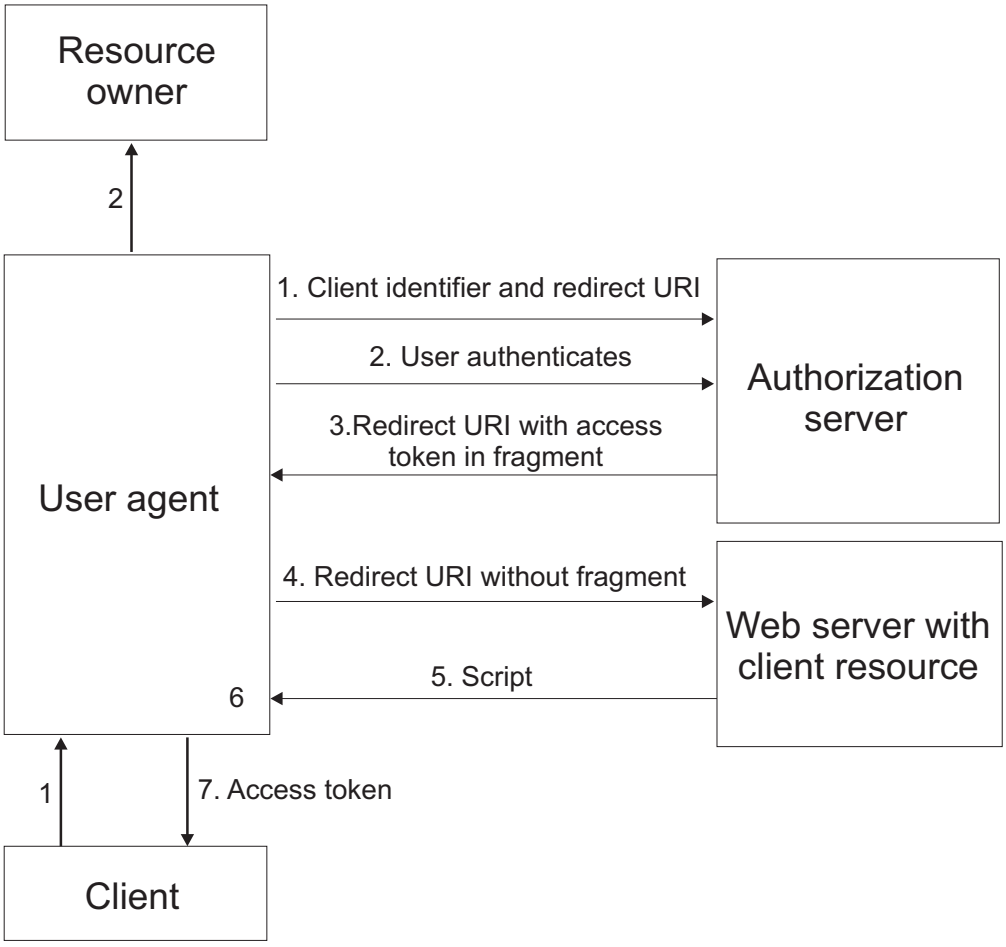
1. The OAuth client requests an access token by authenticating with the authorization server with its client credentials, and presenting an authorization grant.
2. The authorization server validates the client credentials and the authorization grant. If valid, the authorization server issues an access token and a refresh token.
3. The OAuth client makes a protected resource request to the resource server by presenting the access token.

4. The resource server validates the access token. If the access token is valid, the resource owner serves the request.
5. Repeat steps 3 and 4 until the access token expires. If the OAuth client knows that the access token has expired, skip to Step 7. Otherwise, the OAuth client makes another protected resource request.
6. If access token is not valid, the resource server returns an error.
7. The OAuth client requests a new access token by authenticating with the authorization server with its client credentials, and presenting the refresh token.
8. The authorization server validates the client credentials and the refresh token, and if valid, issues a new access token and a new refresh token.

Implicit grant flow

The implicit grant type is suitable for clients that are not capable of maintaining their client credentials confidential for authenticating with the authorization server. An example can be in the form of client applications that are in a user agent, typically implemented in a browser using a scripting language such as JavaScript.

As a redirection-based flow, the OAuth client must be able to interact with the user agent of the resource owner, typically a web browser. The OAuth client must also be able to receive incoming requests through redirection from the authorization server.



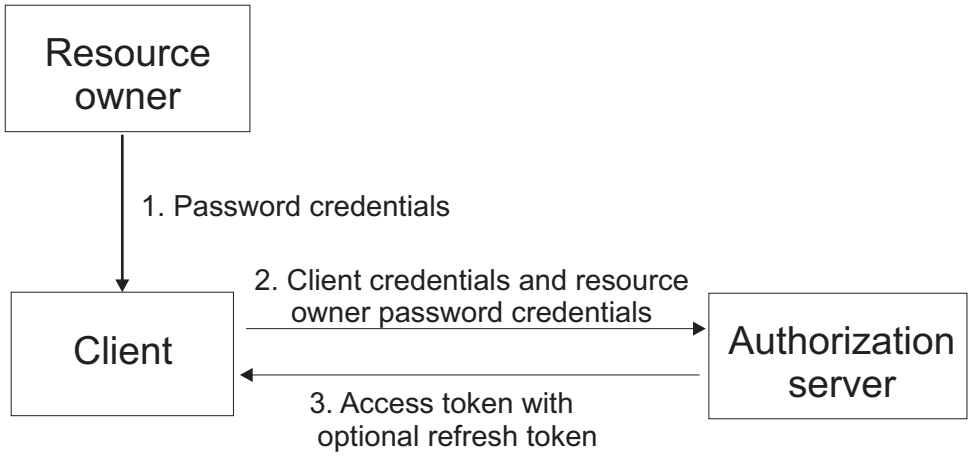
The implicit grant workflow diagram involves the following steps:

1. The OAuth client initiates the flow by directing the user agent of the resource owner to the authorization endpoint. The OAuth client includes its client identifier, requested scope, local state, and a redirection URI. The authorization server sends the user agent back to the redirection URI after access is granted or denied.
2. The authorization server authenticates the resource owner through the user agent and establishes whether the resource owner grants or denies the access request.
3. If the resource owner grants access, the authorization server redirects the user agent back to the client using the redirection URI provided earlier. The redirection URI includes the access token in the URI fragment.
4. The user agent follows the redirection instructions by making a request to the web server without the fragment. The user agent retains the fragment information locally.
5. The web server returns a web page, which is typically an HTML document with an embedded script. The web page accesses the full redirection URI including the fragment retained by the user agent. It can also extract the access token and other parameters contained in the fragment.
6. The user agent runs the script provided by the web server locally, which extracts the access token and passes it to the client.

Resource owner password credentials flow

The resource owner password credentials grant type is suitable in cases where the resource owner has a trust relationship with the client. For example, the resource owner can be a computer operating system of the OAuth client or a highly privileged application.

You can only use this grant type when the OAuth client has obtained the credentials of the resource owner. It is also used to migrate existing clients using direct authentication schemes by converting the stored credentials to an access token.



The resource owner password credentials workflow diagram involves the following steps:

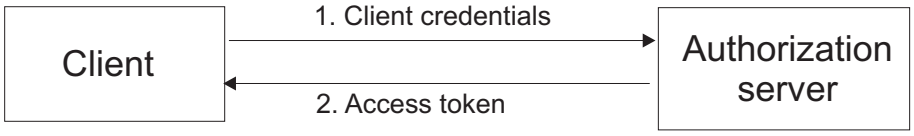
1. The resource owner provides the client with its user name and password.

2. The OAuth client requests an access token from the authorization server through the token endpoint. The OAuth client authenticates with its client credentials and includes the credentials received from the resource owner.
3. After the authorization server validates the resource owner credentials and the client credentials, it issues an access token and optionally a refresh token.

Client credentials flow

The client credentials flow is used when the OAuth client requests an access token using only its client credentials. This flow is applicable in one of the following situations:

- The OAuth client is requesting access to the protected resources under its control.
- The OAuth client is requesting access to a different protected resource, where authorization has been previously arranged with the authorization server.



The client credentials workflow diagram involves the following steps:

1. The OAuth client requests an access token from the token endpoint by authenticating with its client credentials.
2. After the authorization server validates the client credentials, it issues an access token.

Client authentication considerations at the OAuth 2.0 token endpoint

The OAuth 2.0 token endpoint is used for direct communications between an OAuth client and the authorization server.

The token endpoint is used to obtain an OAuth token. The client type, whether public or confidential, determines the authentication requirements of the OAuth 2.0 token endpoint.

OAuth 2.0 workflows for confidential clients that require client authentication at the token endpoint, can be configured in one of the following ways:

1. The Tivoli Federated Identity Manager point of contact requires authentication at the token endpoint:
   - The point of contact is responsible for authenticating the client.
   - The **Allow public clients to access the token endpoint** check box from the Federation properties panel is not relevant. A `client_secret` parameter must *not* be sent in the token endpoint request.
   - If a `client_id` parameter is sent in the request, it must match the identity of the client that is authenticated by the point of contact.
2. The Tivoli Federated Identity Manager point of contact permits unauthenticated access to the token endpoint:
   - The `client_id` parameter in the token endpoint request is used to identify the client.
   - The federation partner, also known as the client, must be enabled in order for it to be identified.

- The **Allow public clients to access the token endpoint** check box from the Federation properties panel determines whether a `client_secret` parameter is required in the token endpoint request. A client secret is required for confidential clients only.

Note: When enforcing client authentication at the token endpoint, the point of contact must contain the client ID and client secret within its user registry. The point of contact must be able to map the authenticated user credential to the `client_id` parameter sent in the OAuth 2.0 token endpoint request.

Based on this information, the following configurations are supported:

Table 113. Configurations supported

Client types	Configurations	WebSEAL point of contact token endpoint URI considerations	WebSphere Application Server point of contact token endpoint URI considerations	Check box setting for the "Allow public clients to access the token endpoint" parameter
Confidential clients	The point of contact performs client authentication.	• Authenticated ACL on token endpoint is required. • Token endpoint port must match WebSEAL port.	• Token endpoint port must match Tivoli Federated Identity Manager SOAP port. • OAuth Client must be in the **FIMSoapClient** role.	N/A
Confidential clients	The **client_id** and **client_secret** parameters in the token endpoint request are used to perform client authentication.	• Unauthenticated ACL on token endpoint is required. • Token endpoint port must match WebSEAL port.	Token endpoint must use the same point-of-contact host name and port as the authorize and clients manager endpoints.	Must be set to *false*.
Public clients	The **client_id** parameter is used to perform client validation.	• Unauthenticated ACL on token endpoint is required. • Token endpoint port must match the WebSEAL port.	Token endpoint must use the same point-of-contact host name and port as the authorize and clients manager endpoints.	Must be set to *true*.

Using WebSphere Application Server as the point of contact at the token endpoint

When enforcing authentication at the token endpoint for a WebSphere Application Server point of contact, the token endpoint URL must use the Tivoli Federated Identity Manager SOAP port. This condition ensures that authorization is enforced by the **FIMSoapClient** role. The Tivoli Federated Identity Manager SOAP endpoint can then be configured for the appropriate client authentication mechanisms, such as Basic Authentication or client certificate. See "Configuring the SOAP endpoint authentication settings" on page 379 for more details.

Note: You must manually set the **`TFIM.SOAP.Port`** and **`SOAP.AuthType`** runtime custom properties when using WebSphere Application Server in the following manner:

- As the point of contact server in a cluster
- To enforce authentication for the OAuth 2.0 token endpoint

The **Allow public clients to access the token endpoint** check box from the Federation properties panel has no influence on request processing when the point of contact is enforcing authentication.

The token endpoint URL must use the same point of contact host name and port as the authorize and clients manager endpoints when the following conditions apply:

- WebSphere Application Server is used as the point of contact.
- Unauthenticated access to the token endpoint is accepted.

In this case, the **FIMUnauthenticated** role is used. Additional authorization is based on whether the client is currently enabled. The **Allow public clients to access the token endpoint** check box from the Federation properties panel determines whether the `client_secret` parameter is required in the token endpoint request. A public client is not required to provide a `client_secret` parameter.

Using Tivoli Access Manager WebSEAL as the point of contact at the token endpoint

You can use the Tivoli Federated Identity Manager `tfimcfg` utility to configure WebSEAL as a point of contact for an OAuth 2.0 federation.

When enforcing authentication at WebSEAL for the token endpoint, use separate WebSEAL instances for the token and authorization endpoints. This condition makes it possible for clients to authenticate with authentication mechanisms, such as Basic Authentication and client certificates at the token endpoint. At the same time, users can still authenticate by using forms authentication at the authorize and clients manager endpoints. In this case, the token endpoint configuration within the OAuth 2.0 federation must match the host name and port of the appropriate token endpoint WebSEAL. For more details on how to use the `tfimcfg` utility to configure WebSEAL as a point of contact for an OAuth 2.0 federation, see "Configuring a WebSEAL point of contact server for the OAuth federation" on page 400.

Configuring the SOAP endpoint authentication settings

You can configure the Tivoli Federated Identity Manager SOAP endpoint to use Basic Authentication or client certificate as its client authentication mechanism.

About this task

The token endpoint URL uses the Tivoli Federated Identity Manager SOAP port when authentication for a WebSphere Application Server point of contact is enforced.

You can set how a client is authenticated by selecting a SOAP endpoint authentication type.

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. Select the SOAP endpoint authentication type from the following options:
   - Basic Authentication

     Authentication that requires your OAuth client to provide the client identifier and shared-secret.
   - Client Certificate Authentication

     Authentication that requires your OAuth client to present a certificate to establish a secure authenticated session.
6. Click **OK**.
7. Click **Load configuration changes to Tivoli Federated Identity Manager runtime**.

Client registration

A client is added to an OAuth federation as a partner. It neither is a *Service Provider* or an *Identity Provider*.

Creating a partner in an OAuth federation is the same as registering a client to an OAuth server or authorization server. An OAuth server or authorization server can have more than one client. Consequently, an OAuth federation can have more than one partner.

An OAuth federation can communicate with OAuth clients that are either managed in Tivoli Federated Identity Manager, or from an external client provider.

The OAuth federation generates a unique set of client credentials, during each partner creation. The client key and client secret are examples of the set of client credentials the OAuth federation generates. Clients use these credentials to identify themselves to an OAuth server or authorization server when requesting access to a protected resource.

State management

The `state_id` parameter in the STSUniversalUser module is used as a key to store or retrieve state information for each invocation of the trust chain of an OAuth flow.

Tivoli Federated Identity Manager provides sample mapping rules for the demonstration application that can place information into a WebSphere Application Server distributed map, IDMappingExtCache. These sample mapping rules use state management API and are applicable to both OAuth 1.0 and OAuth 2.0 protocols. The location of these sample mapping rules is:

`/opt/IBM/FIM/examples/demo/demo_rules/`

You can call the state management API from the XSLT or JavaScript mapping rule, or from a custom mapping module. This function is not available in a Tivoli Directory Integrator mapping rule. The Tivoli Directory Integrator server runs in a separate Java process, and therefore cannot use IDMappingExtUtil functions to access the WebSphere Application Server distributed map.

OAuth 1.0

OAuth 1.0 tokens have a `state_id` parameter that is used in Security Token Service mapping rules. The `state_id` parameter maintains state between associated Security Token Service calls in an OAuth 1.0 flow.

The `state_id` attribute is established when an OAuth client requests temporary credentials and remains the same in an entire OAuth 1.0 flow. It helps differentiate between two OAuth flows for the same OAuth client.

The sample mapping rule adds the storage time of the tokens to a distributed map and retrieves it during a request for a protected resource.

Figure 39 on page 382 shows sections of the sample OAuth 1.0 XSLT mapping rule that demonstrates the use of the state management API.

```
xmlns:cache-ext="com.tivoli.am.fim.trustserver.sts.utilities.IDMappingExtCache"
xmlns:mapping-ext="com.tivoli.am.fim.trustserver.sts.utilities.IDMappingExtUtils"
extension-element-prefixes="mapping-ext cache-ext" version="1.0">

...
<!-- The token_type parameter for this request. -->
    <xsl:variable name="token_type"
        select="//stsuuser:ContextAttributes/stsuuser:Attribute[@name='token_type']
               [@type='urn:ibm:names:ITFIM:oauth:request']/stsuuser:Value"/>
...
<!-- The state id handle for this OAuth token -->
  <xsl:variable name="state_id">
        <xsl:value-of select="//stsuuser:ContextAttributes/stsuuser:Attribute[@name='state_id']
               [@type='urn:ibm:names:ITFIM:oauth:state']/stsuuser:Value"/>
    </xsl:variable>
...
<!-- This template matches on a ContextAttribute for the parameter "state_id"
        with type "urn:ibm:names:ITFIM:oauth:state". If the current request is in request mode,
        we store the UTC time of the request into a cache with state_id as the key. If
        the current request is in validate mode and authorization succeeded, we retrieve the
        stored state value (i.e. the original UTC time of the request for a temporary token) and
        put it into an attribute named recovered_state. This is here just to demonstate use of the
        state management API.
    -->
    <xsl:template
        match="//stsuuser:ContextAttributes/stsuuser:Attribute[@name='state_id']
               [@type='urn:ibm:names:ITFIM:oauth:state']">

        <!-- First preserve this attribute in the output -->
        <stsuuser:Attribute>
            <xsl:attribute name="name">
                <xsl:value-of select="@name"/>
            </xsl:attribute>
            <xsl:attribute name="type">
                <xsl:value-of select="@type"/>
            </xsl:attribute>
            <xsl:for-each select="stsuuser:Value">
                <stsuuser:Value>
                    <xsl:value-of select="."/>
                </stsuuser:Value>
            </xsl:for-each>
        </stsuuser:Attribute>

        <!-- If the mode is "request", store something in the state cache -->
        <xsl:if test="$token_type = 'request'">
            <xsl:variable name="utc_time"
              select="concat('State storage time was: ', mapping-ext:getCurrentTimeStringUTC())"/>

            <!-- Get the cache -->
            <xsl:variable name="cache" select="mapping-ext:getIDMappingExtCache()"/>

             <!--
                  Store the UTC time to cache
                  Some containers ignore the variable declaration if the variable is not used.
                  That's why here we output a comment even though no output is needed to
                  call the method 'put' on the cache extension.
             -->
            <xsl:comment>
                <xsl:value-of
                   select="cache-ext:put($cache, $state_id, $utc_time, 1000)" />
            </xsl:comment>

        </xsl:if>

        <!-- If the mode is "validate" and authorization succeeded,
                                        get something from the state cache. -->
        <xsl:if test="$token_type = 'validate' and $authorizationResult = 'TRUE'">
            <!-- Get the cache and put it in recovered_state attribute -->
            <xsl:variable name="cache" select="mapping-ext:getIDMappingExtCache()"/>

            <stsuuser:Attribute name="recovered_state"
                     type="urn:ibm:names:ITFIM:oauth:response:attribute">
                <stsuuser:Value>
                    <xsl:value-of select="cache-ext:get($cache, $state_id)"/>
                </stsuuser:Value>
            </stsuuser:Attribute>
        </xsl:if>
    </xsl:template>
```

Figure 39. OAuth 1.0 XSL sample code with state management

OAuth 2.0

OAuth 2.0 tokens, such as grants, access tokens, and refresh tokens, have a `state_id` parameter that is used in Security Token Service mapping rules. The `state_id` parameter maintains state between associated Security Token Service calls in an OAuth 2.0 flow.

Similar to OAuth 1.0, the OAuth 2.0 mapping rule uses the `state_id` as the key to issue an authorization grant. The key is used to add the token storage time to a distributed map. The storage time is then retrieved from the cache during a request for a protected resource.

Figure 40 on page 384 shows sections of the sample OAuth 2.0 XSLT mapping rule that demonstrates the use of the state management API.

```
<!--
    The request_type parameter for this request. If none is supplied, assume "resource"
   -->
  <xsl:variable name="requestType">
    <xsl:choose>
      <xsl:when test="//stsuuser:ContextAttributes/stsuuser:Attribute[@name='request_type']
         [@type='urn:ibm:names:ITFIM:oauth:request']/stsuuser:Value">
        <xsl:value-of select="//stsuuser:ContextAttributes/stsuuser:Attribute[@name='request_type']
         [@type='urn:ibm:names:ITFIM:oauth:request']/stsuuser:Value"/>
      </xsl:when>
      <xsl:otherwise>resource</xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <!-- The state id handle for this OAuth token -->
  <xsl:variable name="stateId">
    <xsl:value-of select="//stsuuser:ContextAttributes/stsuuser:Attribute[@name='state_id']
         [@type='urn:ibm:names:ITFIM:oauth:state']/stsuuser:Value"/>
  </xsl:variable>
  <!-- The grant type for this OAuth request -->
  <xsl:variable name="grantType">
    <xsl:value-of select="//stsuuser:ContextAttributes/stsuuser:Attribute[@name='grant_type']
         [@type='urn:ibm:names:ITFIM:oauth:body:param']/stsuuser:Value"/>
  </xsl:variable>
...

<!-- This template matches on a ContextAttribute for the parameter "state_id"
        with type "urn:ibm:names:ITFIM:oauth:state". It does the following, if:

request_type = 'authorization' ==> Store the UTC time of the request into a cache
        with state_id as key [authorization_code, implicit]
request_type = 'access_token' && grant_type = 'client_credentials' ==> Store the UTC time
                of the request into a cache with state_id as key [client_credentials]
request_type = 'access_token' && grant_type = 'password' ==> Store the UTC time of the request
                into a cache with state_id as key [password]
request_type = 'resource' ==> Retrieve the stored time and
                put it into an attribute named recovered_state

This is here just to demonstrate use of the state management API.
    -->
  <xsl:template match="//stsuuser:ContextAttributes/stsuuser:Attribute[@name='state_id']
                  [@type='urn:ibm:names:ITFIM:oauth:state']">
    <!-- First preserve this attribute in the output -->
    <stsuuser:Attribute>
      <xsl:attribute name="name">
        <xsl:value-of select="@name"/>
      </xsl:attribute>
      <xsl:attribute name="type">
        <xsl:value-of select="@type"/>
      </xsl:attribute>
      <xsl:for-each select="stsuuser:Value">
        <stsuuser:Value>
          <xsl:value-of select="."/>
        </stsuuser:Value>
      </xsl:for-each>
    </stsuuser:Attribute>
    <xsl:choose>

      <!-- Store the UTC time as state when this is either:
       Authorization step, for authorization_code or implicit flows
       Token step, but only for client_credentials or resource owner credentials flows
      -->
      <xsl:when test="$requestType = 'authorization' or
        ($requestType = 'access_token' and
        ($grantType = 'client_credentials' or $grantType = 'password'))">

        <xsl:variable name="utc_time"
          select="concat('State storage time was: ', mapping-ext:getCurrentTimeStringUTC())"/>
        <!-- Get the cache -->
        <xsl:variable name="cache" select="mapping-ext:getIDMappingExtCache()"/>
        <!--
              Store the UTC time to cache.
              Some containers ignore the variable declaration if the variable is not used.
              That's why here we output a comment even though no output is needed to
              call the method 'put' on the cache extension.
           -->
        <xsl:comment>
          <xsl:value-of select="cache-ext:put($cache, $stateId, $utc_time, 1000)"/>
        </xsl:comment>
      </xsl:when>
      <xsl:when test="$requestType = 'resource'">
        <!-- Get the cache and put it in recovered_state attribute -->
        <xsl:variable name="cache" select="mapping-ext:getIDMappingExtCache()"/>
        <stsuuser:Attribute name="recovered_state"
                 type="urn:ibm:names:ITFIM:oauth:response:attribute">
          <stsuuser:Value>
            <xsl:value-of select="cache-ext:get($cache, $stateId)"/>
          </stsuuser:Value>
        </stsuuser:Attribute>
      </xsl:when>
    </xsl:choose>
  </xsl:template>
```

Figure 40. OAuth 2.0 XSL sample code with state management

Trusted clients management

Tivoli Federated Identity Manager stores trusted client information based on the decisions of a resource owner on which clients to trust.

The trusted clients manager support applies to both OAuth 1.0 and OAuth 2.0 federations.

In an OAuth flow, the resource owner is asked to provide consent on the scopes requested by an OAuth client to access the protected resource. The resource owner can either grant permission or deny the OAuth client from its access request.

The OAuth server or authorization server uses the trusted clients manager endpoint to manage information about trusted clients.

During the authorization step, the OAuth server or authorization server checks whether an OAuth client with a specific scope is stored in the trusted clients management endpoint. If the OAuth client has stored scope information, the OAuth server does not require the resource owner to grant authorization to access the protected resource.

Administrators can save information of a trusted client partner through the following options:

- Store trusted clients information in a browser cookie
- Do an auto-consent to all trust decisions in a closed authentication environment
- Store trusted clients information in memory (*to be used for test use only*)

By default, trusted clients information is stored on the browser of the resource owner as persistent cookies. Tivoli Federated Identity Manager provides a `TrustedClientsManager` extension point that administrators can use to write their own custom plug-in for storing and retrieving information.

OAuth EAS overview

The external authorization service (EAS) is a modular authorization service plug-in. System designers can use IBM Tivoli Access Manager authorization as an add-on to their own authorization models when they have the external authorization service (EAS).

The EAS plug-in uses the IBM Tivoli Federated Identity Manager OAuth capabilities.

Using the OAuth EAS, OAuth decisions can be made part of the standard authorization on WebSEAL requests. This means that WebSEAL can be used as the authorization enforcement point for access to OAuth-protected resources. The OAuth EAS plug-in works for both OAuth 1.0 and 2.0 protocols.

To learn more about the EAS plug-in, see the topic on "External authorization service plug-ins" in the IBM Tivoli Access Manager for e-business Information Center: http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.itame.doc/welcome.htm.

The OAuth EAS is responsible for the following actions in an authorization process:

- Receives the authorization request.

- Verifies that all required data is available in the request.
- Constructs a Request Security Token (RST) and sends it to the Policy Decision Point (PDP), which is Tivoli Federated Identity Manager.
- Grants or denies access to the protected resource based on the decision received from Tivoli Federated Identity Manager.

The OAuth EAS communicates with Tivoli Federated Identity Manager through the Security Token Service (STS) interface. See "OAuth STS Interface for Authorization Enforcement Points" on page 411 for more information.

OAuth data

Tivoli Federated Identity Manager requires specific information regarding the request to return an authorization decision.

Configuration data

The OAuth EAS accepts two optional configuration parameters as query string parameters in the request. These parameters affect the composition of the message the OAuth EAS sends to Tivoli Federated Identity Manager to obtain an authorization decision.

These parameters can also be configured statically in the OAuth EAS configuration stanza. See "[oauth-eas] stanza" on page 428 for more information.

- Mode (*Either OAuth10 or OAuth20Bearer*)
- Federation ID (*The provider ID of the Tivoli Federated Identity Manager OAuth federation associated with the OAuth Client*)

Authorization data

The following OAuth request data is obtained from either the authorization header, the post body, or the query string.

An OAuth 1.0 request consists of the following data:

- Realm (*Optional*)
- Consumer Key
- Token (*Optional*)
- Signature Method
- Time Stamp
- Nonce
- Signature
- Version (*Optional*)

An OAuth 2.0 request requires the Access Token. Tivoli Federated Identity Manager currently supports the bearer token specification only.

Resource information

This data is obtained from the HTTP request and is used by Tivoli Federated Identity Manager to validate the OAuth signature. This data is sent to Tivoli Federated Identity Manager regardless of the OAuth version.

The resource information includes the following data:

- Request Method (*For example, GET or POST*)
- Scheme (*For example, HTTP or HTTPS*)
- Host header from the request
- Request Path

- Request Query
- Request Body

  Only if the following conditions apply:
  - The entity-body is single-part.
  - The entity-body follows the encoding requirements of the `application/x-www-form-urlencoded` content-type as defined by the W3C HTML 4.0 specification. See http://www.w3.org/TR/1998/REC-html40-19980424/.
  - The HTTP request entity-header includes the **Content-Type** header field set to `application/x-www-form-urlencoded`.
- Port

  Only if the following conditions apply:
  - It is present in the request URL.
  - It is not the default port for the scheme. For example, if the scheme is HTTP and the port is not 80.

WebSEAL uses the EAS plug-in to provide the required data and use the OAuth feature in Tivoli Federated Identity Manager.

Error responses

An HTTP response indicates the type of error that has occurred when an action in an authorization process fails.

In some circumstances, the following HTTP error responses must be returned to the client:

- 400 Bad Request
- 401 Unauthorized
- 502 Bad Gateway

For the 401 response, an additional `WWW-Authenticate` header is added to the response in the following format:

`WWW-Authenticate: OAuth realm = <realm-name>`

The HTML component of the responses is preinstalled from files that have been specified in the EAS configuration.

See the WebSEAL Administration Guide for details on how to configure response template files for the OAuth EAS.

Federation and partner configuration information

Tivoli Federated Identity Manager provides wizards to create OAuth 1.0 and OAuth 2.0 federations and register a client as a partner. Complete the appropriate worksheets before running the Federation and Partner creation wizards.

Use the appropriate worksheet to gather information when preparing for the configuration process:

- "OAuth 1.0 service provider worksheet" on page 388
- "OAuth 1.0 service provider partner worksheet" on page 391
- "OAuth 2.0 service provider worksheet" on page 392
- "OAuth 2.0 service provider partner worksheet" on page 396

OAuth 1.0 service provider worksheet

Use this worksheet to plan your properties in creating an OAuth 1.0 federation, and refer to it when running the wizard.

The following table provides descriptions of the OAuth 1.0 federation properties and a space for you to write your values for each property.

Table 114. Worksheet for OAuth 1.0 federation configuration properties

Property	Description	Your value
Federation Name	Specifies the name of the federation. (**Required**) Use a name that describes the purpose of the federation. Enter an alphanumeric value. For example, a social networking site can be an OAuth server. A photo printing service that can print photos stored in the social networking site can be an OAuth client. The name of the federation might be: *MySocialNetwork*	
My Role	Specifies your role in the federation. **Default**: Service Provider A service provider provides a service to users. In most cases, the OAuth service provider protects the resources, and resource owners can authorize OAuth Clients to access these protected resources.	Service Provider
Company Name	Specifies the name of the company that is creating this federation. The value can be any string. You can also use the space character. (**Required**)	
Company URL	Specifies the URL of the company that is creating this federation. (**Optional**) For example: `http://www.example.com`	
First Name and Last Name	Specifies the name of the contact person of the company in this federation. (**Optional**) For example: `John Smith`	
Email Address	Specifies the email address of the contact person of the company in this federation. (**Optional**) For example: `johnsmith@example.com`	

Table 114. Worksheet for OAuth 1.0 federation configuration properties (continued)

Property	Description	Your value
Phone Number	Specifies the telephone number of the contact person of the company in this federation. (**Optional**) For example: `+1-555-555-5555`	
Contact Type	Specifies the type of contact. (**Optional**) The choices are: • Technical • Support • Administrative • Billing • Other	
Other Information	Specifies an optional text field for entering additional information about the federation contact. (**Optional**)	
Federation Protocol	Specifies the federation protocol. **Default**: OAuth 1.0	OAuth 1.0
Point of contact server	Specifies the URL address of the server that acts as initial point of contact for incoming requests. The address consists of a protocol specification, the server host name, and (optionally) a port number. When WebSEAL is the point of contact server, the WebSEAL junction is specified. (**Required**) Example value: `https://webseald.example.com/FIM`	
OAuth Client Provider	Specifies the client provider for your OAuth 1.0 federation. (**Required**) You can select from one of the options: • Clients managed by IBM Tivoli Federated Identity Manager • Clients managed by external client provider **Default**: Clients managed by IBM Tivoli Federated Identity Manager	

Table 114. Worksheet for OAuth 1.0 federation configuration properties (continued)

Property	Description	Your value
External Client Provider Implementation	Specifies how OAuth clients are managed externally. (**Required if the *Clients managed by external client provider* option is selected as the OAuth Client Provider**) You can write custom implementations for the OAuth 1.0 Client Provider Extension Point. Tivoli Federated Identity Manager reads client configuration data from your external configuration source. The external client provider plug-in supports GUIXML for parameter configuration.	
Maximum allowable clock skew between OAuth server and client (seconds)	Specifies the maximum time difference between the OAuth server and the OAuth client. This parameter accounts for clock skew between: • The clock of the OAuth server • The clock of the OAuth client Typically the skew time is a small number. (**Required**) **Default**: 300	
Temporary credentials and verification code lifetime (seconds)	Specifies the validity of the temporary credentials and verification code in seconds. (**Required**) **Default**: 300	
Maximum token credentials lifetime (seconds)	Specifies the OAuth access token lifetime lapse in seconds. When this lifetime expires, the client cannot access the protected resource. The resource owner must reauthorize the client to access the protected resource. (**Required**) **Default**: 604800	

Table 114. Worksheet for OAuth 1.0 federation configuration properties (continued)

Property	Description	Your value
Identity mapping options	Specifies how you want to do identity mapping for your OAuth federation. (**Required**) Select one of the following options: • **Use XSLT or Javascript mapping rules for identity mapping** Select this option when you create an XSLT or a Javascript mapping rule for identity mapping. • **Use Tivoli Directory Integrator for mapping** Select this option when you have a Tivoli Directory Integrator assembly line for the identity mapping. • **Use custom mapping module instance** Select this option when you have a custom trust service module for the identity mapping.	
Identity mapping rules file	Specifies the mapping rule file name if the XSLT or JavaScript for identity mapping is used.	
Custom mapping modules	Specifies the name of the module if the custom mapping module is used as the identity mapping rule.	*Make note of the module name:*

OAuth 1.0 service provider partner worksheet

Use this worksheet to plan your OAuth 1.0 partner properties, and refer to it when running the wizard.

Note: Partner creation and configuration are applicable only to federations that use Tivoli Federated Identity Manager as its client provider.

The following table provides descriptions of the OAuth 1.0 partner properties and a space for you to write your values for each property.

Table 115. Worksheet for OAuth 1.0 partner configuration properties

Property	Description	Your value
OAuth Client Company Name	Specifies the name of the company for this partner. It can be any character string. (**Required**)	
Company URL	Specifies the URL of the company for this partner. (**Optional**) For example: `http://www.example.com`	
First Name and Last Name	Specifies the name of the contact person for this partner. (**Optional**) For example: `John Smith`	

Table 115. Worksheet for OAuth 1.0 partner configuration properties (continued)

Property	Description	Your value
Email Address	Specifies the email address of the contact person for this partner. (**Optional**) For example: `johnsmith@example.com`	
Phone Number	Specifies the telephone number of the contact person for this partner. (**Optional**) For example: `+1-555-555-5555`	
Contact Type	Specifies the type of contact. (**Optional**) The choices are: • Technical • Support • Administrative • Billing • Other	
Other Information	Specifies an optional text field for entering additional information about the partner contact. Use any character string. (**Optional**)	
Client Identifier	Specifies a unique identifier that is supplied to the OAuth client to identify itself to the OAuth server. You cannot modify this value from this panel. (**Required**)	
Client Shared-Secret	Specifies a secret shared between the OAuth client and OAuth server that is used for signing requests. This field is automatically generated by the OAuth server, but you can overwrite it with a value of your choice. (**Required**)	
Client Callback URI	Specifies a callback URI to which the resource owner is redirected to when authorization is completed. An out-of-band configuration to receive callbacks can be supported by setting this value to `oob`. If you do not register a callback URI, Tivoli Federated Identity Manager treats it as an `oob`. (**Optional**)	
Override the registered client callback URI	Specifies whether to override the registered client callback URI. Select this check box to override the registered client callback URI with the callback URI parameter in the request for temporary credentials. (**Optional**)	

OAuth 2.0 service provider worksheet

Use this worksheet to plan your properties in creating an OAuth 2.0 federation, and refer to it when running the wizard.

The following table provides descriptions of the OAuth 2.0 federation properties and a space for you to write your values for each property.

Table 116. Worksheet for OAuth 2.0 federation configuration properties

Property	Description	Your value
Federation Name	Specifies the name of the federation. (**Required**) Use a name that describes the purpose of the federation. Enter an alphanumeric value. For example, a social networking site can be an Authorization server. A photo printing service that can print photos stored in the social networking site can be an OAuth client. The name of the federation might be *MySocialNetwork*.	
My Role	Specifies your role in the federation. (**Required**) **Default**: Service Provider A service provider provides a service to users. In most cases, the OAuth service provider protects the resources, and resource owners can authorize OAuth Clients to access these protected resources.	Service Provider
Company Name	Specifies the name of the company that is creating this federation. The value can be any string. You can also use the space character. (**Required**)	
Company URL	Specifies the URL of the company that is creating this federation. (**Optional**) For example: `http://www.example.com`	
First Name and Last Name	Specifies the name of the contact person of the company in this federation. (**Optional**) For example: `John Smith`	
Email Address	Specifies the email address of the contact person of the company in this federation. (**Optional**) For example: `johnsmith@example.com`	
Phone Number	Specifies the telephone number of the contact person of the company in this federation. (**Optional**) For example: `+1-555-555-5555`	
Contact Type	Specifies the type of contact. (**Optional**) The choices are: • Technical • Support • Administrative • Billing • Other	

Table 116. Worksheet for OAuth 2.0 federation configuration properties (continued)

Property	Description	Your value
Other Information	Specifies an optional text field for entering additional information about the federation contact. (**Optional**)	
Federation Protocol	Specifies the federation protocol. (**Required**) **Default**: OAuth 2.0	OAuth 2.0
Point of contact server	Specifies the URL address of the server that acts as initial point of contact for incoming requests. (**Required**) The address consists of a protocol specification, the server host name, and (optionally) a port number. When WebSEAL is the point of contact server, the WebSEAL junction is specified. Example value: `https://webseald.example.com/FIM`	
OAuth Client Provider	Specifies the client provider for your OAuth 2.0 federation. (**Required**) You can select from one of the options: • Clients managed by IBM Tivoli Federated Identity Manager • Clients managed by external client provider **Default**: Clients managed by IBM Tivoli Federated Identity Manager	
External Client Provider Implementation	Specifies how OAuth clients are managed externally. (**Required if the *Clients managed by external client provider* is selected as the OAuth Client Provider**) You can write custom implementations for the OAuth 2.0 Client Provider Extension Point. Tivoli Federated Identity Manager reads client configuration data from your external configuration source. The external client provider plug-in supports GUIXML for parameter configuration.	
Authorization Grant Types	Specifies the list of supported grant types for an OAuth 2.0 federation. (**Required**) You must select at least one grant type: • Authorization code • Implicit grant • Client credentials • Resource owner password credentials **Default**: Authorization code and implicit grant	

Table 116. Worksheet for OAuth 2.0 federation configuration properties (continued)

Property	Description	Your value
Maximum authorization grant lifetime (seconds)	Specifies the maximum duration of a grant where the resource owner authorized the OAuth client to access the protected resource. (**Required**) This field applies only to the authorization code and resource owner password credentials grant types. This lifetime affects the validity of an authorization code, an access token, and a refresh token. The value for this lifetime must be greater than the values specified for the authorization code and access token lifetimes. When this lifetime expires, the resource owner must reauthorize the OAuth client to obtain an authorization grant to access the protected resource. **Default**: 604800	
Authorization code lifetime (seconds)	Specifies the validity of the authorization code in seconds. (**Required**) This option applies only to an authorization code grant type. The authorization server generates an authorization code and issues it to the OAuth client. The OAuth client uses the authorization code in exchange for an access token. **Default**: 300	
Issue refresh token	Specifies whether a refresh token is issued to the OAuth client. (**Optional**) A refresh token obtains a new set of an access token and a refresh token. This option applies only to authorization code and resource owner password credentials grant types. **Default**: No (The check box is not selected)	
OAuth Access Token Type	Specifies the type of OAuth access token used by an OAuth client to make protected resource requests. (**Required**) The default value is **OAuth Bearer Token Type**. A bearer token is a security token that gives ownership to whoever holds the token. **Default**: OAuth Bearer Token Type	
Access token lifetime (seconds)	Specifies the validity of the access token in seconds. (**Required**) When this lifetime expires, the OAuth client cannot use the current access token to access the protected resource. **Default**: 3600	

Table 116. Worksheet for OAuth 2.0 federation configuration properties (continued)

Property	Description	Your value
Identity mapping options	Specifies how you want to do identity mapping for your OAuth federation. (**Required**) Select one of the following options: • **Use XSLT or Javascript mapping rules for identity mapping** Select this option when you create an XSLT or a Javascript mapping rule for identity mapping. • **Use Tivoli Directory Integrator for mapping** Select this option when you have a Tivoli Directory Integrator assembly line for the identity mapping. • **Use custom mapping module instance** Select this option when you have a custom trust service module for the identity mapping.	
Identity mapping rules file	Specifies the mapping rule file name if the XSLT or JavaScript for identity mapping is used.	
Custom mapping modules	Specifies the name of the module if the custom mapping module is used as the identity mapping rule.	*Make note of the module name:*

OAuth 2.0 service provider partner worksheet

Use this worksheet to plan your properties for your OAuth 2.0 partner, and refer to it when running the wizard.

Note: Partner creation and configuration are applicable only to federations that use Tivoli Federated Identity Manager as its client provider.

The following table provides descriptions of the OAuth 2.0 partner properties and a space for you to write your values for each property.

Table 117. Worksheet for OAuth 2.0 partner configuration properties

Property	Description	Your value
OAuth Client Company Name	Specifies the name of the company for this partner. It can be any character string. (**Required**)	
Company URL	Specifies the URL of the company for this partner. (**Optional**) For example: `http://www.example.com`	
First Name and Last Name	Specifies the name of the contact person for this partner. (**Optional**) For example: `John Smith`	
Email Address	Specifies the email address of the contact person for this partner. (**Optional**) For example: `johnsmith@example.com`	

Table 117. Worksheet for OAuth 2.0 partner configuration properties (continued)

Property	Description	Your value
Phone Number	Specifies the telephone number of the contact person for this partner. (**Optional**) For example: `+1-555-555-5555`	
Contact Type	Specifies the type of contact. (**Optional**) The choices are: • Technical • Support • Administrative • Billing • Other	
Other Information	Specifies an optional text field for entering additional information about the partner contact. Use any character string. (**Optional**)	
Client Identifier	Specifies a unique identifier supplied to the client to identify itself to the authorization server. This field is automatically generated by the authorization server, but you can overwrite it with an alphanumeric value of your choice. (**Required**)	
Client Shared-Secret	Specifies a secret shared between the OAuth client and the authorization server. This field is automatically generated by the OAuth server, but you can overwrite it with a value of your choice. If you do not register the OAuth client a secret, it becomes a public client. If you do register a secret, the OAuth client becomes a confidential client. (**Optional**)	
Client Redirection URI	Specifies a URI to which the resource owner is redirected to when authorization is completed. The authorization server returns grants or tokens only to this registered redirection URI or its child. (**Optional**)	

Chapter 28. Configuring an OAuth federation

To configure an OAuth federation, you must create the federation, add your partner to your federation, and configure the enforcement point for the protected resource.

Configuring an OAuth service provider federation

Use the federation wizard to create and configure a service provider federation.

Before you begin

Before beginning this procedure, complete the worksheet that is appropriate for the OAuth protocol:

- "OAuth 1.0 service provider worksheet" on page 388
- "OAuth 2.0 service provider worksheet" on page 392

About this task

For detailed descriptions of the fields in the Federation wizard, see the online help.

Procedure

1. Log on to the Integrated Solutions Console.
2. Select **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations** The Current Domain and Federations portlets open.
3. Click **Create** to start the Federation wizard.
4. Use the completed worksheet as a guide for completing the fields.
5. To proceed to the next panel, click **Next**.
   a. (Optional) If you must go back to adjust a configuration setting, click **Back**.
   b. (Optional) If you want to end the configuration, click **Cancel**.

   When you have completed all configuration panels, the Summary panel opens.
6. Verify that the configuration settings are correct.
7. Click **Finish**. The Create Federation Complete portlet opens.
8. (Optional) If you are using the internal client provider, you can add your partner now or later.
   - Click **Add partner** to start the Partner wizard and register an OAuth client to your OAuth service provider federation. See the steps described in "Adding a partner to an OAuth federation" on page 402.
   - Click **Done** to add your partner at a later time.
9. (Optional) If you are using an external client provider, click **Done** to return the Federation portlet.
10. Click **Load configuration changes to Tivoli Federated Identity Manager runtime** to deploy the changes.

Enabling two-legged OAuth validation

Configure the properties of an existing OAuth 1.0 federation when you want to enable two-legged OAuth validation in your federation.

Before you begin

You must have an existing OAuth 1.0 federation before doing this task. To configure an OAuth 1.0 federation, see "Configuring an OAuth service provider federation" on page 399.

If you use the WebSphere Trust Association Interceptor as an enforcement point, you must create a user in the WebSphere registry that matches the user name returned by your mapping rule. See the WebSphere Application Server Version 6.0 Information Center for information about user creation. To understand how the Security Token Service and enforcement point behave in a two-legged OAuth, see "Security Token Service interface for two-legged OAuth flow" on page 372.

Procedure

1. Log on to the Integrated Solutions Console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations**. The Federation panel shows a list of configured federations.
3. Select an OAuth federation from the table and click **Properties**. The **Federation Properties** panel opens.
4. Select the **Enable two-legged OAuth validation** check box.
5. Cick **OK** to exit the panel.
6. Click **Load configuration changes to Tivoli Federated Identity Manager runtime** to reload your changes.

Results

Two-legged OAuth validation has been enabled in the federation.

Configuring a WebSEAL point of contact server for the OAuth federation

If you use WebSEAL as the point of contact server for your OAuth federation, you must configure it using the configuration utility tool.

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.

Before starting this procedure:

- The WebSEAL point of contact profile must be activated.
- You must know the Tivoli Access Manager administration user (default: sec_master) and administration user password.

About this task

The Federation wizard provides a button that you can use to obtain the configuration utility tool. The procedure includes information on how to obtain and run the utility. The utility configures endpoints on the WebSEAL server, creates a WebSEAL junction, attaches the appropriate ACLs, and enables the necessary authentication methods.

The steps are applicable for OAuth 1.0 and 2.0 federations.

To configure WebSEAL as the point of contact server, complete the steps in this procedure:

Procedure

1. After creating the federation, click **Load configuration changes to Tivoli Federated Identity Manager runtime** to reload your changes.
2. Click **Done** to return to the Federations panel.
3. Click **Download Tivoli Access Manager Configuration Tool**.
4. Save the configuration tool to the file system on the computer that hosts the WebSEAL server.
5. Run the configuration tool from a command line. The syntax is:

   ```
   java -jar /download_dir/tfimcfg.jar -action tamconfig
   -cfgfile webseald-instance_name.conf
   ```

   Notes:

   - If Federal Information Processing Standards (FIPS) is enabled, you must specify the secure socket connection factory. For example:

     ```
     java -jar /download_dir/tfimcfg.jar -action tamconfig
     -cfgfile webseald-instance_name.conf -sslfactory TLS
     ```

   - **For OAuth 1.0 federations:** If an OAuth client sends OAuth protocol parameters through the HTTP Authorization header, the OAuth server must be able to accept the HTTP Authorization header. Use the `-b ignore` option on the junction between WebSEAL and Tivoli Federated Identity Manager to forward the HTTP Authorization header to the backend server. This option is not required on the junction if the OAuth client uses either the query string or POST body method.
   - **For OAuth 2.0 federations:** If an OAuth client accesses a Policy Enforcement Point that expects an HTTP Authorization header, the OAuth server must be able to accept the HTTP Authorization header. Use the `-b ignore` option on the junction between WebSEAL and Policy Enforcement Point to forward the HTTP Authorization header to the backend server. This option is only necessary if the Policy Enforcement Point reading the OAuth Authorization header is on a server behind WebSEAL. It is not necessary to run the `-b ignore` option when using the WebSEAL EAS enforcement point for OAuth 2.0.

Example

For example, when you have placed `tfimcfg.jar` file in `/tmp`, and the WebSEAL instance name is `default`, the command is:

```
java -jar /tmp/tfimcfg.jar -action tamconfig -cfgfile webseald-default
```

For more information, see Appendix A, “tfimcfg reference,” on page 753.

Configuring WebSphere as a point of contact server

Tivoli Federated Identity Manager is configured by default to use Tivoli Access Manager WebSEAL component as the point of contact server. To configure WebSphere as your point of contact server, you must make a configuration change.

Procedure

1. Log on to the administration console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Point of Contact**.
3. Select **WebSphere**.
4. Click **Make Active**.

Results

The WebSphere server is now configured to be the point of contact server.

Adding a partner to an OAuth federation

You can add a partner to your OAuth federation through the administration console.

Before you begin

Note: You can add partners to a federation only if the federation is configured to use the Tivoli Federated Identity Manager as the client provider.

Before beginning this procedure, complete the worksheet that is appropriate for the OAuth protocol:

- "OAuth 1.0 service provider partner worksheet" on page 391
- "OAuth 2.0 service provider partner worksheet" on page 396

About this task

These steps apply to OAuth 1.0 and OAuth 20 federations with internal client providers.

After completing the appropriate partner worksheet, use the Partner wizard in the console to add the partner.

For detailed descriptions of the fields in the federation wizard, see the online help.

Procedure

1. Log on to the Integrated Solutions Console.
2. Select **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Partners**. The Current Domain and Federation Partners portlet opens.
3. Click **Create** to start the Federation Partner wizard.
4. Select the OAuth federation to which you want to add a partner.
5. Click **Next**.
6. Enter the contact properties.
7. Click **Next**.

8. Configure the OAuth client registration.
9. Click **Next**.
10. Click **Next** to see a summary of all the information you entered.
11. Verify that the settings are correct.
12. Click **Finish**. The Add Partner Complete portlet opens.

    Note: The partner has been added to the federation, but it is disabled as a security precaution.
13. Click **Enable Partner** to activate this partner.
14. Click **Load configuration changes to Tivoli Federated Identity Manager runtime** to deploy the changes.

Configuring the WebSphere OAuth Trust Association Interceptor

You can configure the WebSphere Trust Association Interceptor (TAI) component to act as an OAuth authorization enforcement point.

Before you begin

Review the properties that you must configure your enforcement point for either an OAuth 1.0 or an OAuth 2.0 federation. See "OAuth Trust Association Interceptor and Servlet Filter custom properties" on page 422 for more information.

About this task

You can use the WebSphere OAuth Trust Association Interceptor only to protect access to WebSphere resources that require authentication for access. Trust Association Interceptors are not started on every web request, but only when authentication is required.

The steps are applicable for OAuth 1.0 and OAuth 2.0 federations.

Procedure

1. Copy the `com.tivoli.am.fim.ws.oauth.jar` file from the following locations:
   - AIX, Linux, or Solaris

     `/opt/IBM/FIM/tools/oauth/com.tivoli.am.fim.ws.oauth.jar`
   - Windows

     `C:\Program Files\IBM\FIM\tools\oauth\com.tivoli.am.fim.ws.oauth.jar`

   to the following directories:
   - AIX, Linux, or Solaris

     `/opt/IBM/WebSphere/AppServer/lib/ext`
   - Windows

     `C:\Program Files\IBM\WebSphere\AppServer\lib\ext`
2. Log on to the WebSphere administration console.

   For WebSphere 6.1:
   select **Security** > **Secure administration, applications and infrastructure** > **Web security** > **Trust association**.

   For WebSphere 7 and WebSphere 8:
   Select **Security** > **Global security** > **Web and SIP security** > **Trust association**.
3. Select **Enable trust association**.

4. Click **Apply**.
5. Click **Interceptors**.
6. Click **New** to add a new interceptor.
7. Enter the interceptor class name:
   `com.tivoli.am.fim.ws.oauth.tai.OAuthTAI`
8. Click **Apply**.
9. Click **Custom properties**.
10. Add custom properties for your environment. See "OAuth Trust Association Interceptor and Servlet Filter custom properties" on page 422 for a list of the properties.
11. Save the configuration updates.
12. Restart WebSphere.

Configuring the WebSphere OAuth Servlet Filter

You can configure the WebSphere Servlet Filter component as an OAuth authorization enforcement point.

About this task

This topic applies to OAuth 1.0 and OAuth 2.0 federations.

The OAuth Servlet Filter handles authentication for protected resources hosted in WebSphere, the same way that the trust association interceptor does.

You can use the `com.tivoli.am.fim.ws.oauth.jar` file as your Servlet Filter application library.

Procedure

1. Include the `com.tivoli.am.fim.ws.oauth.jar` file in your custom EAR application.
2. Open the `MANIFEST.MF` file of the `WAR` application.
3. Specify `com.tivoli.am.fim.ws.oauth.jar` in the **class path header** field.
4. Save the manifest file.
5. Open the web deployment descriptor of the protected resource, which is the `web.xml` file.
6. Add the filter with initialization parameters to match your environment. See the topic on "OAuth Trust Association Interceptor and Servlet Filter custom properties" on page 422 for information about the initialization parameters. The Servlet Filter has the same parameters as the configuration properties for the trust association interceptor.

Example

Figure 41 on page 405 shows a sample code (`web.xml`) for OAuth 1.0 that contains a JSP that is protected by the Servlet Filter

```
  <display-name>com.tivoli.am.fim.war.fimivt</display-name>
  <filter>
    <description>Performs OAuth authorization</description>
    <display-name>OAuth servlet filter</display-name>
    <filter-name>OAuth servlet filter</filter-name>
    <filter-class>com.tivoli.am.fim.ws.oauth.sf.OAuthServletFilter</filter-class>
       <init-param>
      <description/>
      <param-name>DefaultMode</param-name>
      <param-value>OAuth10</param-value>
    </init-param>
    <init-param>
      <description/>
      <param-name>ModeParameterName</param-name>
      <param-value>mode</param-value>
    </init-param>
    <init-param>
      <description/>
      <param-name>URIPrefix</param-name>
      <param-value>/fimivt/oauth/sfprotected.jsp</param-value>
    </init-param>
    <init-param>
      <description/>
      <param-name>STSEndpoint</param-name>
      <param-value>http://server.oauth.com/TrustServer/SecurityTokenService</param-value>
    </init-param>
    <init-param>
      <description/>
      <param-name>OAuthRealm</param-name>
      <param-value>https://server.oauth.com//</param-value>
    </init-param>
    <init-param>
      <description/>
      <param-name>PointOfContact</param-name>
      <param-value>https://server.oauth.com/</param-value>
    </init-param>
     <init-param>
      <description/>
      <param-name>DefaultFederationId</param-name>
      <param-value>https://server.oauth.com/FIM/MySocialNetwork/oauth10</param-value>
    </init-param>
  <init-param>
      <description/>
      <param-name>OAuthTokenCacheSize</param-name>
      <param-value>2</param-value>
    </init-param></filter>
  <init-param>
      <description/>
      <param-name>FederationIdRequestParameterName</param-name>
      <param-value>FederationId</param-value>
    </init-param>
</filter>
  <filter-mapping>
    <filter-name>OAuth servlet filter</filter-name>
    <url-pattern>/oauth/sfprotected.jsp</url-pattern>
  </filter-mapping>
```

Figure 41. Sample JavaScript code for OAuth 1.0

Figure 42 on page 406 shows a sample code (`web.xml`) for OAuth 2.0 that contains a JSP that is protected by the Servlet Filter

```
<display-name>com.tivoli.am.fim.war.fimivt</display-name>
<filter>
  <description>Performs OAuth authorization</description>
  <display-name>OAuth servlet filter</display-name>
  <filter-name>OAuth servlet filter</filter-name>
  <filter-class>com.tivoli.am.fim.ws.oauth.sf.OAuthServletFilter</filter-class>
 <init-param>
    <description/>
    <param-name>DefaultMode</param-name>
    <param-value>OAuth20Bearer</param-value>
  </init-param>
  <init-param>
    <description/>
    <param-name>ModeParameterName</param-name>
    <param-value>mode</param-value>
  </init-param>
  <init-param>
    <description/>
    <param-name>URIPrefix</param-name>
    <param-value>/fimivt/oauth/sfprotected.jsp</param-value>
  </init-param>
  <init-param>
    <description/>
    <param-name>STSEndpoint</param-name>
    <param-value>http://server.oauth.com/TrustServer/SecurityTokenService</param-value>
  </init-param>
  <init-param>
    <description/>
    <param-name>OAuthRealm</param-name>
    <param-value>https://server.oauth.com//</param-value>
  </init-param>
  <init-param>
    <description/>
    <param-name>PointOfContact</param-name>
    <param-value>https://server.oauth.com/</param-value>
  </init-param>
  <init-param>
    <description/>
    <param-name>DefaultFederationId</param-name>
    <param-value>https://server.oauth.com/FIM/MySocialNetwork/oauth20</param-value>
  </init-param>
<init-param>
    <description/>
    <param-name>FederationIdRequestParameterName</param-name>
    <param-value>FederationId</param-value>
  </init-param>
<init-param>
    <description/>
    <param-name>OAuthTokenCacheSize</param-name>
    <param-value>2</param-value>
  </init-param></filter>
<filter-mapping>
  <filter-name>OAuth servlet filter</filter-name>
  <url-pattern>/oauth/sfprotected.jsp</url-pattern>
</filter-mapping>
```

Figure 42. Sample JavaScript code for OAuth 2.0

WebSEAL OAuth EAS configuration

Configure the OAuth external authorization service (EAS), which is a WebSEAL policy enforcement point (PEP), to support both OAuth 1.0 and OAuth 2.0.

Configure the WebSEAL OAuth external authorization service with one of the following methods:

- Configuring the WebSEAL OAuth EAS manually
- "Configuring the WebSEAL OAuth EAS with the tfimcfg tool" on page 409

OAuth decisions are included as part of the standard authorization on WebSEAL requests. Both configuration methods ensure that the correct data is passed to the OAuth EAS for each request.

Related concepts:

"OAuth EAS overview" on page 385
The external authorization service (EAS) is a modular authorization service plug-in. System designers can use IBM Tivoli Access Manager authorization as an add-on to their own authorization models when they have the external authorization service (EAS).

Configuring the WebSEAL OAuth EAS manually

Manually configure the OAuth external authorization service (EAS) for circumstances when you do not want to use default values.

Before you begin

IBM® Tivoli® Access Manager for e-business version 6.1.1 or later must be installed. If version 6.1.1 is installed, fix pack 5 or later must be applied.

Procedure

1. Enable OAuth EAS.
   a. Open the default WebSEAL configuration file with any file editor.

      UNIX or Linux
      ```
      /opt/pdweb/etc/webseald-default.conf
      ```

      Windows
      ```
      C:\Program Files\Tivoli\PDWeb\etc\webseald-default.conf
      ```

   b. Specify the <*policy-trigger*> entry in the `[aznapi-external-authzn-services]` stanza. The OAuth EAS requires a single parameter that corresponds to the configuration file that contains the OAuth EAS configuration data. The plug-in name for the OAuth EAS is `amwoautheas`. Its library is contained in the `pdwebrte/lib` directory.

   For example:

   UNIX or Linux
   ```
   oauth_pop_trigger = /opt/pdwebrte/lib/libamwoautheas.so &
   /opt/pdweb/etc/oauth_eas.conf
   ```

   Windows
   ```
   oauth_pop_trigger = C:\Program Files\Tivoli\PDWebRTE\bin\
   libamwoautheas.dll & C:\Program Files\Tivoli\PDWeb\etc\
   oauth_eas.conf
   ```

   For more information, see "[aznapi-external-authzn-services] stanza" on page 425.
2. Configure the required authorization decision data.

The OAuth EAS requires various data from the request. You can specify the request as HTTP request elements in the [azn-decision-info] stanza. The following configuration entries are required for the OAuth EAS to function correctly:

```
[azn-decision-info]
##
The following information will be provided to the authorization
# framework for every authorization request. This information
# is required by the OAuth EAS when validating an OAuth token.
#
HTTP_REQUEST_METHOD = method
HTTP_REQUEST_SCHEME = scheme
HTTP_REQUEST_URI = uri
HTTP_HOST_HDR = header:host
HTTP_CONTENT_TYPE_HDR = header:content-type
HTTP_TRANSFER_ENCODING_HDR = header:transfer-encoding
HTTP_AZN_HDR = header:authorization

[aznapi-configuration]

resource-manager-provided-adi = AMWS_pb_
```

Note: The required request data is the same for all environments.

3. Create the required HTML response files.

   For more information about the response file configuration parameters, see "[oauth-eas] stanza" on page 428.

4. Configure the extra EAS-specific data.

   The OAuth EAS requires specific configuration data to function correctly. This data is contained in the [oauth-eas] stanza of the configuration file. The configuration file name is provided as an argument to the [aznapi-external-authzn-services] stanza for the amwoautheas configuration entry of the WebSEAL configuration file.

   a. Open the OAuth EAS configuration file that you specified in Step 1b with any file editor.
   b. Specify the *default-fed-id*, *default-mode*, *realm-name*, *custom response files*, and *server url*. See "Sample EAS configuration data" on page 428 for an example of the [oauth-eas] stanza with the configuration details.

   Note: One of the required configuration entries in the [oauth-eas] stanza is cluster-name. It specifies the name of the Tivoli Federated Identity Manager cluster that hosts the OAuth service. You must configure a corresponding [tfim-cluster:<cluster>] stanza to define the specified cluster.

   See "[oauth-eas] stanza" on page 428 for more details on the required configuration entries.

5. Define the policy to start the OAuth EAS. For example:

```
#pdadmin -a sec_master
Enter password: passw0rd
pop create test-pop
pop modify test-pop set attribute eas-trigger oauth_pop_trigger
pop attach /WebSEAL/webseal.example.com-default/oauth test-pop
server replicate
quit
```

   Note: If OAuth Clients are send OAuth parameter data in the Authorization header mechanism, set -b ignore flag for the junction which you attach to the OAuth POP. For more information, see IBM WebSEAL Administration Guide.

6. Restart WebSEAL to apply the configuration changes.

UNIX or Linux

```
/opt/pdweb/bin/pdweb_start restart
```

Windows

Use the Services Control Panel:

Start > **Settings** > **Control Panel** > **Administrative Tools** > **Services**

Configuring the WebSEAL OAuth EAS with the tfimcfg tool

Use the tfimcfg tool to easily configure WebSEAL OAuth EAS.

Before you begin

IBM® Tivoli® Access Manager for e-business version 6.1.1 or later must be installed. If version 6.1.1 is installed, fix pack 5 or later must be applied.

Procedure

1. Run the `tfimcfg` tool.

   Use the following `tfimcfg` attributes:

   - For a WebSEAL server:

     ```
     java -jar tfimcfg.jar -action tamconfig -cfgfile
     WebSEAL_filename
     ```

   - For a Web Gateway Appliance server:

     ```
     java -jar tfimcfg.jar -action wgaconfig -cfgurl
     Web_Gateway_Appliance_URL
     ```

   See the `tfimcfg` reference for information about the tool parameters in the Federated Identity Manager Information Center.

   If you use the JRE provided with WebSphere Application Server, version 8.0 or later, you might encounter an error when you use `-action tamconfig`. See the "Known problems and solutions" topic in the Federated Identity Manager Information Center for the required `tfimcfg` parameters.

2. Provide the following OAuth PEP specific information:

   a. For the **Federation to configure** prompt, select **OAuth policy enforcement point**.

   b. Provide the following OAuth PEP parameters:

      - OAuth external authorization service library
        - Linux:

          ```
          /opt/pdwebrte/lib/libamwoautheas.so
          ```

        - AIX:

          ```
          /opt/pdwebrte/lib/libamwoautheas.a
          ```

        - Windows:

          ```
          C:\Program Files\Tivoli\PDWebRTE\bin\amwoautheas.dll
          ```

      - The `'400 Bad Request'` response page
      - The `'401 Unauthorized'` response page
      - The`'502 Bad Gateway'` response page
      - The default OAuth federation
      - The default OAuth mode

   c. Provide the following service parameters:

      Provide the following service parameters:

      - `Optional ITFIM Security Token Service client user ID`

- Optional ITFIM Security Token Service client password

d. The tfimcfg tool attempts to connect to the server that hosts OAuth service. If the connection is invalid, you are prompted to enter the parameters again.

3. Attach the oauth-pop to the resource that OAuth PEP must protect. For example, if MyJct is the resource to protect, run the following command:

```
#pdadmin -a sec_master
Enter password: passw0rd
pop attach /WebSEAL/localhost-default/MyJct oauth-pop
server replicate
quit
```

Chapter 29. OAuth reference

This topic contains references about the enforcement points and their custom properties, external authorization service (EAS) stanzas, and HTML template pages for both. This topic applies to both OAuth 1.0 and OAuth 2.0.

OAuth STS Interface for Authorization Enforcement Points

Use the WS-Trust interface to directly contact an OAuth Security Token Service (STS) trust chain in Tivoli Federated Identity Manager to validate a request for an OAuth protected resource. An OAuth enforcement point intercepts requests for OAuth protected resources. The OAuth enforcement point also validates the request with Tivoli Federated Identity Manager, and passes the request through, if it is valid. If the request is not valid, the enforcement point denies access to the protected resource.

OAuth STS overview

You can develop your own customized policy enforcement point to work with the Security Token Service (STS) trust chain through the STS interface. Some examples of existing customized policy enforcement points are WebSphere Servlet Filter, Trust Association Interceptor (TAI), and a reverse proxy such as WebSEAL. As Tivoli Federated Identity Manager supports both OAuth 1.0 and OAuth 2.0 federations, you can develop customized policy enforcement points to work with either type of the OAuth federations. The following diagram illustrates the relationship between the OAuth STS trust chain and other OAuth components.

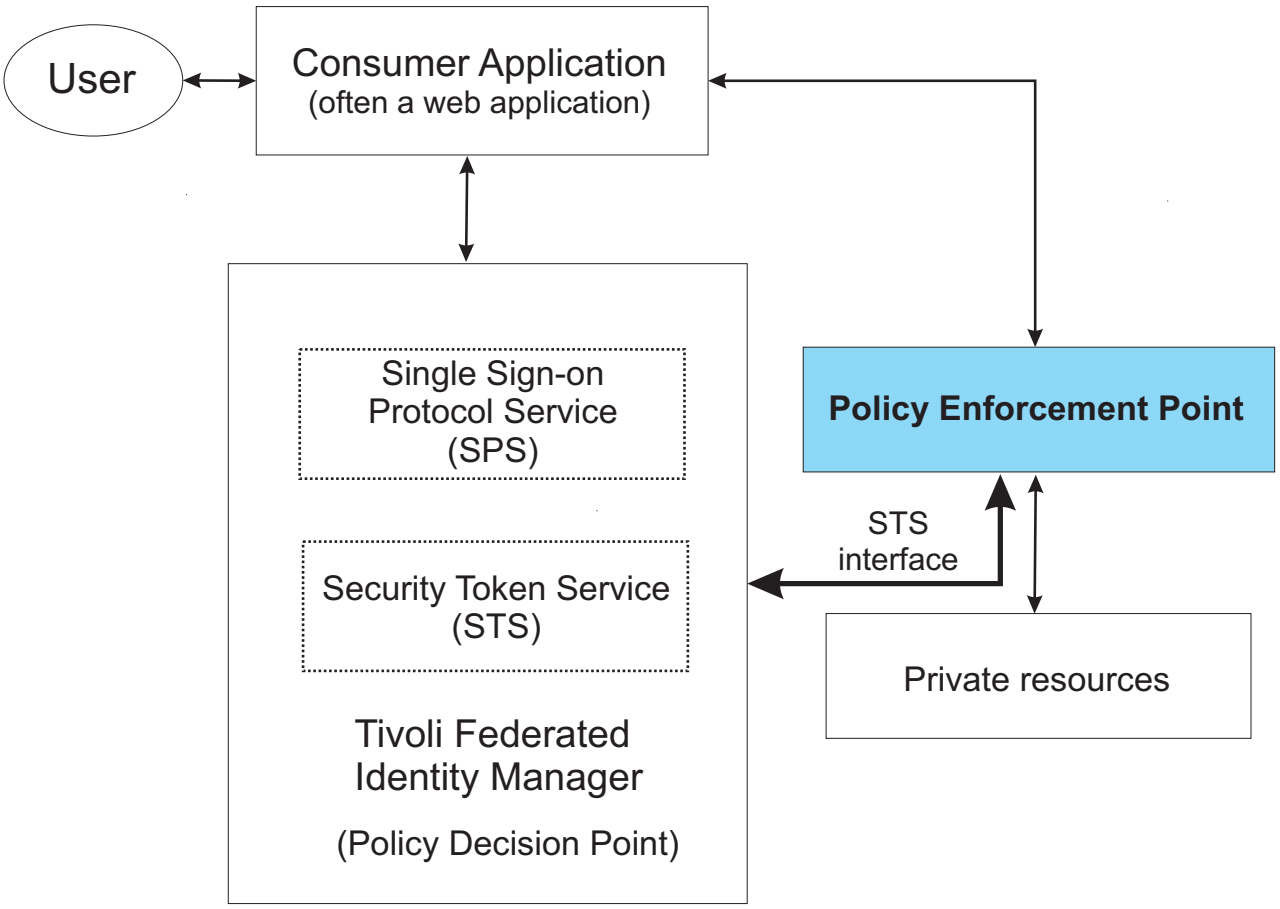


Figure 43. OAuth STS trust chain workflow

This section describes the process an OAuth enforcement point undertakes to transform an HTTP request for an OAuth protected resource into a WS-Trust message.

The transformation makes it possible for the Tivoli Federated Identity Manager STS to validate the request. It also describes the possible responses an enforcement point can receive from the STS and how to deal with them.

The interface and message structure is the same for both OAuth 1.0 and OAuth 2.0. However, this document provides distinct examples of each case to highlight the different requirements.

The following information about the policy decision point in Tivoli Federated Identity Manager must be made available to the enforcement point:

- The absolute URL of the Tivoli Federated Identity Manager trust service endpoint. (For example: `http://idp.tfim622.com:9080/TrustServer/SecurityTokenService`)
- The basic authentication user name and password for the Tivoli Federated Identity Manager trust service (if required).
- The ProviderID of the Tivoli Federated Identity Manager federation the client belongs to, which is used as the AppliesTo address for WS-Trust requests. Optionally, the enforcement point accepts a provider ID from the OAuth client as a request parameter to serve more than one federation concurrently.

Authorization decision request (OAuth 1.0)

Configuration

For OAuth 1.0 requests, the enforcement point must additionally know the Tivoli Federated Identity Manager OAuth 1.0 issuer address prefix (`urn:ibm:ITFIM:oauth:consumer:`).

HTTP request

When an OAuth 1.0 client retrieves a protected resource with its access token, it constructs a request similar to the following examples. These three examples are logically the same request.

OAuth 1.0 Example 1 (authorization header)

```
POST /fimivt/oauth/sfprotected.jsp?username=steve HTTP/1.1
  Host: idp.tfim622.com:9443
  Content-Type: application/x-www-form-urlencoded
  Authorization: OAuth oauth_consumer_key="YvMhsjmtEEi2gjv8Tqsl",
       oauth_token="YPxa78JggdW7hvcFRJph",
       oauth_signature_method="HMAC-SHA1",
       oauth_timestamp="1302828764",
       oauth_nonce="xWlY1PbsxjpiSZ4lVGVf",
       oauth_signature="Jpo6apiLE9hVSa8GqBSHUjFt7lg="
```

OAuth 1.0 Example 2 (post body)

```
POST /fimivt/oauth/sfprotected.jsp?username=steve HTTP/1.1
  Host: idp.tfim622.com:9443
  Content-Type: application/x-www-form-urlencoded

    oauth_consumer_key=YvMhsjmtEEi2gjv8Tqsl&oauth_token=YPxa78JggdW7hvcFRJph&
    oauth_signature_method=HMAC-SHA1&oauth_timestamp=1302828764&
    oauth_nonce=xWlY1PbsxjpiSZ4lVGVf&oauth_signature=Jpo6apiLE9hVSa8GqBSHUjFt7lg%3D
```

OAuth 1.0 Example 3 (query string)

```
POST /fimivt/oauth/sfprotected.jsp?username=steve&
    oauth_consumer_key=YvMhsjmtEEi2gjv8Tqsl&oauth_token=YPxa78JggdW7hvcFRJph&
    oauth_signature_method=HMAC-SHA1&oauth_timestamp=1302828764&
    oauth_nonce=xWlY1PbsxjpiSZ4lVGVf&oauth_signature=Jpo6apiLE9hVSa8GqBSHUjFt7lg%3D HTTP/1.1
  Host: idp.tfim622.com:9443
  Content-Type: application/x-www-form-urlencoded
```

Authorization decision request

The OAuth 1.0 enforcement point is responsible for the following actions:

- Transform the HTTP request into a WS-Trust SOAP message.
- Send the WS-Trust SOAP message to the Tivoli Federated Identity Manager STS for request validation.

The HTTP request is transformed into the following WS-Trust SOAP message:

OAuth 1.0 Token Validate Request (Request Security Token)

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <SOAP-ENV:Body>
        <wst:RequestSecurityToken xmlns:wst="http://schemas.xmlsoap.org/ws
        /2005/02/trust">
            <wst:RequestType xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
                http://schemas.xmlsoap.org/ws/2005/02/trust/Validate
            </wst:RequestType>
            <wst:Issuer xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
                <wsa:Address xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08
                /addressing">
                    urn:ibm:ITFIM:oauth:consumer:YvMhsjmtEEi2gjv8Tqsl
                </wsa:Address>
            </wst:Issuer>
            <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
                <wsa:EndpointReference xmlns:wsa="http://schemas.xmlsoap.org
                /ws/2004/08/addressing">
```

```
                    <wsa:Address>https://idp.tfim622.com:9443/sps/oauth10fed
                    /oauth10</wsa:Address>
                </wsa:EndpointReference>
            </wsp:AppliesTo>
            <wst:Base xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
                <stsuuser:STSUniversalUser xmlns:stsuuser="urn:ibm:names:ITFIM
                :1.0:stsuuser">
                    <stsuuser:Principal/>
                    <stsuuser:AttributeList/>
                    <stsuuser:ContextAttributes>
                        <stsuuser:Attribute name="oauth_token"
                          type="urn:ibm:names:ITFIM:oauth:param">
                            <stsuuser:Value>YPxa78JggdW7hvcFRJph</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="port"
                          type="urn:ibm:names:ITFIM:oauth:request">
                            <stsuuser:Value>9443</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="method"
                          type="urn:ibm:names:ITFIM:oauth:request">
                            <stsuuser:Value>POST</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="username"
                          type="urn:ibm:names:ITFIM:oauth:query:param">
                            <stsuuser:Value>steve</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="path"
                          type="urn:ibm:names:ITFIM:oauth:request">
                            <stsuuser:Value>/fimivt/oauth/sfprotected.jsp
                             </stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="scheme"
                          type="urn:ibm:names:ITFIM:oauth:request">
                            <stsuuser:Value>https</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="oauth_nonce"
                          type="urn:ibm:names:ITFIM:oauth:param">
                            <stsuuser:Value>xWlY1PbsxjpiSZ4lVGVf</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="host"
                          type="urn:ibm:names:ITFIM:oauth:request">
                            <stsuuser:Value>idp.tfim622.com</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="oauth_timestamp"
                          type="urn:ibm:names:ITFIM:oauth:param">
                            <stsuuser:Value>1302828764</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="oauth_consumer_key"
                          type="urn:ibm:names:ITFIM:oauth:param">
                            <stsuuser:Value>YvMhsjmtEEi2gjv8Tqsl</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="oauth_signature"
                          type="urn:ibm:names:ITFIM:oauth:param">
                            <stsuuser:Value>Jpo6apiLE9hVSa8GqBSHUjFt7lg=
                            </stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="oauth_signature_method"
                        type="urn:ibm:names:ITFIM:oauth:param">
                            <stsuuser:Value>HMAC-SHA1</stsuuser:Value>
                        </stsuuser:Attribute>
                    </stsuuser:ContextAttributes>
                </stsuuser:STSUniversalUser>
            </wst:Base>
        </wst:RequestSecurityToken>
    </soapenv:Body>
</soapenv:Envelope>
```

The following attributes are defined by the WS-Trust specification. They are used by Tivoli Federated Identity Manager to identify the federation associated with this request and to identify the OAuth 1.0 client.

The issuer address element (highlighted in *italics*) must be set to the Tivoli Federated Identity Manager OAuth 1.0 issuer address prefix (`urn:ibm:ITFIM:oauth:consumer:`) with the consumer key appended to the end.

The AppliesTo address element (highlighted with underline) must be set to the Provider ID of the OAuth 1.0 federation at Tivoli Federated Identity Manager. This element can be found on the federation properties page.

The following attributes are defined by the OAuth 1.0 protocol. The attributes that are not marked as optional are mandatory in the WS-Trust message that is sent to Tivoli Federated Identity Manager.

The attributes must be appended to the **ContextAttributes** section of the **STSUniversalUser** within the WS-Trust Request Security Token, and must have the type `urn:ibm:names:ITFIM:oauth:param`. If one of the mandatory parameters is missing from the request from the OAuth 1.0 client, the enforcement point does not validate the request with Tivoli Federated Identity Manager. It can instantly return an HTTP 400 Bad Request status code, and can also include a description of the error in the body.

- **consumer_key**
- **nonce**
- **realm** (optional)
- **signature**
- **signature_method**
- **timestamp**
- **token** (optional only if two-legged OAuth is enabled)
- **version** (optional)

The following attributes are defined by the OAuth 2.0 protocol. The attributes are mandatory in the WS-Trust message sent to Tivoli Federated Identity Manager. The attributes are also used to reconstruct the original signature base string URI of the request.

The attributes must be appended to the **ContextAttributes** section of the **STSUniversalUser** within the WS-Trust Request Security Token, and must have the type `urn:ibm:names:ITFIM:oauth:request`.

- **host** - host header from the request
- **method** - the HTTP method of the request (GET/POST)
- **path** - the requested path
- **port** - the port number on the host (Only if the request is received on a non-standard HTTP/HTTPS port.)
- **scheme** - (HTTP/HTTPS)

Any additional parameters that the OAuth 1.0 enforcement point finds in the request must be appended to the **ContextAttributes** section of the **STSUniversalUser** within the WS-Trust Request Security Token. Additional parameters can be a query, or post body parameters that are not of OAuth 1.0. The type value is determined by the following table.

HTTP parameter location	Attribute type value
URL Query String Parameters	`urn:ibm:names:ITFIM:oauth:query:param`
HTTP Request Body Parameters	`urn:ibm:names:ITFIM:oauth:body:param`

Post body parameters must be included only if the following conditions are met:

- The entity-body is single-part.

- The entity-body follows the encoding requirements of the "application/x-www-form-urlencoded" content-type as defined by [W3C.REC-html40-19980424].
- The HTTP request entity-header includes the "Content-Type" header field set to "application/x-www-form-urlencoded"

Authorization decision request (OAuth 2.0)

Configuration

For OAuth 2.0 requests, the enforcement point must additionally know the Tivoli Federated Identity Manager OAuth 2.0 issuer address prefix (`urn:ibm:ITFIM:oauth20:token:`).

HTTP request

When an OAuth 2.0 client retrieves a protected resource with its access token, it constructs a request similar to any of the following examples. Each of these three examples are logically the same request. All that differs is the transmission mechanism (HTTP header, query string, post body) for sending the OAuth 2.0 bearer access token:

OAuth 2.0 Example 1 (Access token in authorization header)

```
POST /fimivt/oauth/sfprotected.jsp HTTP/1.1
  Host: idp.tfim622.com:9443
  Authorization: Bearer YPxa78JggdW7hvcFRJph
  Content-Type: application/x-www-form-urlencoded

username=steve
```

OAuth 2.0 Example 2 (Access token in post body)

```
POST /fimivt/oauth/sfprotected.jsp HTTP/1.1
  Host: idp.tfim622.com:9443
  Content-Type: application/x-www-form-urlencoded

username=steve&access_token=YPxa78JggdW7hvcFRJph
```

OAuth 2.0 Example 3 (Access token in query string)

```
POST /fimivt/oauth/sfprotected.jsp?access_token=YPxa78JggdW7hvcFRJph HTTP/1.1
  Host: idp.tfim622.com:9443
  Content-Type: application/x-www-form-urlencoded

username=steve
```

Authorization decision request

The OAuth 2.0 enforcement point is responsible for the following actions:

- Transform HTTP requests into a WS-Trust SOAP message.
- Send the WS-Trust SOAP message to the Tivoli Federated Identity Manager STS for request validation.

The HTTP request is transformed into the following WS-Trust SOAP message:

OAuth 2.0 Token Validate Request (Request Security Token)

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <SOAP-ENV:Body>
        <wst:RequestSecurityToken xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
            <wst:RequestType xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
                http://schemas.xmlsoap.org/ws/2005/02/trust/Validate
            </wst:RequestType>
            <wst:Issuer xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
                <wsa:Address xmlns:wsa="http://schemas.xmlsoap.org/ws/2004
                /08/addressing">
                    urn:ibm:ITFIM:oauth20:token:bearer
                </wsa:Address>
            </wst:Issuer>
```

```
            <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004
            /09/policy">
                <wsa:EndpointReference xmlns:wsa="http://schemas.xmlsoap
                .org/ws/2004/08/addressing">
                    <wsa:Address>https://idp.tfim622.com:9443/sps/oauth20fed/oauth20</wsa:Address>
                </wsa:EndpointReference>
            </wsp:AppliesTo>
            <wst:Base xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
                <stsuuser:STSUniversalUser xmlns:stsuuser="urn:ibm:names:ITFIM:1.0:stsuuser">
                    <stsuuser:Principal/>
                    <stsuuser:AttributeList/>
                    <stsuuser:ContextAttributes>
                        <stsuuser:Attribute name="access_token"
                          type="urn:ibm:names:ITFIM:oauth:param">
                            <stsuuser:Value>YPxa78JggdW7hvcFRJph</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="username"
                          type="urn:ibm:names:ITFIM:oauth:body:param">
                            <stsuuser:Value>steve</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="port"
                          type="urn:ibm:names:ITFIM:oauth:request">
                            <stsuuser:Value>9443</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="method"
                          type="urn:ibm:names:ITFIM:oauth:request">
                            <stsuuser:Value>POST</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="path"
                          type="urn:ibm:names:ITFIM:oauth:request">
                            <stsuuser:Value>/fimivt/oauth/sfprotected.jsp</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="scheme"
                          type="urn:ibm:names:ITFIM:oauth:request">
                            <stsuuser:Value>https</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="host"
                          type="urn:ibm:names:ITFIM:oauth:request">
                            <stsuuser:Value>idp.tfim622.com</stsuuser:Value>
                        </stsuuser:Attribute>
                    </stsuuser:ContextAttributes>
                </stsuuser:STSUniversalUser>
            </wst:Base>
        </wst:RequestSecurityToken>
    </soapenv:Body>
</soapenv:Envelope>
```

The following attributes are defined by the WS-Trust specification. They are used by Tivoli Federated Identity Manager to identify the federation associated with this request and to identify the type of OAuth 2.0 access token being used.

- The Issuer address element (highlighted in **bold**) must be set to the Tivoli Federated Identity Manager OAuth 2.0 issuer address prefix (`urn:ibm:ITFIM:oauth20:token:`). The token type must be appended at the end, separated by a colon. Currently, the only token type supported is *bearer*, which means the issuer address must be set to `urn:ibm:ITFIM:oauth20:token:bearer`.
- The AppliesTo address element (highlighted in *italics*) must be set to the Provider ID of the OAuth federation at Tivoli Federated Identity Manager. This element can be found on the federation properties page.

The **access_token** attribute with type `urn:ibm:names:ITFIM:oauth:param` is mandatory in the WS-Trust message sent to Tivoli Federated Identity Manager. It must be appended to the **ContextAttributes** section of the **STSUniversalUser** within the WS-Trust Request Security Token.

If **access_token** attribute is missing from the request from the OAuth 2.0 client, the enforcement point does not validate the request with Tivoli Federated Identity Manager. It can instantly return an HTTP 400 Bad Request status code and optionally can include a description of the error in the body.

Note: If the access token is included in the authorization header in the `Authorization: Bearer <token>` format, the token must still be added to the **ContextAttributes** section of the STSUU. The same format must be used as if the access token was sent through a query string or post body.

The following attributes are *not* mandatory in the WS-Trust message sent to Tivoli Federated Identity Manager for OAuth 2.0. However, they might be useful to a custom mapping rule that is being ran by Tivoli Federated Identity Manager.

The following attributes must be appended to the **ContextAttributes** section of the **STSUniversalUser** within the WS-Trust Request Security Token, and must have the type `urn:ibm:names:ITFIM:oauth:request`.

- **method** - the HTTP method of the request (GET/POST)
- **scheme** - (http/https)
- **host** - host header from the request
- **port** - the port number on the host (only if it is a non-standard port. For example, not 80 if the method is HTTP or not 443 if the method is HTTPS)
- **path** - the requested path

Any additional parameters that the OAuth 2.0 enforcement point finds in the request, such as query or post body parameters that are not of OAuth 2.0, must be appended to the **Context Attribute** section of the **STSUniversalUser** within the WS-Trust Request Security Token. The type value is determined by the following table.

In an OAuth 1.0 request, additional request parameters are required to be appended to the **STSUniversalUser** to calculate the correct request signature. In OAuth 2.0 requests, these parameters are NOT required. However, they might be useful to a custom mapping rule being ran by Tivoli Federated Identity Manager, and so must be appended.

HTTP Parameter Location	Attribute Type Value
URL Query String Parameters	`urn:ibm:names:ITFIM:oauth:query:param`
HTTP Request Body Parameters	`urn:ibm:names:ITFIM:oauth:body:param`

Post body parameters must be included only if the following conditions are met:

- The entity-body is single-part.
- The entity-body follows the encoding requirements of the “application/x-www-form-urlencoded”content-type as defined by [W3C.REC-html40-19980424].
- The HTTP request entity-header includes the “Content-Type” header field set to “application/x-www-form-urlencoded”.

Authorization decision response (OAuth 1.0 and OAuth 2.0)

The SOAP message response from Tivoli Federated Identity Manager (regardless of OAuth version) echoes all the context attributes sent in the original request and some extra response context attributes.

OAuth Token Validate Response (RSTR)

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <SOAP-ENV:Body>
        <wst:RequestSecurityTokenResponse wsu:
```

```
        Id="uuid56a54e7c-012f-1207-9133-c24cad886d75"
        xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401
        -wss-wssecurity-utility-1.0.xsd">
            <wsp:AppliesTo xmlns:wsa="http://schemas.xmlsoap.org/ws/2004
            /08/addressing"
            xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
                <wsa:EndpointReference>
                    <wsa:Address>https://idp.tfim622.com:9443/sps/oauth10fed
                    /oauth10</wsa:Address>
                </wsa:EndpointReference>
            </wsp:AppliesTo>
            <wst:RequestedSecurityToken>
                <stsuuser:STSUniversalUser xmlns:stsuuser="urn:ibm:names
                :ITFIM:1.0:stsuuser">
                    <stsuuser:Principal/>
                    <stsuuser:AttributeList/>
                    <stsuuser:ContextAttributes>
                        <stsuuser:Attribute name="authorized"
                        type="urn:ibm:names:ITFIM:oauth:response:decision">
                            <stsuuser:Value>TRUE</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="expires" type="urn:ibm
                        :names:ITFIM:oauth:response:decision">
                            <stsuuser:Value>2011-04-22T00:52:18Z</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="scope" type="urn:ibm
                        :names:ITFIM:oauth:response:attribute">
                                <stsuuser:Value>email</stsuuser:Value>
                                <stsuuser:Value>first</stsuuser:Value>
                                <stsuuser:Value>last</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="username" type="urn:ibm
                        :names:ITFIM:oauth:response:attribute">
                            <stsuuser:Value>wasadmin</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="username_is_self"
                        type="urn:ibm:names:ITFIM:oauth:response:attribute">
                            <stsuuser:Value>FALSE</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="oauth_token" type="urn:ibm
                        :names:ITFIM:oauth:response:attribute">
                            <stsuuser:Value>YPxa78JggdW7hvcFRJph</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="recovered_state" type="urn:ibm
                        :names:ITFIM:oauth:response:attribute">
                            <stsuuser:Value>State storage time was:
                            2011-04-15T00:52:18Z</stsuuser:Value>
                        </stsuuser:Attribute>
                        <stsuuser:Attribute name="state_id" type="urn:ibm
                        :names:ITFIM:oauth:state">
                            <stsuuser:Value>2cJsZ3QhXV5rDVZHNePp</stsuuser:Value>
                        </stsuuser:Attribute>
                    </stsuuser:ContextAttributes>
                    <stsuuser:AdditionalAttributeStatement id=""/>
                </stsuuser:STSUniversalUser>
            </wst:RequestedSecurityToken>
            <wst:Status>
                <wst:Code>http://schemas.xmlsoap.org/ws/2005/02/trust/status
                /valid</wst:Code>
            </wst:Status>
        </wst:RequestSecurityTokenResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

The following context attributes returned to the enforcement point by Tivoli Federated Identity Manager relate to the authorization decision. It also has the attribute type `urn:ibm:names:ITFIM:oauth:response:decision` highlighted in *italics* in the previous RSTR example. It is up to the enforcement point to decide whether to down-stream these attributes to the OAuth protected resource.

These attributes are primarily for the use of the enforcement point itself to determine the authorization status.

Context attributes	Description
authorized	The value is set to TRUE if the OAuth request is valid and authorized; FALSE if otherwise.
expires	The UTC time that the access token used in the request is no longer valid. This attribute is not present for OAuth 1.0 two-legged as that flow does not use an access token.

The following context attributes returned to the enforcement point by Tivoli Federated Identity Manager must be down-streamed from the enforcement point to the OAuth protected resource. They might be appended to the original HTTP request in any way deemed suitable by the enforcement point and the protected resource. This way, the protected resource can retrieve them (for example, as additional HTTP headers).

These context attributes have the attribute type `urn:ibm:names:ITFIM:oauth:response:attribute` (highlighted in **bold** in the previous RSTR example).

Custom mapping rules that are ran after the OAuth trust chain might also append attributes with this type. Therefore, any attribute with this type must be down-streamed to the requested protected resource.

Context attributes	Description
access_token (OAuth 2.0)	The OAuth access token used in the protected resource request.
client_type (OAuth 2.0)	The type of client that this token was issued to, can be either public or confidential. Public clients are clients that do not have client credentials and therefore cannot authenticate to the authorization server.
oauth_token_client_id (OAuth 2.0)	The unique identifier of the client to which the current access token was issued. This parameter is not returned for OAuth 1.0 requests as the consumer key is sent in the initial request. Therefore, it is still in the STSUU with the name `consumer_key` and the type `urn:ibm:names:ITFIM:oauth:request`.
oauth_token (OAuth 1.0)	The OAuth access token used in the protected resource request. This attribute is not present for OAuth 1.0 two-legged as that flow does not use an access token.
scope	A list of strings that represents the resource scope authorized by the user at the OAuth resource owner authorization step. The OAuth protected resource can use this attribute to determine which resources to return in the response. This attribute is only present for OAuth flows that include a user authorization step.
username	The name of the user who authorized the OAuth token to access their protected resources on their behalf. With OAuth flows that do not involve a separate resource owner, this value is the client identifier.

Additional attributes with the type `urn:ibm:names:ITFIM:oauth:response:attribute` are sometimes appended by a custom mapping rule, such is the case with **recovered_state** and **username_is_self** in the example.

The **state_id** context attribute returned to the enforcement point by Tivoli Federated Identity Manager is used by a custom mapping rule that is ran after the OAuth trust chain. It has the attribute type `urn:ibm:names:ITFIM:oauth:state` (highlighted with an underline) and can be ignored by the enforcement point.

The **state_id** attribute is a unique identifier for the current OAuth token used to store state information.

If the **state_id** attribute is required by the OAuth protected resource, a custom mapping rule can be implemented to make a copy of this attribute. The type can be changed to `urn:ibm:names:ITFIM:oauth:response:attribute` from the custom mapping rule to ensure that it is down-streamed to the resource.

Error responses

An OAuth enforcement point can do as much or as little validation of OAuth requests as it prefers. Any validation it performs is repeated by Tivoli Federated Identity Manager. Doing some validation before sending an authorization request to Tivoli Federated Identity Manager might improve performance. The following validation must be performed by the enforcement point before sending a request to Tivoli Federated Identity Manager.

- Validate that some OAuth data is present. If not, return an HTTP 401 Unauthorized status code.
- Validate that none of the required OAuth parameters are missing. If any of them are not present in the request, return an HTTP 400 Bad Request status code.
- Validate that none of the required OAuth parameters occur more than once in the request. They must also occur only in the one component of the request; for example, the query string or the authorization header. If the validation fails, return an HTTP 400 Bad Request status code.

The enforcement point must return an HTTP 401 Unauthorized status code to the OAuth client if the following scenarios occur:

- The enforcement point sends an authorization request to Tivoli Federated Identity Manager.
- The enforcement point receives a SOAP message with an authorized context attribute that has a value of FALSE.

The enforcement point must return an HTTP 503 Service Unavailable status code to the OAuth client if the following scenarios occur:

- Tivoli Federated Identity Manager encounters an error.
- Tivoli Federated Identity Manager does not return a constructed SOAP message or the SOAP message does not contain an authorized context attribute.

The enforcement point might also optionally return a WWW-Authenticate HTTP header to indicate its support for OAuth.

Flow chart

The following chart shows the expected workflow of an OAuth authorization enforcement point.

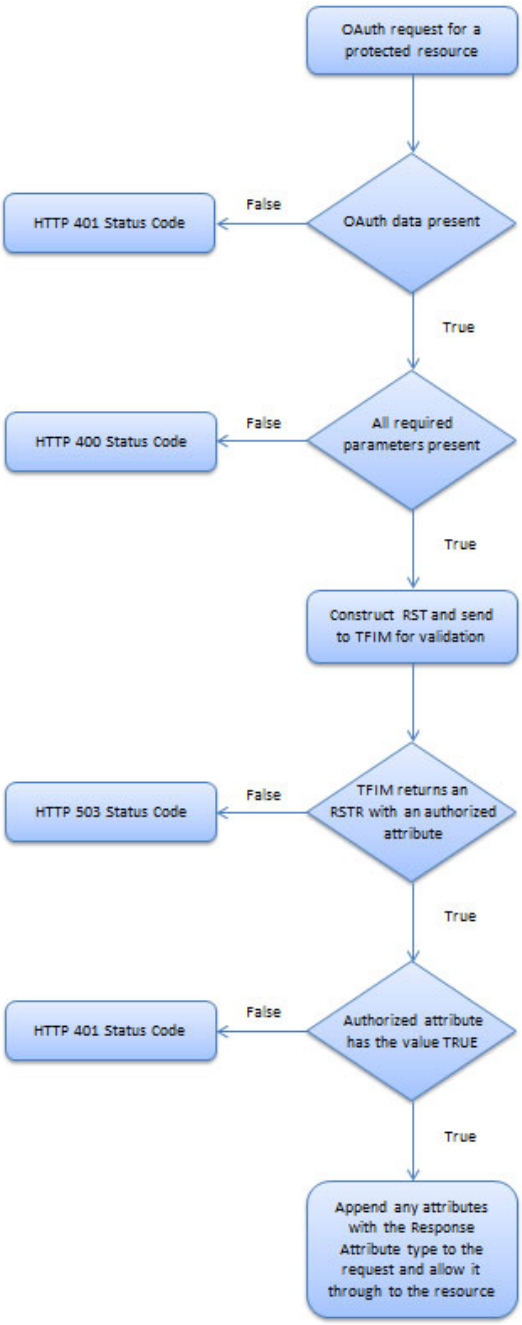


Figure 44. OAuth authorization enforcement point workflow

OAuth Trust Association Interceptor and Servlet Filter custom properties

You must customize the property of the WebSphere Trust Association Interceptor (TAI) or the Servlet Filter (SF) component as an enforcement point to your OAuth federation.

The enforcement points properties are used to call the Tivoli Federated Identity Manager Security Token Service (STS) for validation and authorization.

This topic lists the configuration properties of the WebSphere TAI and SF components for both OAuth 1.0 and OAuth 2.0 federations.

Table 118. Trust association interceptor and servlet filter properties

Property Name	Description	Example

FederationIdRequestParameterName	Specifies the name of the request parameter. (**Optional**)
	
	The value of the corresponding runtime request parameter is used as the AppliesTo address in calls to the STS. It must match the Provider ID of the federation for which the OAuth client is a member.
	
	Customization of the request parameter name can be done through this property. You can modify the protected resource URL to include a query string parameter with:
	• a name matching the value of this configuration property, and
	• a value matching the Provider ID of the federation that the OAuth client is a member of.
	
	This property makes it possible for one enforcement point to service requests for more than one federation at a time.
	
	If this property is not supplied, the value of the **DefaultFederationId** property is used as the static Provider ID value in calls to the STS.
		`FederationId`
		
		Example usage:
		`sfprotected.jsp?FederationId=https://server.oauth.com/FIM/MySocialNetwork/oauth20`

DefaultFederationId	Sets the default value of the Federation Provider ID used for communication with the STS. (**Required**) It is used when: • the **FederationIdRequestParameterName** property is not provided. • there is no request parameter in the incoming request with a name matching the value of the **FederationIdRequestParameterName** property.	`https://server.oauth.com/FIM/MySocialNetwork/oauth20`

DefaultMode	Determines how to validate a request against either OAuth 1.0 or OAuth 2.0. (**Required**)
	
	It is used to distinguish the different versions of an OAuth protocol. The supported token type for an OAuth 2.0 protocol is also specified in the value.
	
	It is used when:
	• the **ModeParameterName** property is not provided.
	• there is no request parameter in the incoming request with a name matching the value of the **ModeParameterName** property.
		For OAuth 1.0:
		`OAuth10`
		
		For OAuth 2.0:
		`OAuth20Bearer`

Table 118. Trust association interceptor and servlet filter properties (continued)

Property Name	Description	Example
ModeParameterName	Specifies the name of the request parameter. (**Optional**) The request parameter name can be customized to carry the mode value. You can modify the protected resource URL to include a query string parameter with: • a name matching the value of this configuration property, and • a value matching the Provider ID of the federation that the OAuth client is a member of. A single policy enforcement point (PEP) can service both OAuth 1.0 and OAuth 2.0 federations at the same time if these conditions occur: • the **ModeParameterName** property used with the **FederationIdRequestParameterName** property. • the OAuth clients send the FederationId and mode parameters in the request for the protected resource. If this property is not supplied, the value of the **DefaultMode** property is used to determine whether to validate the incoming request as OAuth 1.0 or OAuth 2.0.	mode Example usage: **For OAuth 1.0:** sfprotected.jsp?mode=OAuth10 **For OAuth 2.0:** sfprotected.jsp?mode=OAuth20Bearer
OAuthRealm	Specifies the realm in the WWW-Authenticate header that is sent back to a request that does not contain an authorized OAuth token. (**Required**)	https://server.oauth.com/FIM/
OAuthTokenCacheSize	Specifies the maximum size of a cache. This cache is used to map OAuth 2.0 bearer tokens to results, such as token existence and expiry time, from the Security Token Services call. (**Optional**)	2
PointOfContact	Specifies the point of contact URL for clients of the server. The IBM HTTP Server or WebSEAL can be used in front of WebSphere, in which case the URL is going to look different from the example. (**Optional**)	https://server.oauth.com/FIM/
STSEndpoint	Specifies the WS-Trust 1.2 endpoint of the STS. (**Optional**)	https://server.oauth.com/FIM/
STSUsername	Specifies the basic authentication user name for communication with the STS. (**Required** depending on the security of the TrustClientInternalRole in the ITFIMRuntime.)	wasadmin
STSPassword	Specifies the basic authentication password for communication with the STS. (**Required** depending on the security of the TrustClientInternalRole in the ITFIMRuntime.)	password
STSSSLConfiguration	Specifies a WebSphere SSL configuration object that contains keys suitable for server, and client if necessary, SSL authentication of the WS-Trust URL. (**Required** only if HTTPS URL to STS endpoint is used.)	mysslcfg
URIPrefix	Specifies a string that is compared with the start of the request URI to see if the TAI or servlet filter must protect this request. To protect ALL resources, use **/**. (**Required**)	/snoop

OAuth EAS stanza reference

This topic contains the stanza reference for the OAuth EAS configuration.

- [aznapi-external-authzn-services] stanza
- [azn-decision-info] stanza
- [aznapi-configuration] stanza
- [oauth] stanza

[aznapi-external-authzn-services] stanza

policy-trigger

Syntax

```
policy-trigger = plug-in_location [-weight N [& plug-in_parameters]]
```

Description

Defines the external authorization service.

Options

policy-trigger

Any string that is recognized as a valid key name. Stanza key names cannot contain white space or the open bracket ([) and close bracket (]) characters. The bracket characters are used to define new stanza names. The policy-trigger is case sensitive for action set definitions because the actions themselves are case sensitive. However, the policy-trigger is case insensitive if the trigger is a protected object policy (POP) attribute.

plug-in_location

The path name to the shared library or DLL module that contains the implementation of the plug-in for the specified policy trigger. The path name can be in a truncated form if the external authorization service is to be loaded by clients on multiple platforms. In this case, the service dispatcher searches for the plug-in using platform-specific prefixes and suffixes to match DLL names.

The name of the OAuth EAS plug-in is **amwoautheas** and its library is contained in the `pdwebrte/lib` directory. For example:

```
/opt/pdwebrte/lib/libamwoautheas.so
```

N

The weight parameter is an unsigned **size_t** value and is optional. The value signifies the weight that any decision returned by this external authorization service is given in the entire decision process.

plug-in_parameters

Optionally, the external authorization service can be passed additional initialization information in the form of arguments. The arguments must be preceded by the ampersand "&". The authorization service takes the remainder of the string following the ampersand &, breaks the string up into white space separated tokens, and passes the tokens directly to the administration service's initialization interface, `azn_svc_initialize()`, in the **argv** array parameter. The number of strings in the **argv** array is indicated by the **argc** function parameter.

A single parameter is required by the OAuth EAS. This parameter corresponds to the name of the OAuth EAS configuration file. That is, the file that contains the **[oauth-eas]** stanza and the corresponding **[tfim-cluster:<cluster>]** stanza.

Usage

This stanza entry is required when configuring OAuth EAS authentication.

Default value

None.

Example

The following example is an operation-based trigger with a user-defined action group of Printer and the actions rxT within that group. To specify the primary action group you would specify only :rxT. The primary action group can be represented with an empty action group name or the string primary can be used explicitly. All lowercase letters are required if primary is used explicitly. Any policy-trigger that does not contain a colon (**:**) character is considered to be a POP attribute name.

```
Printer:rxT = eas_plugin -weight 60 & -server barney
```

The following example is for a POP attribute trigger called **webseal_pop_trigger**. When a POP that contains a reference to this string is encountered, the appropriate external authorization service is called to take part in the access decision.

```
webseal_pop_trigger = eas_plugin_2 -weight 70 & -hostname fred
```

Note that in order for the above POP attribute trigger to work, POP configuration must have been completed previously by the secure domain administrator, using the **pdadmin pop** commands.

The following is an example configuration for the OAuth EAS, where the file `/opt/pdweb/etc/oauth_eas.conf` contains the **[oauth-eas]** stanza and the corresponding **[tfim-cluster:<cluster>]** stanza. This example is entered as one line in the WebSEAL configuration file:

```
webseal_pop_trigger = /opt/pdwebrte/lib/libamwoautheas.so & /opt/pdweb/etc
/oauth_eas.conf
```

[azn-decision-info] stanza

azn-decision-info

Syntax

```
<attr-name> = <http-info>
```

Description

This stanza defines any extra information that is available to the authorization framework when making authorization decisions. This extra information can be obtained from various elements of the HTTP request, namely:

- HTTP method
- HTTP scheme
- Request URI
- HTTP headers
- POST data

If the requested element is not in the HTTP request, no corresponding attribute is added to the authorization decision information.

Options

<attr-name>
The name of the attribute that contains the HTTP information.

<http-info>
The source of the information. It can be one of the following values:

- method
- scheme
- uri
- header:<header-name>
- post-data:<post-data-name>

Usage

This stanza entry is required when configuring OAuth EAS authentication and must contain the following elements:

```
HTTP_REQUEST_METHOD = method
HTTP_REQUEST_SCHEME = scheme
HTTP_REQUEST_URI = uri
HTTP_HOST_HDR = header:host
HTTP_CONTENT_TYPE_HDR = header:content-type
HTTP_TRANSFER_ENCODING_HDR = header:transfer-encoding
HTTP_AZN_HDR = header:authorization
```

Default value

N/A

Example

```
HTTP_REQUEST_METHOD = method
HTTP_HOST_HEADER= header:Host
```

[aznapi-configuration] stanza

resource-manager-provided-adi

Syntax

```
resource-manager-provided-adi = prefix
```

Description

A list of string prefixes that identify Access Decision Information (ADI) to be supplied by the resource manager (in this case, WebSEAL).

Options

prefix The default settings tell the authorization engine that when it requires ADI with the prefixes AMWS_hd_, AMWS_qs_ ,or AMWS_pb_ to evaluate a boolean authorization rule, and the ADI is not available in either the credential or application context passed in with the access decision call, that the engine should fail the access decision and request that the resource manager retry the request and provide the required data in the application context of the next request.

Usage

This stanza entry is required when configuring OAuth EAS authentication.

Default value

```
AMWS_hd_, AMWS_pb_, AMWS_qs_
```

Example

```
resource-manager-provided-adi = AMWS_hd_
resource-manager-provided-adi = AMWS_pb_
resource-manager-provided-adi = AMWS_qs_
```

[oauth-eas] stanza

You can configure the [oauth-eas] stanza to support OAuth authorization decisions as part of WebSEAL requests.

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.

Note: This stanza can be included in a separate configuration file that is specified for **amwoautheas** in the **[aznapi-external-authzn-services]** stanza.

Sample EAS configuration data

The **[oauth-eas]** stanza contains the configuration details for the OAuth EAS.

The example shows the OAuth EAS configuration data with the default federation id and mode set for an OAuth 1.0 federation.

```
[oauth-eas]

# The maximum number of OAuth 2.0 bearer token authorization decisions to cache.
# This EAS has a built in cache for storing authorization decisions so that
# repeated use of the same OAuth 2.0 bearer token does not require repeated
# requests to TFIM. Bearer token decisions can be cached because they do not
# require signing of the request, unlike OAuth 1.0 requests. The lifetime of the
# cache entry is based on the Expires attribute returned by TFIM. If this
# attribute is not returned, the decision will not be cached.
#
# This EAS implements a Least Recently Used cache, meaning the decision
# associated with the least recently used bearer token will be forgotten when a
# new bearer token decision is cached. A cache-size of 0 will disable caching of
# authorization decisions
cache-size = 0

# The Provider ID of the default OAuth federation at TFIM. If a Provider ID
# is not provided in the request using the fed-id-param option, this provider
# ID will be used for OAuth requests. The Provider ID of a federation can be
# found on the federation properties page.
default-fed-id = https://server.oauth.com/FIM/sps/oauthfed/oauth10

# The name of the request parameter that can be used to override the
# default-fed-id option configured above. By deleting this configuration
# option, you can enforce that the default fed id is always used.
fed-id-param = FederationId

# The default OAuth mode that this EAS will operate under. It affects the
# validation of request parameters, as well as the construction of the RST
```

```
# sent to TFIM. The default mode can be overriden for an individual request
# by providing a valid mode value [OAuth10|OAuth20Bearer] in a request
# parameter with the name specified in the mode-param option below.
default-mode = OAuth10

# The name of the request parameter that can be used to override the
# default-mode option configured above. By deleting this configuration
# option, you can enforce that the default mode is always used.
mode-param = mode

# The name of the OAuth realm which will be used in a 401 request
# for OAuth data.
realm-name = oauth-realm

# The name of the file which contains the body used when constructing a
# '400 Bad Request' response.  This response will be generated when
# required OAuth elements are missing from a request.
bad-request-rsp-file = /EAS/oauth_eas/400.html

# The name of the file which contains the body used when constructing a
# '401 Unauthorized' response.  This response will be generated when:
#     - all OAuth data is missing from a request, or
#     - the OAuth data fails validation.
unauthorized-rsp-file = /EAS/oauth_eas/401.html

# The name of the file which contains the body used when constructing a
# '502 Bad Gateway' response.  This response will be generated when
# TFIM fails to process the request.
bad-gateway-rsp-file = /EAS/oauth_eas/502.html

# The name of the TAM trace component which is used by the EAS.
trace-component = pdweb.oauth

# Should the native TAM ACL policy still take affect, in addition to the
# OAuth authorization?
apply-tam-native-policy = false

# The name of the TFIM cluster which houses this OAuth service.  There should
# also be a corresponding [tfim-cluster:<cluster>] stanza which contains the
# definition of the cluster.
cluster-name = oauth-cluster

[tfim-cluster:oauth-cluster]

#
# This stanza contains definitions for a particular cluster of TFIM
# servers.
#

#
# A specification for the server which is used when communicating with a
# single TFIM server which is a member of this cluster.  Values for this
# entry are defined as follows:
#
#       {[0-9],}<URL>
#
# Where the first digit (if present) represents the priority of the server
# within the cluster (9 being the highest, 0 being lowest).  If the priority
# is not specified, a priority of 9 is assumed.  The <URL> can be any
# well-formed HTTP or HTTPS URL.
#
# Multiple server entries can be specified for failover and load balancing
# purposes.  The complete set of these server entries defines the
# membership of the cluster for failover and load balancing.
```

```
#
# server = 9,http://tfim.example.com/TrustServerWST13/services/RequestSecurityToken

...
```

cache-size

Syntax

```
cache-size = cache_size
```

Description

The maximum number of OAuth 2.0 bearer token authorization decisions to cache. This cache stores authorization decisions so that repeated use of the same token does not require repeated requests to Tivoli Federated Identity Manager. A `cache-size` of 0 will disable caching of authorization decisions.

Options

cache_size
The size of the OAuth token cache.

Usage

This stanza entry is required when configuring OAuth EAS authentication.

Default value

None.

Example

```
cache-size = 2
```

default-fed-id

Syntax

```
default-fed-id = provider_id
```

Description

The Provider ID of the default OAuth federation in Tivoli Federated Identity Manager. If a Provider ID is not provided in the request using the **fed-id-param** option, this provider ID is used for OAuth requests. The Provider ID of a federation can be found on the federation properties page.

Options

provider_id
The Provider ID of the OAuth federation.

Usage

This stanza entry is required when configuring OAuth EAS authentication.

Default value

None.

Example

```
default-fed-id = https://server.oauth.com/FIM/MySocialNetwork/oauth20
```

fed-id-param

Syntax

```
fed-id-param = request_param_name
```

Description

The name of the request parameter that can be used to override the **default-fed-id** option. The value must match the Provider ID of the federation for which the OAuth client is a member.

Delete this configuration option to enforce that the default fed id is always used.

Options

request_param_name
The name of the request parameter.

Usage

This stanza entry is optional. If it is not supplied, the value of the **default-fed-id** option is used as the static Provider ID value in calls to the STS.

Default value

None.

Example

```
fed-id-param = FederationId
```

default-mode

Syntax

```
default-mode = mode_value
```

Description

The default OAuth mode that the EAS operates under. It affects the validation of request parameters, as well as the construction of the RST sent to Tivoli Federated Identity Manager. Provide a valid mode value in a request parameter with the name specified in the **mode-param** option to override the default mode for an individual request.

Options

mode_value
The valid mode value for the OAuth 1.0 protocol is `OAuth10`, while the valid mode value for the OAuth 2.0 protocol is `OAuth20Bearer`.

Usage

This stanza entry is required when configuring OAuth EAS authentication.

Default value

None.

Example

For OAuth 1.0:

```
default-mode = OAuth10
```

For OAuth 2.0:

```
default-mode = OAuth20Bearer
```

mode-param

Syntax

```
mode-param = request_param_name
```

Description

The name of the request parameter that can be customized to carry the mode value. This configuration option can be used to override the **default-mode** option. Delete this configuration option to enforce that the default mode is always used.

Options

request_param_name
The name of the request parameter.

Usage

This stanza entry is optional. If it is not supplied, the value of the **default-mode** option is used to determine whether or not to validate the incoming request as OAuth 1.0 or OAuth 2.0.

Default value

None.

Example

```
mode-param = mode
```

realm-name

Syntax

```
realm-name = realm_name
```

Description

The name of the OAuth realm that is used in a 401 request for OAuth data.

Options

realm_name
The name of the OAuth realm.

Usage

This stanza entry is required when configuring OAuth EAS authentication.

Default value

None.

Example

```
realm-name = realmOne
```

bad-request-rsp-file

Syntax

```
bad-request-rsp-file = file_name
```

Description

The fully qualified name of the file that contains the body that is used when constructing a '400 Bad Request' response. This response is generated when required OAuth elements are missing from a request.

Options

file_name
The name of the 400 Bad Request response file.

Usage

This stanza entry is required when configuring OAuth EAS authentication.

Default value

None.

Example

```
bad-request-rsp-file = /tmp/bad_rqst.html
```

Here is an example of the HTML response file:

```
<html>
<body>
400 Bad Request
</body>
</html>
```

unauthorized-rsp-file

Syntax

```
unauthorized-rsp-file = file_name
```

Description

The fully qualified name of the file that contains the body that is used when constructing a '401 Unauthorized' response. This response is generated when:

- All OAuth data is missing from a request, or
- The OAuth data fails validation.

Options

file_name
The name of the 401 Unauthorized response file.

Usage

This stanza entry is required when configuring OAuth EAS authentication.

Default value

None.

Example

```
unauthorized-rsp-file = /tmp/unauth_response.html
```

Here is an example of the HTML response file:

```
<html>
<body>
401 Unauthorized
</body>
</html>
```

bad-gateway-rsp-file

Syntax

```
bad-gateway-rsp-file = file_name
```

Description

The fully qualified name of the file that contains the body that is used when constructing a '502 Bad Gateway' response. This response is generated when Tivoli Federated Identity Manager fails to process the request.

Options

file_name
The name of the 502 Bad Gateway response file.

Usage

This stanza entry is required when configuring OAuth EAS authentication.

Default value

None.

Example

```
bad-gateway-rsp-file = /tmp/bad_gateway.html
```

Here is an example of the HTML response file:

```
<html>
<body>
502 Bad Gateway
</body>
</html>
```

trace-component

Syntax

```
trace-component = component_name
```

Description

The name of the Tivoli Access Manager trace component that is used by the EAS.

Options

component_name
The name of the Tivoli Access Manager trace component.

Usage

This stanza entry is required when configuring OAuth EAS authentication.

Default value

None.

Example

```
trace-component = pdweb.oauth
```

apply-tam-native-policy

Syntax

```
apply-tam-native-policy = <true | false>
```

Description

Determines whether the native Tivoli Access Manager ACL policy still takes affect, in addition to the OAuth authorization.

Options

true The native Tivoli Access Manager ACL policy still takes affect.

false The native Tivoli Access Manager ACL policy does not take affect.

Usage

This stanza entry is required when configuring OAuth EAS authentication.

Default value

None.

Example

```
apply-tam-native-policy = false
```

cluster-name

Syntax

```
cluster-name = cluster_name
```

Description

The name of the Tivoli Federated Identity Manager cluster that hosts this OAuth service. There should also be a corresponding **[tfim-cluster:<cluster>]** stanza, which contains the definition of the cluster.

Options

cluster_name
The name of the Tivoli Federated Identity Manager cluster where the OAuth service is hosted.

Usage

This stanza entry is required when configuring OAuth EAS authentication.

Default value

None.

Example

```
cluster-name = oauth-cluster
```

For this example, there needs to be a corresponding **[tfim-cluster:oauth-cluster]** stanza to define the cluster.

OAuth 1.0 and OAuth 2.0 template pages for trusted clients management

Tivoli Federated Identity Manager provides an HTML page template which resource owners can use to show and manage trusted clients information for OAuth 1.0 and OAuth 2.0 federations.

There are different trusted clients management template pages for each OAuth protocol. These pages look the same, and use the same replacement macros. The template pages for OAuth 1.0 and OAuth 2.0 are both named as `clients_manager.html`.

The resource owner establishes the OAuth clients through the `user_consent.html` page during authorization requests.

The templates have the following replacement macros:

@USERNAME@
This macro is replaced with the Tivoli Federated Identity Manager user name.

@OAUTH_CLIENT_COMPANY_NAME@
A multi-valued macro that belongs inside a `[RPT trustedClients]` repeatable replacement list. The values are replaced with the name of the company that requests access to the protected resource.

@PERMITTED_SCOPES@
A multi-valued macro that belongs inside a `[RPT trustedClients]` repeatable replacement list. The values are replaced with the token scopes to which the OAuth client has access.

@DENIED_SCOPES@
A multi-valued macro that belongs inside a [RPT trustedClients] repeatable replacement list. The values are replaced with the token scopes to which the OAuth client does *not* have access.

@OAUTH_CUSTOM_MACRO@
A multi-valued macro that belongs inside a [RPT trustedClients] repeatable replacement list. The values are replaced with trusted client information that contains additional information about an authorized OAuth client.

@OAUTH_CLIENTMANAGERURL@
A multi-valued macro that belongs inside a [RPT trustedClients] repeatable replacement list. The values are replaced with the endpoint of the trusted clients manager.

@UNIQUE_ID@
A multi-valued macro that belongs inside a [RPT trustedClients] repeatable replacement list. The values are replaced with a unique identifier that identifies the trusted clients information for each entry in the list.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
  <title>OAuth Client Manager</title>
 </head>
 <body>
  Username: <b>@USERNAME@</b>
  <p />
  Trusted Clients<br />
  <table border="1">
    <tr><td>Client</td><td>Permitted Scopes</td><td>Denied Scopes</td>
        <td>Additional Information</td><td>Action</td></tr>
<!-- START NON-TRANSLATABLE -->
   [RPT trustedClients]
<!-- END NON-TRANSLATABLE -->
    <tr>
      <td>@OAUTH_CLIENT_COMPANY_NAME@</td>
      <td>@PERMITTED_SCOPES@</td>
      <td>@DENIED_SCOPES@</td>
      <td>@OAUTH_CUSTOM_MACRO@</td>
      <td><a href="@OAUTH_CLIENTMANAGERURL@?action=remove&id=
          @UNIQUE_ID@">Remove</a></td>
    </tr>
<!-- START NON-TRANSLATABLE -->
   [ERPT trustedClients]
<!-- END NON-TRANSLATABLE -->
  </table>
 </body>
</html>
```

Figure 45. Template for clients_manager.html

OAuth 1.0 template page for consent to authorize

The OAuth server uses this page to determine and store user consent information about which OAuth clients are authorized to access the protected resource. This page also indicates which scope is requested by the OAuth client.

The Tivoli Federated Identity Manager provides an HTML page template called `user_consent.html`.

Tivoli Federated Identity Manager stores the decisions made by the resource owner about which OAuth clients to trust. The resource owner is not prompted every time the same client requests authorization to access the protected resource.

The authorization request from the OAuth client shows a list of approved scopes, and a list of scopes to be approved. These lists are shown in the consent page and can be of indeterminate length. The template supports multiple copies of stanzas that are repeated once for each scope in either list.

This template file provides several replacement macros:

@OAUTH_AUTHORIZE_URI@
: This macro is replaced with the URI for the resource owner authorization endpoint.

@OAUTH_CLIENT_CALLBACK@
: This macro is replaced with the callback URI that the OAuth server uses to send the verification code to. The value depends on the following items:
  - Callback URI that is entered during partner registration.
  - `oauth_callback` parameter in the request for a temporary credential.
  - override registered client callback URI setting.

@OAUTH_CLIENT_COMPANY_NAME@
: This macro is replaced with the name of the company that is requesting access to the protected resource.

@OAUTH_CUSTOM_MACRO@
: This macro is replaced with trusted client information that contains additional information about an authorized OAuth client.

@USERNAME@
: This macro is replaced with the Tivoli Federated Identity Manager user name.

@OAUTH_OTHER_PARAM_REPEAT@
: A multi-valued macro that belongs inside a `[RPT oauthOtherParamsRepeatable]` repeatable replacement list. The values show the list of extra parameter names.

@OAUTH_OTHER_PARAM_VALUE_REPEAT@
: A multi-valued macro that belongs inside a `[RPT oauthOtherParamsRepeatable]` repeatable replacement list. The values show the list of extra parameter values.

@OAUTH_TOKEN_SCOPE_REPEAT@
: A multi-valued macro that belongs either inside `[RPT oauthTokenScopePreapprovedRepeatable]` or `[RPT oauthTokenScopeNewApprovalRepeatable]` repeatable replacement lists. The values inside the `[RPT oauthTokenScopePreapprovedRepeatable]` show the list of token scopes that have been previously approved by the resource owner. Alternatively, the values inside the `[RPT oauthTokenScopeNewApprovalRepeatable]` show the list of token scopes that have *not* yet been approved by the resource owner.

@CONSENT_FORM_VERIFIER@
: This macro is replaced with a unique identifier for the `consent_form_verifier` parameter value. The `consent_form_verifier`

parameter value is automatically generated by the OAuth server. The parameter name and value must not be modified.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>OAuth - Consent to Authorize</title>
  </head>
  <body>
      <h1>OAuth - Consent to Authorize</h1>
      <br />

        <p>The following site is requesting access to an OAuth protected resource:</p>
        <p>@OAUTH_CLIENT_CALLBACK@</p>
        <p>Company Name: @OAUTH_CLIENT_COMPANY_NAME@</p>
        <p>Additional Information: @OAUTH_CUSTOM_MACRO@</p>
        <br />

        <p>User Name: @USERNAME@</p>
        <br />

        <form action="@OAUTH_AUTHORIZE_URI@" method="post">
            <p>The client provided the following extra request parameters:</p>
            <!-- START NON-TRANSLATABLE -->
            [RPT oauthOtherParamsRepeatable]
            @OAUTH_OTHER_PARAM_REPEAT@=@OAUTH_OTHER_PARAM_VALUE_REPEAT@
            <input type="hidden" name="@OAUTH_OTHER_PARAM_REPEAT@"
                  value="@OAUTH_OTHER_PARAM_VALUE_REPEAT@" />
            [ERPT oauthOtherParamsRepeatable]
            <!-- END NON-TRANSLATABLE -->
            <br />

            <p>The client requested the following token scopes
                                that have been previously approved:</p>
            <!-- START NON-TRANSLATABLE -->
            <ul>
            [RPT oauthTokenScopePreapprovedRepeatable]
            <li>@OAUTH_TOKEN_SCOPE_REPEAT@</li>
            <input type="hidden" name="scope" value="@OAUTH_TOKEN_SCOPE_REPEAT@" />
            [ERPT oauthTokenScopePreapprovedRepeatable]
            </ul>
            <!-- END NON-TRANSLATABLE -->
            <br />

            <p>The client requested the following token scopes
                                that have not yet been approved:</p>
            <!-- START NON-TRANSLATABLE -->
            [RPT oauthTokenScopeNewApprovalRepeatable]
            <input type="checkbox" name="scope" value="@OAUTH_TOKEN_SCOPE_REPEAT@"
                  checked="checked"/> <label>@OAUTH_TOKEN_SCOPE_REPEAT@</label><br />
            [ERPT oauthTokenScopeNewApprovalRepeatable]
            <!-- END NON-TRANSLATABLE -->
            <br />

         <p>Would you like to approve this access?</p>
            <br />
   <input type="hidden" name="consent_form_verifier" value="@CONSENT_FORM_VERIFIER@" />

            <!--
                The scope parameters can be:
                1. Requested as part of the redirect for authorization by the client
                   by appending them to the authorize URL as query string parameters, or
                2. If not requested by the client, and you know what authorization options
                   are valid for the OAuth-protected resources being requested, you may
                   also manually prompt for them in this page template as demonstrated
                   by the following example scope's
            -->
            <!--
            <table>
                <tr>
                    <td>Scopes to be authorized: </td>
                    <td>Scope 1</td><td><input type="checkbox" name="scope"
                         value="token_scope_1" /></td>
                    <td>:: Scope 2</td><td><input type="checkbox" name="scope"
                         value="token_scope_2" /></td>
                    <td>:: Scope 3</td><td><input type="checkbox" name="scope"
                         value="token_scope_3" /></td>
                </tr>
            </table>
            -->

          <table>
            <tr><td>Permit </td><td><input type="radio" name="trust_level"
                      value="permit" checked /></td></tr>
            <tr><td>Deny </td><td><input type="radio" name="trust_level"
                      value="deny" /></td></tr>
          </table>
          <br />

          <input type="submit" name="submit" value="Submit" style="width:80px"/>
        </form>
  </body>
</html>
```

Figure 46. Template for user_consent.html

OAuth 1.0 template page for response

Use this HTML page when the callback URI is set to **oob** in the request for temporary credentials or in the partner registration.

When the OAuth client does not specify a callback URI or cannot receive callbacks, the OAuth server does not know where to redirect the resource owner after the authorization process. As a result, the OAuth client does not receive the verification code that it must exchange for a set of token credentials.

Tivoli Federated Identity Manager provides an HTML template page called `user_response.html`. This page shows the OAuth token and verification code that the resource owner can provide to a trusted OAuth client.

The template has the following replacement macros:

@OAUTH_TOKEN@
: This macro is replaced with the `oauth_token` parameter specified in the request for temporary credentials.

@OAUTH_VERIFIER@
: This macro is replaced with the `oauth_verifier` parameter specified in authorization response.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>OAuth - Response</title>
 </head>
 <body>
   <h1>OAuth - Response</h1>   <br />
    <p>Your OAuth client did not provide a callback URL.
      Supply these values to your client:</p>
    <br />
    <p>OAuth Token: <span class="client">@OAUTH_TOKEN@</span></p>
    <br />
    <p>OAuth Verification Code: <span class="client">
                @OAUTH_VERIFIER@</span></p>
   </div>
  </div>
 </body>
</html>
```

Figure 47. Template for user_response.html

OAuth 1.0 template page for denied consent

Use the denied consent HTML page when the resource owner has not granted the OAuth client access to the protected resource.

Tivoli Federated Identity Manager provides the file `user_consent_denied.html`.

The template does not have any replaceable macros.

```
!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>OAuth - Consent Denied</title>
 </head>
 <body>
   <h1>OAuth - Consent Denied</h1>
   <br />
   <div id="content">
    <p>You have denied consent to access your protected resources.<p>
   </div>
  </div>
 </body>
</html>
```

Figure 48. Template for user_consent_denied.html

OAuth 1.0 template page for errors

Tivoli Federated Identity Manager uses a generic error template page to show detailed text information when an error occurs in an OAuth 1.0 flow.

The template page is `user_error.html`.

The following replacement macro is supported:

@OAUTH_ERROR@
: This macro is replaced with the native language support (NLS) text of the error message associated with the error.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>OAuth - Error</title>
 </head>
 <body>
   <h1>OAuth - Error</h1>
   <br />
   <p>The following error occured while processing your OAuth request: </p>
   <p><span class="client">@OAUTH_ERROR@</span></p>
 </body>
</html>
```

Figure 49. Template for user_error.html

OAuth 2.0 template page for consent to authorize

The authorization server uses this page to determine and store user consent information about which OAuth clients are authorized to access the protected resource. This page also indicates scopes that the OAuth client requests.

The Tivoli Federated Identity Manager provides an HTML page template called `user_consent.html`. The macros in the template are specifically for an OAuth 2.0 flow.

Tivoli Federated Identity Manager stores the decisions made by the resource owner about which OAuth clients to trust. The resource owner is not prompted every time the same OAuth client requests authorization to access the protected resource.

The authorization request from the OAuth client shows a list of approved scopes, and a list of scopes to be approved. These lists are shown in the consent page and can be of indeterminate length. The template supports multiple copies of stanzas that are repeated once for each scope in either list.

This template file provides several replacement macros:

@OAUTH_AUTHORIZE_URI@
This macro is replaced with the URI for the authorization endpoint.

@OAUTH_CLIENT_COMPANY_NAME@
This macro is replaced with the name of the company that is requesting access the protected resource.

@CLIENT_ID@
This macro is replaced with the `client_id` parameter specified in the authorization request.

@REDIRECT_URI@
This macro is replaced with the redirect URI that the authorization server uses to send the authorization code to. The value depends on the following items:

- Redirect URI that is entered during partner registration
- `oauth_redirect` parameter specified in the authorization request

@STATE@
This macro is replaced with the `state` parameter specified in the authorization request.

@RESPONSE_TYPE@
This macro is replaced with the `response_type` parameter specified in the authorization request.

@OAUTH_CUSTOM_MACRO@
This macro is replaced with trusted client information that contains additional information about an authorized OAuth client.

@USERNAME@
This macro is replaced with the Tivoli Federated Identity Manager user name.

@OAUTH_OTHER_PARAM_REPEAT@
A multi-valued macro that belongs inside a `[RPT oauthOtherParamsRepeatable]` repeatable replacement list. The values show the list of extra parameter names.

@OAUTH_OTHER_PARAM_VALUE_REPEAT@
A multi-valued macro that belongs inside a `[RPT oauthOtherParamsRepeatable]` repeatable replacement list. The values show the list of extra parameter values.

@OAUTH_TOKEN_SCOPE_REPEAT@
A multi-valued macro that belongs either inside `[RPT oauthTokenScopePreapprovedRepeatable]` or `[RPT oauthTokenScopeNewApprovalRepeatable]` repeatable replacement lists. The values inside the `[RPT oauthTokenScopePreapprovedRepeatable]` show the list of token scopes that have been previously approved by the resource

owner. Alternatively, the values inside the [RPT oauthTokenScopeNewApprovalRepeatable] show the list of token scopes that have *not* yet been approved by the resource owner.

@CONSENT_FORM_VERIFIER@

This macro is replaced with a unique identifier for the consent_form_verifier parameter value. The consent_form_verifier parameter value is automatically generated by the authorization server. The parameter name and value must not be modified.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>OAuth 2.0 - Consent to Authorize</title>
  </head>
  <body>
    <form action="@OAUTH_AUTHORIZE_URI@" method="GET">
      <h1>OAuth 2.0 - Consent to Authorize</h1>

      <p>The following site is requesting access to an OAuth 2.0 protected resource:</p>
      <p><b>@OAUTH_CLIENT_COMPANY_NAME@</b></p>

      <p>The client provided the following OAuth 2.0 request parameters:</p>
      <ul>
        <li>Client Id: @CLIENT_ID@</li>
        <li>Redirect URI: @REDIRECT_URI@</li>
        <li>State: @STATE@</li>
        <li>Response Type: @RESPONSE_TYPE@</li>
      </ul>

      <p>Additional Information: @OAUTH_CUSTOM_MACRO@</p>

      <p>By approving this request you will be providing
                                delegated authorization on behalf of:</p>
      <p><b>@USERNAME@</b></p>

      <p>The client provided the following extra request parameters:</p>
      <!-- START NON-TRANSLATABLE -->
      <ul>
      [RPT oauthOtherParamsRepeatable]
      <li>@OAUTH_OTHER_PARAM_REPEAT@=@OAUTH_OTHER_PARAM_VALUE_REPEAT@</li>
      <input type="hidden" name="@OAUTH_OTHER_PARAM_REPEAT@"
                                    value="@OAUTH_OTHER_PARAM_VALUE_REPEAT@" />
      [ERPT oauthOtherParamsRepeatable]
      </ul>
      <!-- END NON-TRANSLATABLE -->

      <p>The client requested the following token scopes
                                     that have been previously approved:</p>
      <!-- START NON-TRANSLATABLE -->
      <ul>
      [RPT oauthTokenScopePreapprovedRepeatable]
      <li>@OAUTH_TOKEN_SCOPE_REPEAT@</li>
      <input type="hidden" name="scope" value="@OAUTH_TOKEN_SCOPE_REPEAT@" />
      [ERPT oauthTokenScopePreapprovedRepeatable]
      </ul>
      <!-- END NON-TRANSLATABLE -->

      <p>The client requested the following token scopes
                                       that have not yet been approved:</p>
      <!-- START NON-TRANSLATABLE -->
      [RPT oauthTokenScopeNewApprovalRepeatable]
      <input type="checkbox" name="scope" value="@OAUTH_TOKEN_SCOPE_REPEAT@"
                checked="checked"/><label>@OAUTH_TOKEN_SCOPE_REPEAT@</label><br />
      [ERPT oauthTokenScopeNewApprovalRepeatable]
      <!-- END NON-TRANSLATABLE -->
      <br />

      <p>Would you like to approve access to this scope?</p>
      <input type="hidden" name="consent_form_verifier" value="@CONSENT_FORM_VERIFIER@" />

            <!--
                The scope parameters can be:
                1. Requested as part of the redirect for authorization by the client
                   by appending them to the authorize URL as query string parameters,
                   and/or
                2. If not requested by the client, and you know what authorization options
                   are valid for the protected resources being requested, you may
                   also manually prompt for them in this page template as demonstrated
                   by the following example scope's
            -->
            <!--
            <table>
                <tr>
                    <td>Scopes to be authorized: </td>
                    <td>Scope 1</td><td><input type="checkbox" name="scope"
                               value="token_scope_1" /></td>
                    <td>:: Scope 2</td><td><input type="checkbox" name="scope"
                               value="token_scope_2" /></td>
                    <td>:: Scope 3</td><td><input type="checkbox" name="scope"
                               value="token_scope_3" /></td>
                </tr>
            </table>
            -->

      <table>
        <tr>
          <td>Permit </td>
          <td><input type="radio" name="trust_level" value="permit" checked /></td>
        </tr>
        <tr>
          <td>Deny </td>
          <td><input type="radio" name="trust_level" value="deny" /></td>
        </tr>
      </table>
      <br />

      <input type="submit" name="submit" value="Submit" style="width: 80px" />
    </form>
  </body>
</html>
```

Figure 50. Template for user_consent.html

OAuth 2.0 template page for response

Use this HTML page to show the authorization code of an OAuth client that did not specify a client redirection URI upon partner registration.

When the OAuth client does not specify a client redirection URI or cannot receive redirects, the authorization server does not know where to send the resource owner after authorization. The OAuth client does not receive the authorization code required to exchange for an access token or refresh token.

The Tivoli Federated Identity Manager provides an HTML template page called `user_response.html`. This page shows the authorization code that the resource owner can provide to a trusted OAuth client.

The following replacement macro is supported:

@OAUTH_CODE@
: This macro is replaced with the `oauth_code` parameter specified in authorization response.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>OAuth - Response</title>
  </head>
  <body>
      <h1>OAuth - Response</h1>
      <br />
        <p>Your OAuth client did not provide a redirect URI.
                Supply this value to your client:</p>
        <br />
        <p>OAuth Authorization Code: <span class="client">@OAUTH_CODE@</span></p>
  </body>
</html>
```

Figure 51. Template for user_response.html

OAuth 2.0 template page for errors

Tivoli Federated Identity Manager uses a generic error template page to show detailed text information when an error occurs in an OAuth 2.0 flow.

The template page is `user_error.html`.

The following replacement macro is supported:

@ERROR_CODE@
: This macro is replaced with characters that uniquely identify the error.

@ERROR_DESCRIPTION@
: This macro is replaced with the native language support (NLS) text of the error message associated with the error.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>OAuth 2.0 - Error</title>
  </head>
  <body>
    <h1>OAuth 2.0 - Error</h1>

    <p>The following error was encountered while processing your OAuth request:</p>

    <p>Error Code: <b>@ERROR_CODE@</b></p>
    <p>Error Description: <b>@ERROR_DESCRIPTION@</b></p>
  </body>
</html>
```

Figure 52. HTML template for user_error

Chapter 30. Planning a Liberty federation

You must specify the values for federation properties when configuring a Liberty federation. Keep in mind however, that support for Liberty protocol will be deprecated in the later versions of IBM Tivoli Federated Identity Manager.

Familiarize yourself with the Liberty standards documentation before implementing a single sign-on federation. The standards specify data exchange and message processing. Know what information you must provide to your partners, and what information your partner must provide to you.

Liberty Alliance

`http://www.projectliberty.org`

The Federation wizard prompts you to supply values for a number of properties. Most of them can be modified later, after federation creation.

The choice of profile (or profiles) to use is based on both business policy decisions and network security architecture. Federation partners must agree on the profile choices in order to activate user single sign-on across the federation. The choice must be made before configuring the federation.

The Liberty standard supports a unique range of single sign-on profiles. The profiles extend beyond specifications for achieving federated single sign-on, and can include other functions such as single logout, federation termination notification, and register name identification.

Identity provider and service provider roles

Each partner in a federation has a role. The role is either **Identity Provider** or **Service Provider**. This section provides descriptions for the two roles.

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

- Identity provider

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

  The identity provider directly authenticates the user by doing any of the following tasks:

  – validating a user name and password,
  – indirectly authenticating the user,
  – validating an assertion about the user identity, as presented by a separate identity provider.

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

- Service Provider

  A service provider is a federation partner that provides services to user. Typically, service providers do not authenticate users but instead request authentication decisions from an identity provider. Service providers rely on

identity providers to assert the identity of a user, and rely on identity providers to manage user identities for the federation.

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

Liberty single sign-on profiles

Liberty supports more than one single sign-on profile. You must select at least one profile. Keep in mind however, that support for Liberty protocol will be deprecated in the later versions of IBM Tivoli Federated Identity Manager.

You can optionally configure both Browser artifact and Browser POST profiles when configuring an identity provider. You can configure only one profile when configuring a service provider.

Browser artifact

Browser artifact uses a SOAP backchannel to exchange an artifact during the establishment and use of a trusted session between an identity provider, a service provider, and a client (browser).

You can optionally configure browser artifact when configuring an identity provider or a service provider.

When you select browser artifact, enter the name of an encryption key for the trusted session. Specify a key even if you choose to not require the signing of assertions for other Liberty message communications.

Browser POST

Browser POST uses a self-posting form during the establishment and use of a trusted session between an identity provider, a service provider, and a client (browser).

You can optionally configure browser POST when configuring an identity provider or a service provider.

Note: When configuring an identity provider, you can select both browser artifact and browser POST profiles. However, when configuring a service provider, you can select only one profile – either browser artifact or browser POST.

Liberty-enabled client/proxy (LECP) single sign-on profile

A Liberty-enabled client or Liberty-enabled proxy has, or knows how to obtain the information required to connect to the identity provider that the user (principal) wants to use with the service provider. A Liberty-enabled proxy is an HTTP proxy such as a Wireless Application Protocol (WAP) gateway that emulates a Liberty-enabled client.

LECP Providers

A comma delimited list of header variables used by LECP. This property is set when configuring both identity providers and service providers. There is no default value.

An example of single header variable:

```
ibm_msisdn
```

An example of multiple header variables:

```
ibm_msisdn,x_msisdn
```

Liberty register name identifier

This profile updates the identifier for a specific user or principal. Keep in mind however, that support for Liberty protocol will be deprecated in the later versions of IBM Tivoli Federated Identity Manager.

Liberty requires identity providers and service providers to exchange an alias (also called an identifier) to each user account. The alias exchange is done instead of exchanging the real account name of the user. The alias exchange enables account linkage while hiding the user account name.

Configuration of the Register Name Identifier profile is optional.

When the profile is selected, the administrator must select the communication bindings to use between providers. The bindings can be specified separately for the identity provider and the service provider. The supported bindings are:

- HTTP redirect

  The identity provider and service provider communicate by sending HTTP 302 redirects to the browser. Updates to name identifiers are accomplished serially through the redirects. HTTP redirect is the default binding for both identity and service providers.
- SOAP/HTTP

  Updates to name identifiers are accomplished by direct exchanges between providers over a SOAP connection.

The endpoints are:

Register Name Identifier Service URL
The URL endpoint is used for user-agent-based Register Name Identifier protocols. A default value is provided. For example:

```
https://idp.example.com/FIM/sps/libertyfed/liberty/rni
```

Register Name Identifier Return URL
The URL endpoint used for redirection after HTTP name registration has taken place. A default value is provided. For example:

```
https://idp.example.com/FIM/sps/libertyfed/liberty/rnireturn
```

This value is required for RNI with HTTP Redirect communication. This value is not required for SOAP/HTTP communication.

Liberty federation termination notification

This profile terminates account linkage across the federation for a specified user. This profile is disabled by default.

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

Configuration of this profile is optional. When the profile is selected, you must select the communication bindings to use between providers. The bindings can be specified separately for the identity provider and the service provider. The supported bindings are:

- HTTP redirect

The identity provider and service provider communicate by sending HTTP 302 redirects to the browser. Termination of account federation is accomplished serially through the redirects. HTTP redirect is the default binding for both identity and service providers.

- SOAP/HTTP

  Termination of account federation is accomplished by direct exchanges between providers over a SOAP connection.

The endpoints are:

Federation Termination Notification Service URL
The URL on the provider to which single federation termination notification processes are sent. A default value is provided. For example:

```
https://idp.example.com/FIM/sps/libertyfed/liberty/ftn
```

Federation Termination Notification Return URL
The URL used by the identity or service Provider when redirecting the user agent at the end of the user-agent-based federation termination notification process.

```
https://idp.example.com/FIM/sps/libertyfed/liberty/ftnreturn
```

This value is required for FTN when using HTTP Redirect communication.

Liberty single logout

This profile terminates all login sessions within the federation for a specified user. This profile is disabled by default.

Note: Support for Liberty protocol will be deprecated in the later versions of IBM Tivoli Federated Identity Manager.

Configuration of this profile is optional. When the profile is selected, the administrator must select the communication bindings to use between providers. The bindings can be specified separately for the identity provider and the service provider. The supported bindings are:

- HTTP redirect

  The identity provider and service provider communicate by sending HTTP 302 redirects to the browser. Logout of user sessions is accomplished serially through the redirects. HTTP redirect is the default binding for both identity and service providers

- HTTP GET

  Identity providers can use Image Tags to cause the browser to use HTTP GET to communicate the logout requests to the service providers. Logout requests are processed concurrently rather than serially. If a logout request fails, any remaining logout requests are unaffected, and are sent to the appropriate service provider. By contrast, when logout requests are processed serially (HTTP redirect) a failed logout request cancels any remaining logout requests.

  Note: This option is specified only on identity providers. Service providers cannot set this option.

- SOAP/HTTP

  Logout of user sessions is accomplished by direct exchanges between providers over a SOAP connection.

The endpoints are:

Single Logout Service URL
The URL to which the service provider sends a request to log out a user. A default value is provided. For example:

```
https://idp.example.com/FIM/sps/libertyfed/liberty/slo
```

Single Logout Return URL
The URL used by the service provider when redirecting the user agent to the identity provider at the end of the single logout profile process. A default value is provided. For example:

```
https://idp.example.com/FIM/sps/libertyfed/liberty/sloreturn
```

This value is required for SLO using HTTP Redirect communication.

Liberty identity provider introduction

Identity Provider Introduction enables a service provider to discover which identity providers are used by a user (Principal).

Support for Liberty protocol will be deprecated in the later versions of IBM Tivoli Federated Identity Manager.

The Introduction profile relies on a cookie that is written in a domain that is common between identity providers and service providers in an identity federation network.

This profile is configured only on an identity provider.

Common DNS Domain

The common DNS domain is a virtual domain into which a component is configured to set or retrieve a cookie. Use of this common domain enables identity providers and service providers, which typically exist in separate domains, to access a cookie. The domain does not have to exist before setting this configuration property. However, you must create it before a user attempts single sign-on while relying on the identity provider introduction profile. This property is set only when configuring an identity provider. There is no default value. For example:

```
cot.projectliberty.org
```

IPI configuration requires that you enter a value for this field.

Common Domain Hostname

The name of a host system in the common DNS domain. This host receives requests to either set or read the common domain cookie used by the identity provider introduction profile. This property is set only when configuring an identity provider. There is no default value. For example:

```
idp.cot.projectliberty.org
```

The domain name portion of this host name must match the value specified in the Common DNS Domain. In this example, the host system value must include `cot.projectliberty.org`.

IPI configuration requires that you enter a value for this field.

Liberty message security

The federation creation wizard prompts you whether you want to sign Liberty messages. When you chose to sign Liberty messages, you must specify a key or certificate to use.

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

Digital signature options

In some cases, you must still enter a key or certificate if you do not select **Sign Liberty Messages**. For example:

- You must specify a key to sign messages sent across the backchannel for the artifact when you select the browser artifact profile.
- You must specify a key or certificate when you select one of the optional profiles, and specify the SOAP communication the service provider initiates.

Provide the following configuration information if you must enter a key or certificate:

Keystore file name
The wizard presents a choice of the keystores that you configured before you began configuration of the single sign-on federation.

Keystore password
You must supply the password for the keystore you specify.

Key name
You must specify which key to use.

Liberty communication properties

You must know how to fill out the Liberty communication properties for your identity provider and service provider. Keep in mind however, that support for Liberty protocol will be deprecated in the later versions of IBM Tivoli Federated Identity Manager.

Liberty Message Lifetime

An integer value indicating the amount of time, in seconds, that a Liberty message remains valid. This property is set on both the identity provider and the service provider.

Minimum value: 60 seconds

Maximum: No maximum other than the maximum integer supported by the data type.

Default: 60 seconds.

Liberty Artifact Lifetime

An integer indicating the time, in seconds, in which a service provider must retrieve an assertion from an identity provider. The service provider uses an artifact to retrieve the assertion. The identity provider keeps the mapping of the artifact to the assertion in its cache for this amount of time. When the service provider does not collect the artifact in this amount of time, the cache purges the artifact and the service provider login fails.

This property is specified only when configuring an identity provider with browser artifact single sign-on profile.

Note: This value is not used for browser POST profile.

Minimum value: 120 seconds

Default: 120 seconds.

Require Consent to Federate

Enables or disables the requirement that the identity provider prompt the user to consent to joining the federation. This property is set on the identity provider only. This message is presented when federating of the user account occurs. Default value is disabled. Select the check box to activate the issuing of the prompt.

SOAP Endpoint

The Simple Object Access Protocol (SOAP) endpoint location at the service provider or identity provider to which Liberty SOAP messages are sent.

This setting is required when one or both of the following conditions is or are true:

- Browser artifact single sign-on profile is selected on the Liberty profiles window.
- One or more of the optional Liberty profiles are selected and SOAP/HTTP communication initiated by at least one of the service providers is selected.

For example:

`https://idp.example.com/FIM/sps/libertyfed/liberty/soap`

Single Sign-on is Passive (Identity Provider does not interact with user)

Enables or disables the requirement that the identity provider must not interact with principal (user) and must not take control of the user interface from the service provider. This property is set only when configuring a service provider. Select the check box to enable this requirement. Default value: Disabled

Force Identity Provider to authenticate user

Enables or disables a requirement that the identity provider must authenticate a user (Principal) regardless of whether the user is already authenticated. This value is specified only when the **Single Sign-on is Passive (Identity Provider does not interact with user)** check box is cleared. This property is set only when configuring a service provider.

When this setting is cleared, the identity provider must authenticate the user (Principal) only when the user is not presently authenticated.

- Select the **Force Identity Provider to authenticate user** check box to enable this requirement.
- Clear the check box to disable this requirement.

Liberty token modules

When you create a single sign-on federation, you must configure an instance of a security token module for the federation. Keep in mind however, that support for Liberty protocol will be deprecated in the later versions of IBM Tivoli Federated Identity Manager.

The token module corresponds to a security token type that defines the format for the encrypted token that contains user credential information.

The identity provider and service provider exchange tokens as part of the authentication and authorization services to process user access request.

When you use the federation creation wizard, a token type is automatically selected for you based on your choice of single sign-on protocol.

Configuration of the Liberty token module is required only on the identity provider. No configuration is required when deploying a service provider.

The configuration property is the same for both Liberty v1.1 tokens and Liberty v1.2 tokens.

Amount of time the assertion is valid after being issued (seconds)
An integer value that specifies the number of seconds that the assertion remains valid. This integer value is specified for Liberty tokens. The minimum value is 120 seconds. The maximum value is 300 seconds.

Liberty identity mapping

The federation creation wizard prompts you to specify either an XSLT mapping rule file or a custom mapping module instance. Keep in mind however, that support for Liberty protocol will be deprecated in the later versions of IBM Tivoli Federated Identity Manager.


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 XSLT mapping file or custom mapping module instance must be prepared before you configure the federation.

XSLT Transformation for Identity Mapping
Select this button in the wizard if you can provide an XSL file containing the identity mapping. Enter the name of a file on the local file system.

Custom Mapping Module Instance
Select this button in the wizard if you can provide a custom mapping module instance to use instead of an XSL file. The system prompts you to enter any configuration properties that your custom mapping module instance requires.

Mapping a Tivoli Access Manager credential to a Liberty or SAML 2 token

This scenario occurs when messages are exchanged between partners in a Liberty or SAML 2 single sign-on federation, and user identity information is managed by Tivoli Access Manager

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


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.

When a user request is received (for example, for access to a remote resource) the trust service contacts Tivoli Access Manager and obtains a Tivoli Access Manager credential for the user identity.

In this scenario, the trust service Tivoli Access Manager credential module operates in validate mode. In this mode, it converts the Tivoli Access Manager credential to an Input STS universal user document (In-STSUUSER). The In-STSUUSER that is created from the Tivoli Access Manager credential module has all of the information from the credential. This information is available for possible use by the trust service module that builds the outgoing token.

The trust service consults its configuration entry for the federation partner (for example, the destination that hosts a requested resource). The configuration indicates the type of token to be created.

Next, the identity mapping module converts the In-STSUUSER into an Output STS universal user (Out-STSUUSER). The Out-STSUUSER must contain the information that is needed by the Tivoli Federated Identity Manager Liberty (or SAML 2) token module to generate a Liberty (or SAML 2) token.

The Out-STSUUSER must contain the following information in order for the token module to be able to generate a valid token:

Table 119. Out-STSUUSER entries used to generate a Liberty or SAML 2 token

Out-STSUUSER element	**Token Information**	**Required?**
Principal Attr: Name	Name to be passed to the alias service	Required
Attribute: AuthenticationMethod	The authentication method. N **Note:** This element is always set to "password" (Username/password) regardless of the authentication mechanism set in the Tivoli Access Manager credential.	Required
Attribute List	Additional custom attributes.	Optional

The mapping module is responsible for:

1. Mapping Principal Attr Name in In-STSUUSER to a Principal name entry in the Out-STSUUSER.

   Note: When the token module generates the token, this Principal name is not directly used. Instead, the value in the Name field is sent as input to the Tivoli Federated Identity Manager alias service. The alias service obtains the alias (name identifier) for the principal, and places the returned alias in the generated token module.

   Figure 53 on page 458 shows a sample mapping rule file from the demonstration application mapping file, ip_liberty.xsl. No

   Note: Liberty tokens are extensions to SAML tokens. Therefore, comments in the sample code that refer to SAML tokens, are correct in this context.

```
</xsl:template>
<!-- This template replaces the entire Principal element with one that contains
     just the email address (from the ivcred tagvalue_email) and the data type
     appropriate for SAML. -->
 <xsl:template match="//stsuuser:Principal">
  <stsuuser:Principal>
   <stsuuser:Attribute name="name"
                type="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
    <stsuuser:Value>
    <xsl:value-of
    select="//stsuuser:AttributeList/stsuuser:Attribute[@name='tagvalue_email']
          [@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
    </stsuuser:Value>
   </stsuuser:Attribute>
  </stsuuser:Principal>
 </xsl:template><!--
```

Figure 53. XSL code sample showing mapping of a value from a Tivoli Access Manager credential into a Principal name for a Liberty token

2. Setting the authentication method to the "password" mechanism, regardless of the value obtained from the Tivoli Access Manager credential. This action is required by the token module.

   Figure 54 shows a sample mapping rule file from the demonstration application mapping file, ip_liberty.xsl.

```
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList><!-- First the authentcation method attribute -->
   <stsuuser:Attribute name="AuthenticationMethod"
                           type="urn:oasis:names:tc:SAML:1.0:assertion">
    <stsuuser:Value>urn:oasis:names:tc:SAML:1.0:am:password</stsuuser:Value>
   </stsuuser:Attribute>
         ....
      </stsuuser:AttributeList>
</xsl:template>
```

Figure 54. XSL code sample showing assignment of authentication method as an Attribute for a Liberty token.

3. Populating the attribute statement of the assertion with the attributes in the AttributeList in the In-STSUUSER. This information becomes custom information in the token.

   There can be custom attributes that are required by applications that use the information transmitted between federation partners.

   Figure 55 on page 459 shows a sample mapping rule file from the demonstration application mapping file, ip_liberty.xsl.

```
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList>
         ....
      <!-- Now the commonName attribute -->
   <stsuuser:Attribute name="commonName"
                        type="http://example.com/federation/v1/commonName">
    <stsuuser:Value>
     <xsl:value-of
     select="//stsuuser:AttributeList/stsuuser:Attribute[@name='tagvalue_name']
           [@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
    </stsuuser:Value>
   </stsuuser:Attribute>
   <!-- Now the ssn attribute -->
   <stsuuser:Attribute name="ssn" type="http://example.com/federation/v1/ssn">
    <stsuuser:Value>
     <xsl:value-of
     select="//stsuuser:AttributeList/stsuuser:Attribute[@name='tagvalue_ssn']
            [@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
    </stsuuser:Value>
   </stsuuser:Attribute>
             ....
     </stsuuser:AttributeList>
 </xsl:template>
```

Figure 55. XSL code sample showing assignment of optional attributes for a Liberty token

4. The GroupList element of the In-STSUUSER is not read by the token module. However, information in this element can optionally be used to populate custom attributes of the Out-STSUUSER.

   Figure 56 shows the optional assignment of a GroupList value to an attribute. This code sample is from the demonstration application mapping file ip_liberty.xsl.

```
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList>
         ....
      <!-- Now the role attribute (can be multi-valued) -->
   <stsuuser:Attribute name="role"
                        type="http://example.com/federation/v1/role">
    <xsl:for-each select="//stsuuser:GroupList/stsuuser:Group">
     <stsuuser:Value>
      <xsl:value-of select="@name" />
     </stsuuser:Value>
    </xsl:for-each>
   </stsuuser:Attribute>
             ....
     </stsuuser:AttributeList>
 </xsl:template>
```

Figure 56. XSL code sample showing optional assignment of GroupList value to an attribute for a Liberty token

Mapping a Liberty or SAML 2 token to a Tivoli Access Manager credential

Map a Liberty or SAML 2.0 token to a Tivoli Access Manager credential for a single sign-on federation scenario.

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 service provider receives a Liberty or SAML 2 token. The token module, operating in validate mode, creates an In-STSUUSER document from the token. Table 120 shows the information from the token that is converted into an In-STSUUSER document.

Table 120. Token information that is converted into a STS universal user document

Token Information	In-STSUUSER element	Required in Out-STSUUSER?
UserID obtained from the alias service	Principal Attr: Name	Required
Additional custom attributes	Attribute List	Optional

Note that the token module does not populate the GroupList element in the In-STSUUSER document.

The token module reads the token and obtains the NameIdentifier. The token module passes the NameIdentifier (an alias) to the alias service. The alias service converts the received alias to the local Tivoli Access Manager User ID. The token module puts the User ID into the Principal element in the In-STSUUSER document.

The trust service must convert this information to a Tivoli Access Manager credential, in order to make an authorization decision on the request from the user identity.

- The NameIdentifier alias that is returned is used to populate the `name` attribute of the Principal. This is the local user ID.

  Figure 57 shows the assignment of a set value for the Principal name. This code sample is from the demonstration application mapping file sp_liberty.xsl.

```
<!-- This will replace the principal name (which was the email address
     in the SAML assertion) with the user "me_guest". -->
   <xsl:template match="//stsuuser:Principal/stsuuser:Attribute[@name='name']">
           <stsuuser:Attribute name="name"
                               type="urn:ibm:names:ITFIM:5.1:accessmanager">
            <stsuuser:Value>
<xsl:value-of
 select="//stsuuser:Principal/stsuuser:Attribute[@name='name']/stsuuser:Value" />
            </stsuuser:Value>
          </stsuuser:Attribute>
       </xsl:template>
```

Figure 57. XSL code sample showing assignment of a value for the Principal name for a Liberty token.

- Other information from the token is used to populate Attributes in the Attribute List.

  Figure 58 on page 461 shows the optional assignment of additional values to attributes. This code sample is from the demonstration application mapping file

sp_liberty.xsl

```
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList>
          ....
              ....
      <!-- The tagvalue_sso attribute -->
      <stsuuser:Attribute name="tagvalue_sso"
                          type="urn:ibm:names:ITFIM:5.1:accessmanager">
                    <stsuuser:Value>isSingleSignOn</stsuuser:Value>
      </stsuuser:Attribute>
      <!-- The tagvalue_fedname attribute -->
      <stsuuser:Attribute name="tagvalue_fedname"
                         type="urn:ibm:names:ITFIM:5.1:accessmanager">
                     <stsuuser:Value>libertyfed</stsuuser:Value>
      </stsuuser:Attribute>
              ....
              ....
     </stsuuser:AttributeList>
 </xsl:template>
```

Figure 58. XSL code sample showing optional assignment of attributes for a Liberty token.

Liberty alias service

The Liberty standards for single sign-on protocols standards require the use of aliases when a user identity is sent in a message between partners in a single sign-on federation. The standards require aliases as a method of increasing the privacy of the end user when accessing resources at a service provider.

The specifications refer to the aliases as *Name Identifiers*. Name identifiers for a user are registered during account federation (account linkage) and are thereafter used in all messages between partners. Aliases are randomly generated and do not contain any meaningful identity information.

For each user, a different Name Identifier is required for use with each partner. Optionally, a different Name Identifier can be created for messages in each direction. This capability means that a different alias is used for a user when the identity provider contacts the service provider rather than when the service provider sends a message to the identity provider.

Tivoli Federated Identity Manager provides an alias service that handles the alias management tasks. This service hides most of the alias generation and exchange tasks from the federation administrator. The alias service provides the following services:

- Generation of new aliases and association of them with local users
- Look up of a local user identity when an alias is received from a partner
- Look up of the alias for a local user when the provider needs to send a message to a partner

The Tivoli Federated Identity Manager alias service stores alias information in a user registry. The alias service supports the following user registries:

- IBM Tivoli Directory Server
- Sun ONE

For each of these LDAP servers, you will set some configuration parameters after you have created the Liberty federation.

The alias service does not support Lotus Domino or Microsoft Active Directory user registries. You can write your own alias service for use with those registries.

Chapter 31. Configuring a Liberty federation

Configure a Liberty federation by creating the federation, adding a partner to your federation, and providing the new federation configuration information to your partner.

About this task

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

Complete these tasks:

Procedure

1. "Creating a Liberty identity provider"
2. "Creating a Liberty service provider" on page 465
3. "Configuring a WebSEAL point of contact server for the Liberty federation" on page 467
4. "Exporting Liberty federation properties" on page 469
5. "Exporting SOAP endpoint authentication information to a Liberty federation partner" on page 469
6. "Obtaining metadata from a Liberty federation partner" on page 470
7. "Importing SOAP endpoint authentication information from a Liberty federation partner" on page 471
8. "Adding a partner to a Liberty federation" on page 473

Creating a Liberty identity provider

Create a Liberty identity provider to authenticate your users directly or indirectly when using your service provider.

About this task

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

To create a Liberty identity provider federation, complete the steps in this procedure:

Procedure

1. Log on to the management console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations**. The Current Domain and Federations portlets open.
3. Click **Create**. The Federation wizard starts. The General Information panel opens.
4. Enter a name for the federation and select a role.
5. Click **Next**.
6. Enter the contact information.
7. click **Next**.

8. Select the Liberty 1.1 or Liberty 1.2 protocol.
9. click **Next**. he Point of Contact Server panel opens.
10. Enter the point of contact address and click **Next**.
11. Specify the profiles to use with this federation.
    a. Select at least one of the Liberty single sign-on profiles.

       Liberty supports three single sign-on profiles. You must select at least one profile. You can optionally select both Browser artifact and Browser POST profiles when configuring an identity provider. You can select only one profile when configuring a service provider.
    b. Select any of the optional profiles that you want to configure:
       - **Register Name Identifier**
       - **Federation Termination Notification**
       - **Single Logout**
       - **Identity Provider Introduction**.

         For identity providers only.
12. When finished, click **Next**. The Digital Signature Options panel opens.
13. Select or clear the check box for **Sign Liberty Messages**. When you chose to sign Liberty messages, you must specify a key or certificate to use.

    In some cases, when you do not select **Sign Liberty Messages**, you must still enter a key or certificate. For example:
    - You must specify a key to sign messages that are sent across the back channel for the artifact when you select a browser artifact profile,
    - You must specify a key or certificate when you select one of the optional profiles, and specify the SOAP communication to be initiated by the service provider.
14. Select a keystore and enter the keystore password if you must enter a key or certificate. Click **List Keys** to show the keys or certificates in the selected keystore, and select a key.
    - The password for the default **DefaultKeyStore** keystore is `testonly`.
    - A sample key is provided for test purposes only. Do not use this key in a production environment.
15. Click **Next**.
16. Configure the Liberty data properties:
    a. A default value is provided for the **SOAP endpoint**. Use this value unless there is an endpoint conflict on your host.
    b. Specify **Liberty Message Lifetime**.
    c. Specify **Liberty Artifact Lifetime**.
    d. Select or clear the check box for **Require Consent to Federate**.
    e. When LECP profile has been selected, enter **LECP providers**.
    f. When Identity Provider Introduction has been selected, enter **Common DNS Domain** and **Common Domain Hostname**.
    g. Click **Next**.

    The Liberty Token Module Configuration panel opens. The panel contents are the same for both Liberty v1.1 tokens and Liberty v1.2 tokens.
17. Specify a value in the **Amount of time the assertion is valid after being issued (seconds)** field.
18. Click **Next**.
19. The Identity Mapping Options panel opens. Select one of the radio buttons.

- Use XSL Transformation for Identity Mapping

  Indicates that you must provide an XSL file containing the required identity mapping.

  a. When you select this choice and click **Next**, the Identity Mapping panel opens. Enter the name of a file on the local file system that contains the identity mapping rule in the **XSLT File Containing Identity Mapping Rule** field.

     A file that you have prepared before the installation.

     Optionally you can click the **Browse** button to locate the file on the local file system.

  b. Click **Next**.

     An error is shown if the file cannot be found or if the file does not contain valid XSLT (eXtensible Stylesheet Language Transform).

- Use Custom Mapping Module Instance

  Indicates that you must provide a custom mapping module instance to use instead of an XSL file.

  a. When you select **Use Custom Mapping Module Instance**, a table of Module Instances shows. Select the radio button for the module instance to use and click **Next**.

  b. When your custom mapping module instance requires you to specify values for properties, you will be prompted for them now. Otherwise, the panel shows a message indicating that there are no properties to configure for the specified module instance.

The Summary panel opens.

20. Verify that the configuration settings are correct.
21. Click **Finish**. The Create Federation Complete portlet opens.

What to do next

If you are using WebSEAL as your Point of Contact server, configure it now. Do not exit the management console. See "Configuring a WebSEAL point of contact server for the Liberty federation" on page 467 for details.

Creating a Liberty service provider

Create a Liberty service provider to request authentication decisions from your identity provider.

About this task

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

Procedure

1. Log on to the management console.
2. click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations**. The Current Domain and Federations portlets open.
3. Click **Create**. The Federation wizard starts. The General Information panel opens.
4. Enter a name for the federation and select a role.
5. Click **Next**.

6. Enter the contact information.
7. Click **Next**.
8. Select the Liberty 1.1 or Liberty 1.2 protocol.
9. Click **Next**. The Point of Contact Server panel opens.
10. Enter the point of contact address and click **Next**.
11. Specify the profiles to use with this federation.
    a. Select at least one of the Liberty single sign-on profiles.

       Liberty supports three single sign-on profiles. You must select at least one profile. You can optionally select both Browser artifact and Browser POST profiles when configuring an identity provider. You can select only one profile when configuring a service provider.
    b. Select any of the optional profiles that you want to configure:
       - **Register Name Identifier**
       - **Federation Termination Notification**
       - **Single Logout**
       - **Identity Provider Introduction**.

         For identity providers only.
12. Click **Next** when finished. The Digital Signature Options panel opens.
13. Select or clear the check box for **Sign Liberty Messages**. When you chose to sign Liberty messages, you must specify a key or certificate to use.

    In some cases, when you do not select Sign Liberty Messages, you must still enter a key or certificate. For example:
    - When you select browser artifact profile, you must specify a key to be used to sign messages that are sent across the backchannel for the artifact.
    - When you select one of the optional profiles, and specify the SOAP communication to be initiated by the service provider, you must specify a key or certificate.
14. If you must enter a key or certificate, select a keystore and enter the keystore password. Click **List Keys** to show the keys or certificates contained in the selected keystore, ans select a key.
    - The password for the default **DefaultKeyStore** keystore is `testonly`.
    - A sample key is provided for test purposes only. Do not use this key in a production environment.
15. Configure the settings for Liberty profile service provider:
    a. If you selected an optional Liberty profile (register name identifier, federation termination notification or single logout), and you chose SOAP/HTTP as the communication protocol, you must specify a SOAP endpoint. A default value is provided. You can accept the default unless you have a specific configuration requirement that calls for a different SOAP endpoint.
    b. Specify a value in the **Liberty Message Lifetime (in seconds)** field.
    c. Select or clear the check mark for **Single Sign-on is Passive (Identity Provider does not interact with user)**.
    d. Select or clear the check mark for **Force Identity Provider to authenticate user**.
    e. If you selected LECP single sign-on profile, enter a **LECP Provider**. Click **Next**.
16. Click **Next**.
17. The Identity Mapping Options panel opens. Select one of the radio buttons.

- Use XSL Transformation for Identity Mapping

  Indicates that you will provide an XSL file containing the required identity mapping.

  a. When you select this choice and click **Next**, the Identity Mapping panel opens. Enter the name of a file on the local file system that contains the identity mapping rule in the **XSLT File Containing Identity Mapping Rule** field.

     A file that you have prepared before this installation.

     Optionally you can click the **Browse** button to locate the file on the local file system.

  b. Click **Next**.

     An error is shown if the file cannot be found or if the file does not contain valid XSLT (eXtensible Stylesheet Language Transform).

- Use Custom Mapping Module Instance

  Indicates that you must provide a custom mapping module instance to use instead of an XSL file.

  a. When you select **Use Custom Mapping Module Instance**, a table of Module Instances is shown. Select the radio button for the module instance to use and click **Next**.

  b. When your custom mapping module instance requires you to specify values for properties, you will be prompted for them now. Otherwise, the panel shows a message indicating that there are no properties to configure for the specified module instance.

The Summary panel opens.

18. Verify that the configuration settings are correct.
19. Click **Finish**. The Create Federation Complete portlet opens.
20. Click **Restart WebSphere**.

What to do next

If you are using WebSEAL as your Point of Contact server, configure it now. Do not exit the management console. See:

- "Configuring a WebSEAL point of contact server for the Liberty federation"

Configuring a WebSEAL point of contact server for the Liberty federation

When you plan to use WebSEAL as the Point of Contact server, you must configure it for the Liberty federation.

Before you begin

The 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.

These instructions assume that the WebSEAL point of contact profile has been activated.

About this task

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

The Create Federation Complete portlet provides a button that you can use to obtain the Tivoli Federated Identity Manager configuration utility tool. You must obtain the tool and run it.

To configure WebSEAL as the point of contact server, complete the steps in this procedure:

Procedure

1. After creating the federation, click **Load configuration changes to Tivoli Federated Identity Manager runtime** to reload your changes.

   Note: The management console gives you the option of adding a partner now, but for this initial configuration of the federation other tasks are completed first.
2. Click **Done** to return to the Federations panel.
3. Click **Download Tivoli Access Manager Configuration Tool**.
4. Save the configuration tool to the file system on the computer that hosts the WebSEAL server.
5. Run the configuration tool from a command line. The syntax is:

   ```
   java -jar /download_dir/tfimcfg.jar -action tamconfig
   -cfgfile webseald-instance_name.conf
   ```

   Note: If Federal Information Processing Standards (FIPS) is enabled in your environment, the secure socket connection factory must be specified. For example:

   ```
   java -jar /download_dir/tfimcfg.jar -action tamconfig
   -cfgfile webseald-instance_name.conf -sslfactory TLS
   ```

   You must know the Tivoli Access Manager administration user (default: sec_master) and administration user password. The utility configures endpoints on the WebSEAL server, creates a WebSEAL junction, attaches the appropriate ACLs, and enables the necessary authentication methods.

Example

For example, when you have placed tfimcfg.jar in `/tmp`, and the WebSEAL instance name is `default`, the command is:

```
java -jar /tmp/tfimcfg.jar -cfgfile webseald-default -action tamconfig
```

For more information, see:

- Appendix A, “tfimcfg reference,” on page 753

What to do next

The next task is to export your Liberty federation properties to a file. See “Exporting Liberty federation properties” on page 469.

Configuring WebSphere as a point of contact server

Tivoli Federated Identity Manager is configured by default to use Tivoli Access Manager WebSEAL as the default point of contact server. To configure WebSphere as your point of contact server, you must make a configuration change.

Procedure

1. Log on to the administration console.
2. Click **Tivoli Federated Identity Manager** > **Manage Configuration** > **Point of Contact**.
3. Select **WebSphere**
4. Click **Make Active**.

Results

The WebSphere server is now configured to be the point of contact server.

Exporting Liberty federation properties

When you want to join a federation hosted by a partner, you must supply your Liberty federation configuration properties.

About this task

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

Use the management console to create a metadata file that contains the properties for your federation. Give the metadata file to your federation partner.

Procedure

1. Log on to the management console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations**. The Federations panel opens.
3. Select your Liberty 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 file name. Place the file in an accessible location.
7. Click **Save**.

Exporting SOAP endpoint authentication information to a Liberty federation partner

Supply your partner with any keys, certificates, user names, or passwords needed to complete SSL communication over SOAP ports.

Before you begin

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

About this task

Note: Securing of SOAP ports with SSL, keys, certificates, user names, and passwords, are not required but are typically done to optimize network security.

Liberty provides a SOAP backchannel that is used with browser artifact single sign-on profile, and can be used with additional Liberty profiles that support SOAP binding. The SOAP backchannel can optionally be protected through the use of SSL (HTTPS endpoints). Use of SSL is common for SOAP endpoints.

For Liberty federations, you might also need to provide authentication information (certificates and basic authentication information) to your partner, for use when accessing SOAP endpoints.

This task is done outside of the management console.

Note: If your federation does not use SSL to secure SOAP ports, you can skip this task.

Procedure

1. Provide your partner with a validation certificate. The validation certificate validates the SSL communication that your federation provider sends to the SOAP endpoint of the partner.
2. If you want your partner to authenticate as a client, you must specify whether the partner is to use client certificate authentication or basic authentication. Only one form of authentication can be specified.

   Client certificate authentication

   - When you require client certificate authentication, you and your partner must decide which certificate the partner must present when establishing the SSL session. Choosing a certificate is a business decision. The certificate can be one that your partner already has, or one that you provide to your partner for this purpose.

   Basic authentication

   - When you require basic authentication, you must supply your partner with a user name and password to be used when establishing an authenticated session

Obtaining metadata from a Liberty federation partner

When you want to add a partner to your Liberty single sign-on federation, you must obtain necessary configuration information about their Liberty federation from them.

Before you begin

Your partner must have already installed and configured a Liberty federation. The federation of your partner plays the opposite role to your federation. For example, when your federation is configured as an identity provider, the federation of your partner is configured as a service provider.

About this task

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

Procedure

1. Your partner must export configuration information about the Liberty federation into a metadata file.

   The partner must use the Tivoli Federated Identity Manager management console to export the configuration settings to the metadata file. This is the same feature that you used to provide your configuration settings to your partner.

   The Export feature uses a naming scheme for metadata files, based on the name of the federation and a time stamp. The administrator can override the default name for the metadata file, and change the name to any arbitrary name.
2. Your partner must provide you with the metadata file.

   This action takes place outside of the Tivoli Federated Identity Manager console. Your partner must use whatever process has been agreed to as part of the business agreement that was previously negotiated between the partner companies.
3. Place the metadata file onto the local file system where the Liberty federation configuration is kept. You can choose any location for the metadata file. You have now completed the preparation task. You must later use the Tivoli Federated Identity Manager management console to add the partner to your Liberty federation. The console provides an Add Partner wizard that prompts you to supply the name of the file containing the metadata of your partner. The documentation guides you through this task at the appropriate time.

Importing SOAP endpoint authentication information from a Liberty federation partner

Liberty provides a SOAP backchannel that is used with browser artifact single sign-on profile, and with additional Liberty profiles that support SOAP binding. The SOAP backchannel can optionally be protected through the use of SSL (HTTPS endpoints). Using SSL is common for SOAP endpoints.

About this task

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

For Liberty federations, you might need to obtain authentication information (certificates and basic authentication information) from your partner. Use the authentication information when you want to access SOAP endpoints on the federation of your partner.

This task is done outside of the management console.

Note: If the federation of your partner does not use SSL to secure SOAP ports, you can skip this task.

Procedure

1. Get a validation certificate from your partner. Use the validation certificate to validate SSL communication or messages that your partner sends to your SOAP endpoint.
2. Get the authentication requirement from your partner. Use the authentication requirement to authenticate when your client contacts the SOAP endpoint of your partner.

If your partner wants your client to authenticate, the partner must tell you whether to use client certificate authentication or basic authentication. Only one form of authentication can be specified.

Client certificate authentication

- When your partner requires client certificate authentication, you and your partner must decide which certificate you to present when establishing the SSL session. Choosing a certificate is a business decision. The certificate can be one that you already have, or one that your partner gives you to use for this purpose.

Basic authentication

- When your partner requires basic authentication, your partner must supply you with a user name and password to be used when establishing an authenticated session

3. When your partner uses SSL communication for SOAP ports, you must import the certificate you obtained from your partner. You can import the certificate into any keystore that is managed by the Tivoli Federated Identity Manager key service.

   Note: Tivoli Federated Identity Manager provides a default keystore (DefaultTrustedKeystore) that contains some common CA certificates that you might be able to use as validation certificates. In most cases, however, you must import a certificate obtained from your partner.

   a. Click **Tivoli Federated Identity Manager** > **Key Service**.

      The Keystores panel is displayed.

   b. Select a keystore from the Keystore table. The **View Keys** button is activated.

   c. Click **View Keys**. The Keys panel opens. Keys in the selected keystore are listed.

   d. Click the **Import** button. The Key wizard starts and opens the Keystore Format panel.

   e. Select the appropriate **Keystore format** for the file you want to import.

      (PEM)
      (Privacy-Enhanced Message) Public certificate

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

   f. For PKCS#12, specify whether the keystore contains multiple key pairs.

      1) Select the **Contains multiple key pairs** field when appropriate.
      2) Clear (remove the check mark from) the **Contains multiple key pairs** field when there is only one key pair. The key wizard automatically imports the key.

   g. Click **Next**. The Import Key panel opens.

   h. Enter a fully qualified path in the **Location of Keystore File** field.

      This field is displayed for all format types.

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

   i. If prompted, enter the **Password** for the keystore file.

      Note: This field is displayed only for PKCS#12 format.

   j. If prompted, enter the key name in the **Enter the name of the key you want to import** field.

Note: This field is displayed only for PKCS#12 format files that contain multiple key pairs.

k. Enter a string that names the new key in the **New Key Label** field.

This field is displayed for all format types.

l. Click **Finish** to exit the wizard.

4. When your partner requires client basic authentication, you must retain the user name and password strings. After you have created your federation, and are using the management console to add a partner, the Partner wizard prompts you to enter these values. You do not need to use these strings in any other way.

Adding a partner to a Liberty federation

You can add a partner to the Liberty federation that you have created.

About this task

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

Procedure

1. Copy the metadata file from the partner to an easily accessible location on your computer For example, `/tmp`.

   Note: When the partner also uses Tivoli Federated Identity Manager this file was created on the partner computer by using the management console to export the federation properties.
2. Log on to the IBM Integrated Solutions Console.
3. Select **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Partners**. The Federation Partner wizard starts and opens the Select Federation panel.
4. Select the radio button next to the Liberty federation.
5. Click **Next**.
6. Enter the full path of the metadata file on the local computer in the **Partner's Liberty Metadata File** field. For example:

   ```
   /tmp/libertyfed11_metadata_sp.xml
   ```
7. Click **Next**.
8. The next configuration task is controlled by whether the imported metadata contains information about a key or certificate.

   Typically, the metadata that you imported contains a key that you must import into an existing keystore.

   - When the imported metadata contains information about a key or certificate, the Partner Key panel opens. Continue with step 9
   - When the imported metadata *does not* contain information about a key or certificate, the Server Certificate Validation for SOAP panel opens. Continue with step 10 on page 474
9. Enter the requested information for the Partner key.

   Note: This key or certificate is used to sign Liberty messages, and to sign or validate Liberty tokens. This key is not used for securing SOAP communications over HTTPS.

a. Select a keystore from the keystore table.
b. Enter the password in the **Keystore Password** field.
c. Enter a value in the **Enter a label for your partner's key field**. For example:

   `benefits.example.com`
d. Select or clear the **Require Partner to Sign Liberty Messages** field.
e. Click **Next**.

10. The next configuration step is determined by whether the imported metadata contains a SOAP endpoint that is specified to use HTTPS. Choose one of the following actions:
    - When the imported metadata contains a SOAP endpoint that is specified to use HTTPS, you are prompted to specify the keys or certificates to use. Continue with step 11.
    - When the SOAP endpoint does not use HTTPS, you do not have to specify keys or certificates. Continue with step 15 on page 475.

    Note: In a typical deployment, you must specify keys or certificates for use with the SOAP endpoint. Typical practice for optimal security is to secure this endpoint with HTTPS.
11. When the Server Certificate Validation for SOAP panel opens, complete the following steps:
    a. Select a keystore from the **Keystore** drop-down menu.

       Tivoli Federated Identity Manager supplies a **DefaultTrustedKeyStore**.

       If you are using one of the default CA certificates (based on your agreement with your partner), you can select this keystore. Otherwise, access the keystore where you placed the certificate you obtained from your partner, for use with SSL communication between SOAP endpoints.

       In a test or prototype environment, you can select **DefaultTrustedKeyStore**.
    b. Enter the password in the **Keystore Password** field.

       The default password for **DefaultTrustedKeyStore** is `testonly`.
    c. Click **List Keys**.
    d. Select the radio button for the certificate you want, as indicated by the value in the **Alias** column in the key table.

       In a test or prototype environment, you can select `testwebseal`.
    e. Click **Next**. The Client Authentication for SOAP panel opens.
12. You are prompted to specify whether the partner requires either *client certificate authentication* or *client basic authentication*.

    The partner can require only one of these authentication methods. When you select one of the authentication types on the wizard panel, the panel entries for the other authentication type are deactivated.
    - When the partner requires client certificate authentication, continue with 13.
    - When the partner requires client basic authentication, continue with 14 on page 475.
13. Specify the values for client certificate authentication.
    a. Select the **Partner Requires Client Certificate Authentication** check box.
    b. Select a keystore in the **Keystore** menu.

       The selected keystore is where you placed the certificate to use for client certificate authentication.

c. Enter the password in the **Keystore Password** field.
d. Click **List Keys**.
e. Select the radio button for the appropriate key in the key table.
f. Click **Next**.
g. Continue with 15.

14. Specify the values for client basic authentication.
    a. Select the **Partner Requires Client Basic Authentication** field.
    b. Enter the **Username** and **Password** that you obtained from your partner.
    c. Click **Next**.
    d. Continue with 15.
15. The next panel to open depends on your federation role (identity provider or service provider) and your version of Liberty (1.1 or 1.2). In most cases, you must specify properties for the Liberty token. Select the following instruction that matches your configuration:
    - When adding a service provider partner to an identity provider federation, continue with 16.
    - When adding an identity provider partner to a service provider federation continue with 17.
16. Specify the token module configuration data for adding a service provider partner to an identity provider federation. The required data is identical for Liberty v1.1 or Liberty v1.2. The Liberty v1.1 or v1.2 Token Module Configuration panel opens.
    a. Specify the types of attributes to include in the Liberty token in the **Include the following types of attribute types (a "*" means include all types)** field.

       You can accept the default entry of asterisk (*) to include all types, or specify attribute types.
    b. Click **Next**.
    c. Continue with 15.
17. Specify the token module configuration data for adding an identity provider partner to a service provider federation. Select the action that matches the version of Liberty protocol (v1.1 or v1.2).
    - When adding an identity provider partner to a service provider federation, using Liberty v1.1, no token module configuration is required. The Identity Mapping panel opens. Continue with 18.
    - When adding an identity provider partner to a service provider federation, using Liberty v1.2, complete the following steps:

    a. You can optionally supply a value for the **Username for anonymous users** field. If you are not using this Liberty feature, you can leave this field blank.
    b. Click **Next**. The Identity Mapping panel opens.
    c. Continue with step 18.
18. Choose the following action that matches your use of an identity mapping rule:
    - If you want to use the mapping file default identity mapping rule that you entered in the Federation Creation wizard, complete the following steps:

      a. Leave the identity mapping rule blank.
      b. Click **Next**.

- If you have a customized mapping file for use with this partner, complete the following steps:
  a. Enter the file path to the file.
  b. Click **Import**.
  c. Click **Next**.

  The Summary panel opens.

19. Verify that the settings are correct.
20. Click **Finish**.

    The Add Partner Complete panel opens.
21. Click **Enable Partner** to activate this partner.

    The partner has been added to the federation, but is disabled by default as a security precaution. You must activate the partner.

Configuring the alias service for Liberty

The alias service must be configured to operate with the same user registry as the Tivoli Federated Identity Manager management service. Keep in mind however, that support for Liberty protocol will be deprecated in the later versions of IBM Tivoli Federated Identity Manager.

About this task

These instructions describe configuration of IBMTivoli Directory Server LDAP.

Procedure

1. "Creating an LDAP suffix for the alias service"
2. "Configuring LDAP server settings" on page 477

Creating an LDAP suffix for the alias service

You must create an LDAP suffix `cn=itfim` to enable the alias service to access the LDAP user registry.

About this task

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

To create an LDAP suffix for the alias service, complete the steps in this procedure:

Procedure

1. Stop the IBM LDAP server.

   UNIX

   ```
   # ibmdirctl -D cn=root -w passw0rd stop
   ```

   Windows

   Use the Services icon.
2. Add the suffix:

   ```
   # idscfgsuf -s "cn=itfim"
   ```
3. Start the IBM LDAP server.

   UNIX

   ```
   # ibmdirctl -D cn=root -w passw0rd start
   ```

Windows
Use the Services icon.

4. Use **ldapmodify** to update the LDAP schema file. For example, on UNIX or Linux:
   - IBM Tivoli Directory Server:

     ```
     ldapmodify -D cn=root -w passw0rd -f
       /opt/IBM/FIM/etc/itfim-secuser.ldif
     ```

   - Sun ONE Directory server:

     ```
     ldapmodify -D cn=root -w passw0rd -f
        /opt/IBM/FIM/etc/itfim-secuser-sunone.ldif
     ```

Configuring LDAP server settings

You must configure the alias service with the correct LDAP settings. Keep in mind, however, that support for Liberty protocol will be deprecated in the later versions of IBM Tivoli Federated Identity Manager

About this task

The alias service is used by the Liberty protocol. The alias service communicates with the user registry server (LDAP) to manipulate user identity information. You must configure the alias service with the correct LDAP settings.

Procedure

1. Click **Tivoli Federated Identity Manager → Domain Management → Alias Service Settings**. The Alias Service Settings panel is displayed.
2. In the Root Suffix field, under **LDAP Search Settings**, specify the property for the alias service to use when searching the LDAP user registry.

Table 121. LDAP Search property

Property	Description
Root suffix	Specifies the root suffix where alias settings are written. This property can have 1 value (suffix) only. For example: `cn=itfim`

3. Specify communication properties for the alias service to use when communicating with LDAP servers. Use the menu choices in the **LDAP Environment** portion of the window to specify communication properties.

Table 122. LDAP environment properties

Property	Description
SSL Enabled	Use this check box to specify whether to use Secure Socket Layer (SSL) to secure the communication between the alias service and the LDAP servers. If the LDAP servers are configured to use SSL, the alias service must use SSL when communicating with them. Select **SSL Enabled** when using SSL. Clear the **SSL Enabled** check box when not using SSL.
Keystore	When the **SSL Enabled** check box is selected, select a keystore from the **Keystore** menu list. The selected keystore is the name of the trusted keystore containing the CA certificate of the LDAP server. **Note:** The certificate authority certificates for all LDAP servers must be in the same keystore.

4. Specify configuration parameters for each LDAP server. Use the **LDAP Servers** portion of the window to configure properties for LDAP servers used by the alias service.

You can perform several configuration actions from this section of the window. For each LDAP server, you can specify values for a number of configuration properties.

- Click **Add** to activate the LDAP configuration fields for the selected server.
- Click **Save** to save the LDAP properties that you entered into the configuration fields for a server. When you save the properties, the console inserts the host name and port number into the **LDAP hosts** box.

Table 123. LDAP server properties

Property	Description
LDAP Hostname	The **LDAP Hosts** box lists the configured servers in order of preference. The alias service tries first to contact the server at the start (top) of the list. If that contact is unsuccessful, the alias service attempts to contact the next server on the list. Use the up and down arrows located on the right side of the box to move individual LDAP servers higher or lower in the order of priority.

Port	The port on which the LDAP server listens.
	
	Default port for non-SSL communication:
	`389`
	
	Default port for SSL communication:
	`636`

Bind DN	The distinguished name (DN) that the alias service uses to bind to the LDAP server. Default value: `cn=root`
Bind Password	The password for the DN specified in the **Bind DN** field.
Key Name	The name of the encryption key to use when establishing SSL communication. Select a key name from the list of names. The names on the list are obtained from the keystore that is specified in the **Keystore** field in the **LDAP environment** portion of this configuration window.
Minimum number of connections	The initial number of connections (binds) for the alias service to establish to the LDAP server. The minimum valid number is zero (0). The maximum valid number is limited only by the maximum value supported by the data type. The default value is 2. Use the default value unless you must increase it.
Maximum number of connections	The maximum number of connections (binds) for the alias service to establish to the LDAP server. The maximum valid number is limited only by the maximum value supported by the data type. The default value is 10. Use the default value unless you must increase it.

5. Click **OK** to save configuration properties and exit from the window.

Chapter 32. Planning a WS-Federation single sign-on federation

You must specify values for federation properties when configuring WS-Federation.

WS-Federation protocol defines a standardized, multi-vendor Web-based single sign-on solution based on a collection of integrated Web Services (WS*) standards including WS-Security, WS-Trust, and WS-Federation. When you configure Tivoli Federated Identity Manager, select the WS-Federation Passive Profile.

You should be familiar with the WS-Federation standards documents before implementing a single sign-on federation. The standards specify data exchange and message processing. You should understand what information you are required to provide to your business partners, and what information your partner must provide to you.

Web Services Federation Language (WS-Federation):

`http://wwww.ibm.com/developerworks/library/ws-fed`

The Federation wizard will prompt you to supply values for a number of properties. Most of them can be modified later, after federation creation.

The profile or profiles to use is based on both business policy decisions and network security architecture. Federation partners must agree on the profile choices in order to enable user single sign-on across the federation. The choice must be made prior to configuring the federation.

SAML 2 supports a unique range of single sign-on profiles. The profiles extend beyond specifications for achieving federated single sign-on, and can include other functions such as single logout and federation termination.

Identity provider and service provider roles

Each partner in a federation has a role. The role is either Identity Provider or Service Provider. Understand the behaviour of each role.

- Identity provider

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

  The identity provider is responsible for the following tasks:

  – Directly authenticates the use by validating a user name and password.
  – Indirectly authenticates the user by validating an assertion about the user's identity as presented by a separate identity provider.

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

- Service Provider

  A service provider is a federation partner that provides services to the end user. Typically, service providers do not authenticate users, but instead request authentication decisions from an identity provider. Service providers rely on

identity providers to assert the identity of a user, and rely on identity providers to manage user identities for the federation.

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

WS-Federation single sign-on profiles

The single sign-on profiles enable a client using a Web browser to achieve single sign-on access to resources within a WS-Federation 1.0 federation.

Typically the user wants to access a resource provided by a service provider, and must authenticate with an identity provider in order to be granted that access.

The profile provides a mechanism for the Web user to obtain an authentication assertion that can be used to establish a security context within the federation. Establishment of the security context enables a user to access multiple resources within the federation without having to authenticate more than once.

WS-Federation support two profiles for use with single sign-on sessions:

Browser POST
Browser POST uses a self-posting form during the establishment and use of a trusted session between an identity provider, a service provider, and a client (browser).

WS-Federation supports browser POST by default. No configuration is required.

Single logout
This profile terminates all log in sessions within the federation for a specified user. WS-Federation supports single logout by default. No configuration is required.

WS-Federation single sign-on properties

WS-Federation Realm
The name of the WS-Federation Realm. This name is the unique identifier for this instance of Tivoli Federated Identity Manager The Realm name is included in assertions that are sent to federation partners. Partners rely on finding a known (defined) Realm name in order to accept the assertions. A default value is provided. For example:

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

In the example above, the string `wsfed` is the name of the federation. The endpoint is automatically created. You can accept the default name.

WS-Federation Endpoint
The endpoint for all requests for WS-Federation services. A default value is provided. For example:

```
https://idp.ibm.com/FIM/sps/wsfed/wsf
```

In the example above, the string `wsfed` is the name of the federation. The endpoint is automatically created. You can accept the default name.

WS-Federation token properties

When you create a single sign-on federation, you must configure an instance of a security token module for the federation. The token module corresponds to a security token type that defines the format for the encrypted token that contains user credential information.

The token is exchanged between the identity provider and service provider as part of the authentication and authorization services for the processing of each user access request.

When you use the federation creation wizard to create a WS-Federation single sign-on federation, the SAML 1 token type is automatically selected for you.

When you configure an identity provider, you will be prompted to specify token module properties. When you configure a service provider, you do not have to specify any token module properties.

Number of seconds before the issue date that an assertion is considered valid
Specified during token configuration on identity provider only. Default value 60 seconds. There is no minimum or maximum enforced.

Amount of time the assertion is valid after being issued (seconds)
An integer value that specifies the number of seconds that the assertion remains valid. The default value is 60 seconds. There is no minimum or maximum enforced. Specified during token configuration on identity provider only.

WS-Federation identity mapping

The federation creation wizard will prompt you to specify either an XSLT mapping rule file or a custom mapping module instance.

The XSLT mapping file or custom mapping module instance must be prepared before you configure the federation.

XSLT Transformation for Identity Mapping
Selection of this button in the wizard indicates that you will provide an XSL file containing the identity mapping. Enter the name of a file on the local file system.

Custom Mapping Module Instance
Selection of this button in the wizard indicates that you will provide a custom mapping module instance to use instead of an XSL file. You will be prompted to enter any configuration properties that your custom mapping module instance requires.

Mapping a Tivoli Access Manager credential to a SAML 1 token

Map the Tivoli Access Manager credential to a SAML 1 token when messages are exchanged between partners in a SAML 1.0, SAML 1.1, or WS-Federation federation. You must also map the credential when Tivoli Access Manager manages the user identity information.

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.

When a user request is received (for example, for access to a remote resource), the trust service contacts Tivoli Access Manager and obtains a Tivoli Access Manager credential for the user identity.

In this scenario, the trust service Tivoli Access Manager credential module operates in validate mode. In this mode, it converts the Tivoli Access Manager credential to an Input STS universal user document (In-STSUUSER).

The In-STSUUSER that is created from the Tivoli Access Manager credential module has all of the information from the credential, as shown in Table 124. This information is available for possible use by the trust service module that will build the outgoing token.

Table 124. In-STSUUSER entries generated from a Tivoli Access Manager credential

Tivoli Access Manager credential	**In-STSUUSER element**
User ID	Principal Attr: name
Domain	Principal Attr: domain
Registry ID	Principal Attr: registryid
User UUID	Principal Attr: uuid
Group Name	Group Name
Group Registry ID	Group Attr: registryid
Group UUID	Group Attr: uuid
Other credential entries xxx	Attrlist Attr: xxx

The trust service consults its configuration entry for the federation partner (for example, the destination that hosts a requested resource). The configuration indicates the type of token to be created. In this case, the token type is SAML.

Next, the identity mapping module converts the In-STSUUSER into an Output STS universal user (Out-STSUUSER). The Out-STSUUSER must contain the information needed by the Tivoli Access Manager SAML token module to generate a SAML token.

The Out-STSUUSER must contain the following information in order for the SAML token module to be able to generate a valid SAML token:

Table 125. Out-STSUUSER entries used to generate a SAML token

Out-STSUUSER element	**SAML Token Information**	**Required?**
Principal Attr: Name	AuthenticationStatement/Subject/NameIdentifier	Required
Attribute List	Additional custom attributes	Optional

The mapping module is responsible for:

1. Mapping Principal Attr Name in In-STSUUSER to a Principal name entry in the Out-STSUUSER.

   The type must be valid for SAML. For example:

   `urn:oasis:names:tc:SAML:1.0:assertion#emailAddress`

   Figure 59 shows a sample mapping rule file from the demonstration application mapping file, ip_saml_10.xsl.

```
<!--
 This template replaces the entire Principal element with one that contains
 just the email address (from the ivcred tagvalue_email) and the data type
 appropriate for SAML.
-->
<xsl:template match="//stsuuser:Principal">
 <stsuuser:Principal>
  <stsuuser:Attribute name="name"
                  type="urn:oasis:names:tc:SAML:1.0:assertion#emailAddress">
   <stsuuser:Value>
    <xsl:value-of
    select="//stsuuser:AttributeList/stsuuser:Attribute[@name='tagvalue_email']
            [@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
   </stsuuser:Value>
  </stsuuser:Attribute>
 </stsuuser:Principal>
</xsl:template>
```

Figure 59. XSL code sample showing mapping of a value from a Tivoli Access Manager credential into a Principal name for a SAML token

2. Setting the authentication method to the "password" mechanism, regardless of the value obtained from the Tivoli Access Manager credential. This action is required by the SAML standard.

   Figure 60 shows a sample mapping rule file from the demonstration application mapping file, ip_saml_10.xsl.

```
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList>
                ....
   <!-- First the authentcation method attribute -->
   <stsuuser:Attribute name="AuthenticationMethod"
                          type="urn:oasis:names:tc:SAML:1.0:assertion">
    <stsuuser:Value>urn:oasis:names:tc:SAML:1.0:am:password</stsuuser:Value>
   </stsuuser:Attribute>
                ....
                ....
     </stsuuser:AttributeList>
 </xsl:template>
```

Figure 60. XSL code sample showing assignment of authentication method as an Attribute for a SAML token

3. Populating the attribute statement of the SAML assertion with the attributes in the AttributeList in the In-STSUUSER. This information becomes custom information in the SAML token.

   There can be custom attributes that are required by applications that will make use of the information to be transmitted between federation partners.

   Figure 61 on page 484 shows the mapping of custom attributes in the sample mapping file for the Tivoli Federated Identity Manager demonstration

application.

```
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList>
          ....
              ....
       <!-- Now the commonName attribute -->
   <stsuuser:Attribute name="commonName"
                          type="http://example.com/federation/v1/commonName">
    <stsuuser:Value>
     <xsl:value-of
     select="//stsuuser:AttributeList/stsuuser:Attribute[@name='tagvalue_name']
             [@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
    </stsuuser:Value>
   </stsuuser:Attribute>

   <!-- Now the ssn attribute -->
   <stsuuser:Attribute name="ssn"
                          type="http://example.com/federation/v1/namevalue">
    <stsuuser:Value>
     <xsl:value-of
      select="//stsuuser:AttributeList/stsuuser:Attribute[@name='tagvalue_ssn']
             [@type='urn:ibm:names:ITFIM:5.1:accessmanager']/stsuuser:Value" />
    </stsuuser:Value>
   </stsuuser:Attribute>
              ....
    </stsuuser:AttributeList>
 </xsl:template>
```

Figure 61. XSL code sample showing assignment of optional attributes for a SAML token

4. Populating custom attributes. The GroupList element of the In-STSUUSER is not read by the SAML token module. However, information in this element can optionally be used to populate custom attributes of the Out-STSUUSER.

   Figure 62 shows the optional assignment of a GroupList value to an attribute. This code sample is from the demonstration application mapping file ip_saml_10.xsl.

```
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList>
          ....
       <!-- Now the role attribute (can be multi-valued) -->
   <stsuuser:Attribute name="role" type="http://example.com/federation/v1/role">
    <xsl:for-each select="//stsuuser:GroupList/stsuuser:Group">
     <stsuuser:Value>
      <xsl:value-of select="@name" />
     </stsuuser:Value>
    </xsl:for-each>
   </stsuuser:Attribute>
              ....
    </stsuuser:AttributeList>
 </xsl:template>
```

Figure 62. XSL code sample showing optional assignment of GroupList value to an attribute for a SAML token

Mapping a SAML 1 token to a Tivoli Access Manager credential

The service provider receives a SAML token. The SAML token module, operating in validate mode, creates an In-STSUUSER document from the SAML token.

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 126 shows the information from the token that is converted into an In-STSUUSER document.

Note: This topic applies to both SAML 1.0 and SAML 1.1 tokens.

Table 126. SAML token information that is converted into a STS universal user document

SAML Token Information	In-STSUUSER element	Required for Out-STSUUSER?
AuthenticationStatement/Subject/ NameIdentifier	Principal Attr: Name	Required
Additional custom attributes	Attribute List	Optional

Note: The SAML token module does not populate the GroupList element in the In-STSUUSER document.

The trust service must convert this information to a Tivoli Access Manager credential, in order to make an authorization decision on the request from the user identity. The identity mapping module converts the In-STSUUSER data into an Out-STSUUSER XML file.

- The NameIdentifier is used to populate the `name` attribute of the Principal.

  Figure 63 shows the assignment of a set value for the Principal name. This code sample is from the demonstration application mapping file sp_saml_1x.xsl.

```
<!--
  This will replace the principal name (which was the email address in
  the SAML assertion) with the user "me_chris".
 -->
 <xsl:template match="//stsuuser:Principal/stsuuser:Attribute[@name='name']">
  <stsuuser:Attribute name="name" type="urn:ibm:names:ITFIM:5.1:accessmanager">
            <stsuuser:Value>me_chris</stsuuser:Value>
            </stsuuser:Attribute>
 </xsl:template>
```

Figure 63. XSL code sample showing assignment of a value for the Principal name for a SAML token.

- Other information from the token is used to populate Attributes in the Attribute List.

  Figure 64 on page 486 shows the optional assignment of additional values to attributes. This code sample is from the demonstration application mapping file sp_saml_1x.xsl.

```
<xsl:template match="//stsuuser:AttributeList">
  <stsuuser:AttributeList>
        ....
     <!-- The tagvalue_name attribute -->
   <stsuuser:Attribute name="tagvalue_name"
                        type="urn:ibm:names:ITFIM:5.1:accessmanager">
    <stsuuser:Value>
    <xsl:value-of
  select="//stsuuser:AttributeList/stsuuser:Attribute[@name='commonName']
      [@type='http://example.com/federation/v1/commonName']/stsuuser:Value" />
    </stsuuser:Value>
   </stsuuser:Attribute>

   <!-- The tagvalue_ssn attribute -->
   <stsuuser:Attribute name="tagvalue_ssn"
                        type="urn:ibm:names:ITFIM:5.1:accessmanager">
    <stsuuser:Value>
     <xsl:value-of
              select="//stsuuser:AttributeList/stsuuser:Attribute[@name='ssn']
          [@type='http://example.com/federation/v1/namevalue']/stsuuser:Value" />
    </stsuuser:Value>
   </stsuuser:Attribute>
              ....
     </stsuuser:AttributeList>
 </xsl:template>
```

Figure 64. XSL code sample showing optional assignment of attributes for a SAML token.

Chapter 33. Configuring a WS-Federation single sign-on federation

To configure a WS-Federation single sign-on federations, you must create the federation, add your partner to your federation, and provide your partner with configuration information from your new federation.

1. "Creating a WS-Federation single sign-on federation"
2. "Configuring WebSEAL as the point of contact server" on page 488
3. "Exporting WS-Federation properties" on page 490
4. "Obtaining configuration information from a WS-Federation partner" on page 490
5. "Adding a partner to your WS-Federation single sign-on federation" on page 492

Creating a WS-Federation single sign-on federation

Use the WS-Federation passive protocol in to create and configure a federation through the Federation wizard.

Procedure

1. Log on to the Integrated Solutions Console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations**. The Current Domain and Federations portlets open.
3. Click **Create**. The Federation wizard starts. The General Information panel opens.
4. Enter a name for the federation and select a role.
5. Click **Next**.
6. Enter the contact information.
7. Click **Next**.
8. Select the WS-Federation Passive Protocol.
9. Click **Next**. The Point of Contact Server panel opens.
10. Enter the point of contact address.
11. Click **Next**.
12. Choose one of the following options:
    - When configuring a service provider, the next step is identity mapping. Continue with step 14
    - When configuring an identity provider, the Configure Security Token panel opens. Specify the requested token properties.

      See Chapter 32, "Planning a WS-Federation single sign-on federation," on page 479.
13. Click **Next**. The Identity Mapping Options panel opens.
14. Select one of the radio buttons.
    - Use XSL Transformation for Identity Mapping

      Indicates that you will provide an XSL file containing the required identity mapping.

a. When you select this choice and click **Next**, the Identity Mapping panel opens. Enter the name of a file on the local file system that contains the identity mapping rule in the **XSLT File Containing Identity Mapping Rule** field.

   This is a file that you have prepared prior to this installation.

   Optionally, you can click the **Browse** button to locate the file on the local file system.

b. Click **Next**.

   An error opens if the file cannot be found or if the file does not contain valid XSLT (eXtensible Stylesheet Language Transform).

- Use Custom Mapping Module Instance

  Indicates that you will provide a custom mapping module instance to use instead of an XSL file.

  a. When you select Use Custom Mapping Module Instance, a table of Module Instances opens. Select the radio button for the module instance to use and click **Next**.

  b. When you custom mapping module instance requires you to specify values for properties, you will be prompted for them now. Otherwise, the panel opens a message indicating that there are no properties to configure for the specified module instance.

The Summary panel opens.

15. Verify that the configuration settings are correct.
16. Click **Finish**. The Create Federation Complete portlet opens.
17. Click **Restart WebSphere**.

What to do next

If you are using WebSEAL as your Point of Contact server, configure it now. Do not exit the management console. See:

- "Configuring WebSEAL as the point of contact server"

Configuring WebSEAL as the point of contact server

When you plan to use WebSEAL as the point of contact server, you must configure it for the WS-Federation single sign-on federation.

Before you begin

The 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.

These instructions assume that the WebSEAL point of contact profile has been activated.

About this task

The Federation wizard provides a button that you can use to obtain the Tivoli Federated Identity Manager configuration utility tool. You must obtain the tool and

run it. To configure WebSEAL as the point of contact server, complete the steps in this procedure:

Procedure

1. After creating the federation, click **Load configuration changes to Tivoli Federated Identity Manager runtime** to reload your changes.

   Note: The management console gives you the option of adding a partner now, but for this initial configuration of the federation other tasks are completed first.
2. Click **Done** to return to the Federations panel.
3. Click **Download Tivoli Access Manager Configuration Tool**.
4. Save the configuration tool to the file system on the computer that hosts the WebSEAL server.
5. Run the configuration tool from a command line. The syntax is:

   ```
   java -jar /download_dir/tfimcfg.jar -action tamconfig
   -cfgfile webseald-instance_name.conf
   ```

   Note: If Federal Information Processing Standards (FIPS) is enabled in your environment, the secure socket connection factory must be specified. For example:

   ```
   java -jar /download_dir/tfimcfg.jar -action tamconfig
   -cfgfile webseald-instance_name.conf -sslfactory TLS
   ```

   You must know the Tivoli Access Manager administration user (default: sec_master) and administration user password. The utility configures endpoints on the WebSEAL server, creates a WebSEAL junction, attaches the appropriate ACLs, and enables the necessary authentication methods.

Example

For example, when you have placed tfimcfg.jar in `/tmp`, and the WebSEAL instance name is `default`, the command is:

```
java -jar /tmp/tfimcfg.jar -action tamconfig -cfgfile webseald-default
```

For more information, see Appendix A, “tfimcfg reference,” on page 753.

What to do next

The next task is to manually export your WS-Federation properties to your partner. See “Exporting WS-Federation properties” on page 490.

Configuring WebSphere as a point of contact server

Tivoli Federated Identity Manager is configured by default to use Tivoli Access Manager WebSEAL as the default point of contact server. To configure WebSphere as your point of contact server, you must make a configuration change.

Procedure

1. Log on to the administration console.
2. Click **Tivoli Federated Identity Manager** > **Manage Configuration** > **Point of Contact**.
3. Select **WebSphere**.

4. Click **Make Active**.

Results

The WebSphere server is now configured to be the point of contact server.

Exporting WS-Federation properties

When you want to join a federation hosted by another business partner, you must supply your federation configuration properties. You can easily export the properties and provide it to your partner.

About this task

For WS-Federation, you must manually prepare a file that contains the configuration properties. Give this file to your federation partner.

Procedure

1. Log on to the management console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations**. The Federations panel opens.
3. Select your WS-Federation single sign-on federation from the table.
4. Display the federation properties. Obtain the properties listed in "WS-Federation properties to exchange with your partner" on page 491
5. Deliver the file to your partner, in the manner specified in the business agreement between your company and your partner's company.

What to do next

You need to provide this file to your partner, when your partner wants to add your configuration information to their WS-Federation single sign-on federation.

Obtaining configuration information from a WS-Federation partner

You must obtain configuration information from your WS-Federation your partner.

Before you begin

When you want to add your business partner as a partner to your WS-Federation single sign-on federation, you must obtain from them the necessary configuration information about their WS-Federation single sign-on federation.

Your partner must have already installed and configured a WS-Federation single sign-on federation. The federation of the partner plays the opposite role to your federation. For example, when your federation is configured as an identity provider, the federation of your partner is configured as a service provider.

You obtain the information by having your partner manually assemble the configuration properties for their federation. The partner must then provide the information to you through a method that has been agreed upon as part of the business agreement between you and your partner.

Procedure

1. Your partner must use the management console to collect the properties for their federation. The partner should provide you with the properties listed in "WS-Federation properties to exchange with your partner."
2. You partner should deliver the file to you, in the manner specified in the business agreement between your company and your partner's company.

WS-Federation properties to exchange with your partner

You and your partner must be prepared with the information that you must exchange before you can add your partner to a federation.

Federation properties

Table 127. WS-Federation properties

Property	Description
Federation name	A character string that names the federation
Role	Identity provider or service provider
Protocol	WS-Federation Passive Profile
Company name	The name of the company that created this federation. (Required)
Company URL	A URL for the company that created this federation. (Optional)
First Name and Last Name	The name of the person who serves as contact for other companies in the federation. (Optional)
Email address	The email address of the person who serves as contact for other companies in the federation. (Optional)
Phone number	The telephone number of the person who serves as contact person for other companies in the federation. (Optional)
Contact Type	A string that describes a business role type such as technical or support. (Optional)

WS-Federation data

Table 128. WS-Federation data

Property	Description
WS-Federation Realm	The unique name of the WS-Federation Realm. For example: `https://idp.example.com/FIM/sps/wsfed/wsf`
WS-Federation Endpoint	The endpoint of your partner for all the WS-Federation services requests. For example: `https://idp.example.com/FIM/sps/wsfed/wsf`
Maximum Request Lifetime (in seconds)	The maximum length of time, in seconds, that a request or message that is received from a WS-Federation partner is valid.

SAML token module configuration

Table 129. SAML token module properties

Property	Description
Enable the Signing of Assertions	Indicates whether the identity provider signs assertions before sending them to the service provider partner.
Select Key for Signing Assertions	The name of the key to use when signing assertions. Specified for a service provider partner.
Signature Algorithm for signing SAML Assertions	Specifies the signature algorithm to use to sign the SAML assertion. Select one from the following options: • RSA-SHA1 • DSA-SHA1 • RSA-SHA256
Include the following attribute types (a '*' means include all types)	The types of attributes to include in the SAML token module. Specified for a service provider partner.
Enable signature validation	When selected, indicates that the service provider validates the signature on assertions received from the identity provider partner. Specified for an identity provider partner
Select Validation Key	The name of the key to use when validating signatures. Specified for an identity provider partner.

Adding a partner to your WS-Federation single sign-on federation

You can use the administration console to add a partner to a WS-Federation single sign-on federation.

About this task

The configuration steps are the same for adding all partners. The configuration properties differ for identity provider and service provider partners. The Partner wizard prompts you for the necessary properties.

Procedure

1. Log on to the IBM Integrated Solutions Console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Partners**. The Federation Partners panel opens.
3. Click **Create**. The Select Federation panel opens.
4. Select the federation to which you would like to add a partner.
5. Click **Next**. The Contact Information panel opens.
6. Enter the Contact properties.

   The company name is required. The other fields are optional.
7. Click **Next**. The WS-Federation Data panel opens.
8. Enter the requested properties.
9. Click **Next**. The Configure Security Token panel opens.

10. Enter the configuration properties for the federated security token.

    The configuration properties are specific to the partner role:

    - When adding an identity provider partner:
      a. When assertions should be signed click **Enable the Signing of Assertions**. When you select this check box, you must specify a key for signing assertions. Select the **Keystore**, enter the **Keystore Password**, click **List Keys** and select the key from the key table.
      b. Optionally specify attributes in the field: **Include the following attribute types (a '*' means include all types**.
      c. Click **Next**.
    - When adding a service provider partner:
      a. When signatures should be validates click **Enable Signature Validation**. When you select this check box, specify a key to use for validating signatures. Select the **Keystore**, enter the **Keystore Password**, click **List Keys** and select the key from the key table.
      b. Click **Next**.

    The Identity Mapping Options panel opens.

11. Select one of the radio buttons.

    - Use XSL Transformation for Identity Mapping

      Indicates that you will use an XSL file to provide any required identity mapping.

      a. When you select this choice and click **Next**, the Identity Mapping panel opens. Leave the identity mapping blank when you want to use the default identity mapping rule that you entered in the Federation Creation wizard.

         When you want to override the default mapping rule with a rule specific to this partner, enter the name of a file on the local file system that contains the identity mapping rule in the **XSLT File Containing Identity Mapping Rule** field.

         Optionally, you can click the **Browse** button to locate the file on the local file system.

      b. Click **Next**.
    - Use Custom Mapping Module Instance

      Indicates that you will provide a custom mapping module instance to use instead of an XSL file.

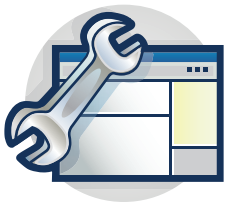
      a. When you select Use Custom Mapping Module Instance, a table of Module Instances opens. Select the radio button for the module instance to use and click **Next**.
      b. When you custom mapping module instance requires you to specify values for properties, you will be prompted for them now. Otherwise, the panel opens a message indicating that there are no properties to configure for the specified module instance.

    The Summary panel opens.

12. Verify that the settings are correct and click **Finish**. The Add Partner Complete panel opens.
13. Click **Enable Partner** to activate this partner.

    The partner has been added to the federation, but is disabled by default as a security precaution. You must enable the partner.

Part 4. Web services security management configuration



The topics in the Configuration section provide a step-by-step guide to configuring Web services security management for Tivoli Federated Identity Manager.

Read the overview section first:

Chapter 34, “Web services security management configuration,” on page 497

Chapter 34. Web services security management configuration

Configuration of Web services security management starts with the establishment of a Tivoli Federated Identity Manager domain. When the domain is established, you can configure the Web services security management component.

Configuration of Web services security management consists of these steps:

1. Configuration of a Tivoli Federated Identity Manager domain.

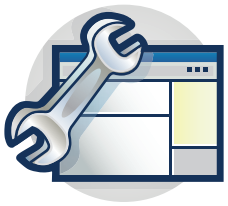
   The deployment of a Tivoli Federated Identity Manager scenario requires the creation of a Tivoli Federated Identity Manager domain.

   You must create and configure a domain before you can configure the Web services security management component.

   See Chapter 3, "Domain configuration," on page 23.
2. Configuration of the Web services security management component.

   The component can be configured in many different ways, to reflect the deployment scenarios. Configuration is described in detail in the *IBM Tivoli Federated Identity Manager Web Services Security Management Guide*.

Part 5. Configuring security token service



The topics in the Configuration section provide a step-by-step guide to configuring a security token service as part of an integrated solution for management of user identities in a distributed network environment.

This section describes the deployment of a Kerberos delegation security token service module as support for a Kerberos junction solution provided by combining Tivoli Federated Identity Manager with Tivoli Access Manager for e-Business WebSEAL, along with WebSphere and additional products.

First read the overview of the deployment scenario:

Chapter 35, “Kerberos constrained delegation overview,” on page 501

Chapter 35. Kerberos constrained delegation overview

Tivoli Federated Identity Manager provides a security token service (STS) that can exchange security token formats. This function is used to move user credential information between different token formats, as needed for different applications.

Note: IBM deprecated the Tivoli Federated Identity Manager Security Token Service (STS) Client in this release.

If you use WebSphere 6.X, you can still use the Tivoli Federated Identity Manager Security STS client while Tivoli Federated Identity Manager supports WebSphere 6.X. When Tivoli Federated Identity Manager discontinues its support for WebSphere 6.X, use WebSphere Application Server version 7 Update 11 and later. See WS-Trust client API and WS-Trust Clients for details.

The STS is an integral part of the Tivoli Federated Identity Manager single sign-on solutions, but can also be used standalone. Tivoli Federated Identity Manager can be integrated into various heterogeneous network deployments given the stand-alone capability.

One such deployment is an environment that uses Microsoft Windows integrated authentication (SPNEGO) with Kerberos tickets. In this environment, Tivoli Federated Identity Manager can be deployed to take user credentials and convert them to the necessary Kerberos format.

To use this capability, Tivoli Federated Identity Manager includes a security token service module specifically for Kerberos constrained delegation. The Kerberos delegation module facilitates the issuing of Kerberos Constrained Delegation application service tickets, also known as Service for User To Proxy (S4U2Proxy).

The module supports only the *issuing* of tokens, and *only* Windows Kerberos application service tickets through the Constrained Delegation model.

A main feature of the Kerberos constrained delegation model is that the password of the user that needs the Kerberos service ticket can be concealed from the application generating the ticket. In this case the application is WebSphere plus Tivoli Federated Identity Manager. The application must know only the user name of the user, and the service principal name (SPN) of the destination Kerberos service.

The Kerberos constrained delegation STS module is primarily intended to enable Tivoli Access Manager WebSEAL to support single sign-on across Kerberos junctions. These junctions are junctions to a Web server that is configured for Integrated Windows Authentication (SPNEGO).

WebSEAL can maintain a user session with whatever authentication mechanism it chooses, and then connect to a Web server (for example, IIS) by using the SPNEGO authentication flow. This authentication flow uses a Kerberos ticket.

The use of Kerberos credentials for single sign-on to junctions provides the following capabilities:

- Kerberos credentials are easily used by ASP.NET web applications without requiring special code to be deployed.

- Kerberos credentials can be forwarded across applications while maintaining a cryptographic signature, providing stronger security.

Note: This module is different from the Tivoli Federated Identity Manager native Java Kerberos STS module. The Java Kerberos module supports the generic issuing and validation of other Kerberos tickets.

More information about the Windows Kerberos extensions can be found at:

- http://technet2.microsoft.com/WindowsServer/en/Library/c312ba01-318f-46ca-990e-a597f3c294eb1033.mspx
- http://msdn2.microsoft.com/en-us/library/aa480585.aspx
- http://msdn.microsoft.com/msdnmag/issues/03/04/SecurityBriefs/

Overview of Kerberos constrained delegation with WebSEAL junctions

Tivoli Federated Identity Manager uses the Kerberos constrained delegation security token service (STS) module to generate Kerberos tokens.

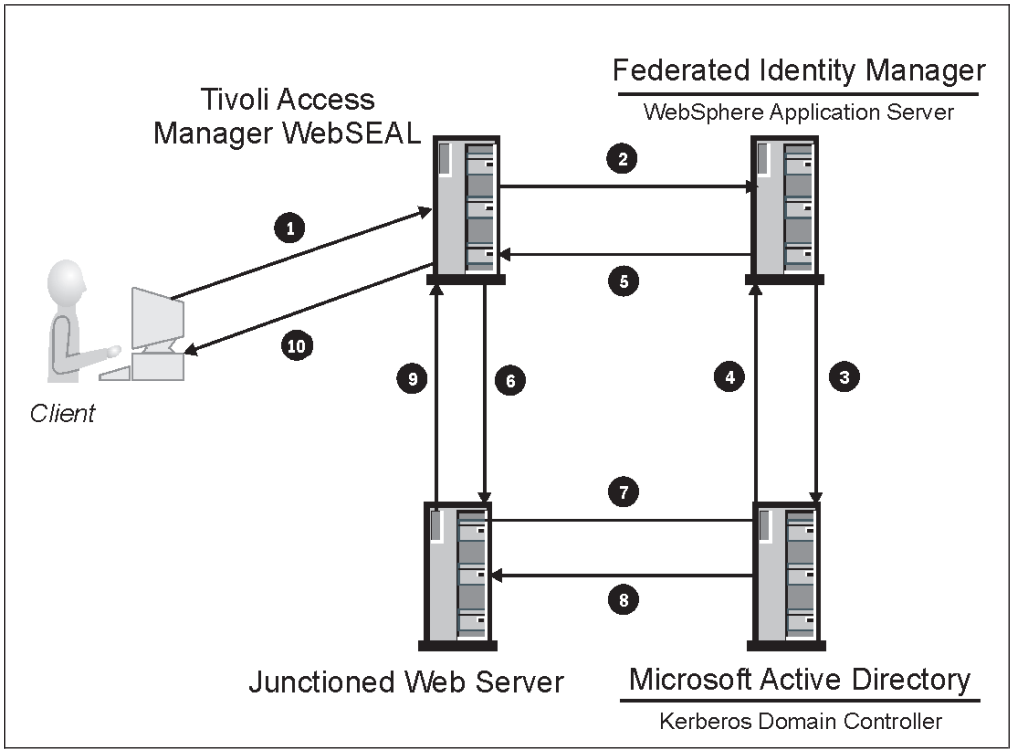


Figure 65. Kerberos constrained delegation with a WebSEAL junction

WebSEAL retrieves the Kerberos tokens by delegating the token request to the STS module.

Figure 65 shows a sample deployment of the applications involved in achieving this type of single sign-on. The diagram also shows how the messages flow between the different physical components.

1. Client uses the standard Tivoli Access Manager authentication process to authenticate to WebSEAL over HTTPS or HTTP and requests an object on the

junctioned server. WebSEAL authorizes the request from the client, and determines that a Kerberos ticket is needed to access the junctioned application.

2. WebSEAL generates a WS-Trust issue key message intended for the Tivoli Federated Identity Manager server. A single WS-Trust issue key can be used to request multiple Kerberos tokens. WebSEAL opens a connection to the WebSphere server running Tivoli Federated Identity Manager. WebSEAL sends the SOAP request to the WebSphere server.
3. The Tivoli Federated Identity Manager server running on the WebSphere server verifies that the WebSEAL server is authorized to start the security token service (STS).

   The STS then starts a Tivoli Federated Identity Manager trust module to request the configured number of Kerberos tickets for the junctioned Web server on behalf of the client. The trust module uses Kerberos, over TCP or UDP port 88 to communicate with the Active Directory domain controller.
4. The Active Directory domain controller verifies that the Tivoli Federated Identity Manager server is authorized to request tickets for the junctioned Web server on behalf of the user. The Active Directory domain controller returns the configured number of Kerberos tickets to the Tivoli Federated Identity Manager runtime.
5. The Tivoli Federated Identity Manager runtime returns the tickets as a SOAP response to the WebSEAL server.
6. The WebSEAL server caches the Kerberos tickets and forwards one of the tickets along with the client request to the junctioned Web server over either HTTP or HTTPS.
7. The junctioned Web server requests validation of the Kerberos ticket from the Kerberos domain controller (KDC). The KDC is shown here as being the same system as the Active Directory server.
8. The KDC verifies that the Kerberos ticket is valid. The Kerberos ticket is used as proof of the client identity, which might also be used for further authorization checks.
9. The junctioned Web server returns an HTTP response to WebSEAL.
10. WebSEAL returns the HTTP response to the client.

A new Kerberos ticket is sent along with the request to the junctioned Web server under the following circumstances:

- on each subsequent request from the same client, and
- on the same login session to the same junctioned Web server.

The new Kerberos ticket is either taken from the WebSEAL cache of Kerberos tickets or a request is sent to the WebSphere server running Tivoli Federated Identity Manager to obtain a new set of Kerberos tickets.

Deployment overview

The Kerberos deployment has specific software prerequisites. This section lists those prerequisites and the tasks involved in the deployment.

Software prerequisites

- Tivoli Federated Identity Manager must run on Windows 2003 Server Service Pack 2 or later.

The service pack is required due to a known memory leak in the Windows Isass.exe process on earlier versions. See http://support.microsoft.com/kb/907524/

- The WebSEAL server can run on any supported platform.

  The WebSEAL server does *not* need to be part of the Active Directory domain.
- WebSphere can be deployed either in standalone mode or in cluster mode. All WebSphere servers in the cluster should be installed on Windows systems, and should be part of the domain.
- When the Tivoli Access Manager users are stored in Active Directory, the Tivoli Access Manager policy server must be on Windows and be a member of the domain.
- All domain controllers in the Active Directory domain should run at the Windows Server 2003 functional level.
- The Tivoli Federated Identity Manager support for Kerberos delegation modules is not included in the Tivoli Federated Identity Manager Business Gateway product.

Deployment task overview

1. Enable integrated Windows authentication
2. Configure Active Directory and WebSphere for constrained delegation
3. Install and configure a Tivoli Federated Identity Manager domain and runtime
4. Configure a Kerberos module instance and trust chain for the Kerberos constrained delegation STS module
5. Configure WebSEAL to support a Kerberos junction

Table 130. Example server hostnames used in this documentation

Server role	Example Value
Backend server (junctioned Web server)	mydataserver.example.com
WebSEAL server	websealhost.example.com
Active Directory hostname	activedirectoryhost.example.com

Chapter 36. Enabling integrated Windows authentication

These instructions describe how to configure Microsoft IIS for SPNEGO authentication.

Before you begin

These instructions assume that you have Windows Server 2003 deployed with Active Directory. These steps must be completed before you can set up constrained delegation.

Procedure

1. On the domain controller, select **Start** > **Programs** > **Administrative Tools** > **Active Directory Users and Computers**.
2. Create a user that acts as a proxy for the IIS server. For example, `iisuser`.
3. Specify the user password as `never expires`.
4. Open a command prompt.
   a. Change directory to `C:\Program Files\Support Tools`.
   b. Enter the appropriate **ktpass** command.

      Syntax for ktpass:

      ```
      ktpass -princ HTTP/IIS_server_name.domain_name@DOMAIN_NAME
        -mapuser IIS_user_name -mapOp set
      ```

      where:

      - `-princ` is the Principal Name, in the form *user@REALM*
      - `-mapuser` maps the -princ value to this use account. This is not done by default.
      - `-mapOp` specifies how to set the mapping attribute: `set` *set_value*
5. View the account properties for `iisuser`. Verify that the field **User logon name** is set to the following value:

   `HTTP/`*IIS_server_name.domain_name*

   For example:

   `HTTP/mydataserver.example.com`
6. Configuring the Application Pool Identity.
   a. On the IIS server system, select **Start** > **Programs** > **Administrative Tools** > **Internet Information Service (IIS) Manager**.
   b. Select ***your_server_name/IIS name*** > **Programs** > **Administrative Pools** > **Default App Pool**.
   c. Right-click and select **Properties**.
   d. Select the identity tab, and specify the domain identity for your IIS user (for example `iisuser`).

   For detailed instructions on the Windows task *Configuring Application Pool Identity with IIS 6.0*, see

   http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/f05a7c2b-36b0-4b6e-ac7c-662700081f25.mspx?mfr=true.
7. Open Windows Explorer.
   a. Go to `C:\WINNT\Microsoft.NET\Framework\v1.1.4322\Temporary ASP.NET Files`.

b. Select **Properties**.
c. Select the **Security** tab.
d. Grant domain user `iisuser` full control over the directory.

8. Go to the IIS system, and select **Start** > **Programs** > **Administrative Tools** > **Computer Management**.
   a. Open **Local Users and groups**.
   b. Open **groups**.
   c. Right click on the local group `IIS_WPG`.
   d. Select properties.
   e. Select **Add**.
   f. Add the domain user (in our case, `iisuser`) to this local group.
9. On IIS system, open the server's local security policy.
   a. Click **Start** > **Run** and enter `secpol.msc`.
   b. Expand local polices and browse to User Rights assignment.
   c. Open up the **Log on as Service** right.

      Note that any account or group in this list can logon as a service.
   d. Click **Add User or Group**.
   e. Enter (or browse for) the domain user `iisuser` account.
   f. When the right is granted, reboot the server.

      The system reboot is required because security settings are applied during the startup phase of any Windows 2003 Server machine.
10. On the IIS system, select **Start** > **Programs** > **Administrative Tools** > **IIS Manager**.
    a. Open the local computer.
    b. Right-click on the DefaultAppPool.
    c. Select **Recycle** to restart the pool.
11. Open a browser and access `http://web_server`.

    When this is a new IIS server without existing content, you should see the IIS `Under Construction` page. When the IIS server has content, you should be able to see the content.
12. On the IIS system, select **Start** > **Programs** > **Administrative Tools** > **IIS Manager**.
    a. Right-click on Default Web Site
    b. Select Properties and select the Directory Security tab.
    c. Click the **Edit** button next to `Enable anonymous access`, and edit the authentication messages for this resource.
    d. Disable anonymous access.
    e. Enable integrated windows authentication.
    f. Click **OK**.
    g. Click **OK** again.
13. Open your browser and access http://*web_server*. You are prompted to log on.
14. Enter a valid domain user. For example, user@mydomain.com. When the log on is successful you can view the IIS content.

Chapter 37. Configuring Active Directory and WebSphere for constrained delegation

You must configure Active Directory and WebSphere for your Kerberos delegation to work.

About this task

The WebSphere node agent that hosts the Tivoli Federated Identity Manager runtime needs to run under a special account in Active Directory in order to have permission to obtain Kerberos tickets for other users and a constrained set of targets. Before your Kerberos delegation trust chain can work, you must complete the following tasks:

- Create the account.
- Set the appropriate options.,
- Modify the WebSphere service to use the account.

The following instructions describe how to complete these tasks.

Procedure

1. Verify that DNS is configured correctly on the Active Directory domain controller.

   The DNS server must be configured for both forward and reverse lookups. Each host in the Active Directory domain must be configured to use the Domain Controller's DNS server.

   To verify, use **nslookup** commands for both hostname and IP address on computers in the domain. The results of the **nslookup** commands should show that the domain part of the resolved name is the domain of the domain controller.
2. Ensure that Time Services are running on all machines in the Active Directory domain and that the clocks of all machines are synchronized.
3. Verify that the Windows Server 2003 system (or multiple systems, when deployed in a WebSphere cluster) is configured into an Active Directory domain. The server can optionally be a domain controller.
4. Verify that all domain controllers in the domain are running at the Windows Server 2003 functional level. To do this:
   a. Open the Active Directory Users and Computers control panel.
   b. Right-click on the domain and select **Raise Domain Function Level**.
   c. Select `Windows Server 2003` and click **OK**.

   The Raise Domain Functional Level window is displayed. It should contain the messages:

   ```
   Current domain functional level
   Windows Server 2003

   This domain is operating at the highest possible functional level.
   ```
5. On the domain controller, create a user in Active Directory for delegation. The WebSphere server that hosts the Tivoli Federated Identity Manager runtime runs as this user identity.
   a. Create a user. For example, `tfimdeleguser`. You can use a different user identity. This user name will be used in these instructions.

b. Select the **Password never expires** check box.

   Note: You can optionally set the password to expire. If you do, then when you change it in the future, you will also need to reset the password for the WebSphere node agent Windows service.

6. On the domain controller, add the `tfimdeleguser` user to the Domain administrative group. To verify the settings:
   a. Select **Active Directory Users and Computer**.
   b. For the domain, click **Users** and click **Domain Admins**.
   c. Select the **Members** tab. Verify that the tfimdeleguser is listed as a group member.
7. Ensure that the Microsoft Support Tools are installed on the domain controller. For example to obtain the Windows Server 2003 Service Pack 1 32-bit Support Tools:

   http://www.microsoft.com/downloads/details.aspx?FamilyId=6EC50B78-8BE1-4E81-B3BE-4E7AC4F0912D&displaylang=en
8. On the domain controller, create an service principal name (SPN) for the user `tfimdeleguser`. To complete this task::
   a. Open a command prompt on the domain controller where the support tools are installed.
   b. Enter the **setspn** command.

      The syntax for the command is:

      ```
      setspn -A tfim/<tfim_delegation_user> <tfim_delegation_user>
      ```

      For example:

      ```
      setspn -A tfim/tfimdeleguser tfimdeleguser
      ```
9. On the domain controller, go to **Active Directory Users and Computers** and open the properties for the user `tfimdeleguser`.
   a. Select the **Delegation** tab.

      Note: If you do not see the **Delegation** tab, return to the previous step and ensure that the setspn command runs successfully.
   b. Select the **Trust this user for delegation to specified services only** radio button.
   c. Select the **Use any authentication protocol** radio button.
   d. Click the **Add** button on the Delegation tab.
   e. Add the target services to which tfimdeleguser can delegate. These are the target services for constrained delegation. In this example, the IIS Web server runs as the user.
   f. Click the **Users or Computers** button to search for particular services.
   g. Select the domain user (service) that runs as the IIS server for the WebSEAL Kerberos junction.

   When you are finished, the **Delegation** tab should show a target service in the window **Services to which this account can present delegated credentials**.

   For example, the window could show a **Service Type** of `HTTP`, with **User or Computer** showing a host and domain name such as `mydataserver.example.com`

   Select the `HTTP/mydataserver.example.com` entry. Press **OK** to continue.
10. Add the tfimdeleguser to the Windows Authorization Access Groups object. To do this:

a. Open the **Active Directory Users and Computers** panel.
b. Select the **Builtin** object under the domain.
c. Locate the **Windows Authorization Access Groups** object.
d. Right click and select **Properties**. Select the **Members** tab.
e. Click **Add** and add the delegation user (in our example, `tfimdeleguser`) as a member.

11. Grant the delegation user (tfimdeleguser) the **Act as part of the operating system** privilege.

    The actual process that must run as a Windows service depends on the WebSphere environment:

    - The service name in a *standalone* environment is the WebSphere Application Server running the Tivoli Federated Identity Manager runtime
    - The service name in a *cluster* environment is the WebSphere Application server running the WebSphere node agent for the Tivoli Federated Identity Manager runtime.

    Note: For a cluster environment, this step must be repeated on all machines hosting a node member of WebSphere cluster running the Tivoli Federated Identity Manager runtime.
    To do this:

    a. Access the menu appropriate for your deployment:
       - On the domain controller, select **Start** > **Programs** > **Administrative Tools** > **Domain Security Policy**.
       - On a non-domain controller computer, select **Start** > **Programs** > **Administrative Tools** > **Local Security Policy**.
    b. Expand Local Policies.
    c. Select **User Rights Assignment** > **Act as part of the operating system**.
    d. Right-click and select **Properties**.
    e. Click the **Define these policy settings** check box.
    f. Click **Add user or group** to add the delegation user (tfimdeleguser) to the list of users authorized to act as part of the operating system.
    g. Click **OK**.

12. Grant the delegation user (tfimdeleguser) the necessary privileges:
    - When the Tivoli Federated Identity Manager application is running on a member of the domain, grant the user the permission **Log on as a service privilege** on the local machine.
    - When the Tivoli Federated Identity Manager application is running on the domain controller, grant the user the permission **Log on as a service privilege** on the domain controller

    a. Return to the Security Policy menu opened in the previous step.
    b. Select **User Rights Assignment** > **Log on as service**.
    c. Right-click and select **Properties**.
    d. Click the **Define these policy settings** check box.
    e. Click **Add user or group** to add the delegation user (tfimdeleguser) to the list of users authorized to act as part of the operating system.
    f. Click **OK**.

13. Enable the WebSphere process that runs the Tivoli Federated Identity Manager application to run as a Windows service.

    Use the **wasservice** command. Default location:

```
C:\Program Files\IBM\WebSphere\AppServer\bin
```

Example command:

```
C:\Program Files\IBM\WebSphere\AppServer\bin>wasservice -add ndagentwinser
-servername nodeagent
-profilePath "C:\Program Files\IBM\WebSphere\AppServer\profiles\Custom01"
-wasHome "C:\Program Files\IBM\WebSphere\AppServer"
-logfile "c:\Program Files\IBM\WebSphere\AppServer\profiles\
   Custom01\logs\ws_startserver.log"
-logRoot "c:\Program Files\IBM\WebSphere\AppServer\profiles\
   Custom01\logs\nodeagent"
-restart true
```

Example output from the command:

```
Adding Service: ndagentwinser
  Config Root:
  C:\Program Files\IBM\WebSphere\AppServer\profiles\Custom01\config
  Server Name: nodeagent
  Profile Path: C:\Program Files\IBM\WebSphere\AppServer\profiles\Custom01
  Was Home: C:\Program Files\IBM\WebSphere\AppServer\
  Start Args:
  Restart: 1
IBM WebSphere Application Server V6.1
   - ndagentwinser service successfully added
```

To obtain a usage message for the **wasservice** command, enter:

```
> WASService.exe
```

without any arguments.

14. If running in a cluster environment, modify the WebSphere service from the previous step to start as the delegation user (tfimdeleguser)
    a. Open the **Services** control panel and locate either the service for the Tivoli Federated Identity Manager runtime or the Tivoli Federated Identity Manager runtime node agent for a cluster environment.
    b. Select the **LogOn** tab.
    c. Specify the delegation user `tfimdeleguser`.
    d. Specify the password for the delegation user.
    e. Click **OK**.
15. Restart the WebSphere nodeagent.

    This step is required to ensure that the Websphere node manager start the managed nodes under the new identity.
    a. Log on to the WebSphere console.
    b. Select **Servers** > **Application servers** for a standalone environment or **Servers** > **Clusters** for a cluster environment.
    c. Select the check box for the server or cluster to be restarted and press the **Stop** button for a standalone environment or the **Ripplestart** button for a cluster environment.
    d. In a standalone environment, after the server has been stopped, select the check box for the server or cluster to be restarted and press the **Start** button.

What to do next

Further information:

- Microsoft configuration principles:

  http://technet2.microsoft.com/windowsserver/en/library/c312ba01-318f-46ca-990e-a597f3c294eb1033.mspx?mfr=true
- Configuration instructions:

http://technet2.microsoft.com/windowsserver/en/library/e5d4cdbd-f071-4a1a-b24e-92713f7fafc11033.mspx?mfr=true

- IBM instructions for configuring WebSphere to run as an account other than **Local System**.

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.base.doc/info/aes/ae/tsec_actwindows.html

Chapter 38. Tivoli Federated Identity Manager configuration for a Kerberos junction scenario

Before you being configuring Kerberos delegation, ensure that you have created a domain, as described in Chapter 3, "Domain configuration," on page 23.

Configuration steps:

1. "Planning configuration of the trust chain"
2. Completing the "Worksheet for trust chain configuration" on page 517
3. "Creating a Kerberos constrained delegation module instance" on page 519
4. "Creating a trust chain for Kerberos constrained delegation" on page 519

Planning configuration of the trust chain

Plan the configuration in deploying a trust chain for Kerberos constrained delegation.

To deploy a trust chain for Kerberos constrained delegation, you must complete two tasks:

1. Create an instance of a Kerberos constrained delegation trust service module.
2. Create a trust chain for Kerberos constrained delegation.

Tivoli Federated Identity Manager provides configuration wizards for each task. The wizards prompt you to supply values for the required configuration properties.

Kerberos delegation module instance

The default set of Tivoli Federated Identity Manager trust modules does not include an instance of the Kerberos constrained delegation module type. You must create the instance.

Although it is possible for you to create more than one instance, you should create only one instance for each Tivoli Federated Identity Manager domain. This instance can be used in any module chain that is required.

The reason for the restriction to only one instance is that Kerberos constrained delegation module loads a native DLL (Windows dynamically loaded library) that is shared by all instances of the module. All instances share the same configuration parameters.

When more than one module instance is created, only the *last* module to be initialized determines the size of the user cache created in the native code. To prevent confusion, the best practice is to create only one module instance.

Module Type

This required property is requested on the Module Type panel. The module type to use is:

```
com.tivoli.am.fim.trustserver.sts.modules.KerberosDelegationSTSModule
```

Module Instance name
This required property is requested on the Module Instance Name panel. Supply a string of your choosing. For example:

```
MyKerberosDelegationInstance
```

Module Instance Description
This optional property is requested on the Module Instance Name panel. You can enter a string that describes the instance.

Maximum size of the user credential cache

This required property is requested on the Kerberos Delegation Module Configuration panel. This number determines the number of impersonation handles and user credentials cached in the DLL loaded by the module.

The caching is done to improve performance. Set this number to the approximate number of expected concurrent end users of the service for high-volume transactions.

The default setting is 100.

Note: The higher the number, the more memory that will be consumed by Tivoli Federated Identity Manager runtime application.

Kerberos delegation trust chain

Chain Mapping Name

This required property is requested on the Chain Mapping Identification panel. You can specify any name for the chain. For example:

```
ivcred_to_kerberos
```

Chain Description
This optional property is requested on the Chain Mapping Identification panel. The description can be any string.

Create a Dynamic Chain
This property is requested on the Chain Mapping Identification panel. This option is not used with Kerberos delegation trust chains. Clear this option.

Request Type
This required property is requested on the Chain Mapping Lookup panel. Select **Issue Oasis URI**.

Lookup Type
Select the radio button **Use Traditional WS-Trust Elements (AppliesTo, Issuer, and Token Type)**.

(AppliesTo) Address
This required property is requested on the Chain Mapping Lookup panel. Enter an **Address** that corresponds to the **applies-to** property in the [tfimsso:*jct_name*] stanza in the WebSEAL configuration file. For example:

```
http://websealhost.example.com/kerbjct
```

(AppliesTo) Service Name
This required property is requested on the Chain Mapping Lookup panel.

This property has two fields.

For the first field, either set this value to asterisk (*) to match all service names, or set it to value of service-name property in the [tfimsso:*jct name*] stanza in the WebSEAL configuration file.

For the second field, always set this value to asterisk (*).

(AppliesTo) Port Type
This property is requested on the Chain Mapping Lookup panel.

This property takes two fields.

Leave both fields blank.

(Issuer) Address
This required property is requested on the Chain Mapping Lookup panel. In the **Address** field, enter:

```
amwebrte-sts-client
```

(Issuer) Service Name
This optional property is requested on the Chain Mapping Lookup panel. Leave this field blank.

(Issuer) Port Type
This optional property is requested on the Chain Mapping Lookup panel. Leave this field blank.

Token Type
This required property is requested on the Chain Mapping Lookup panel. Select **Kerberos GSS V5**.

Initialize the chain upon startup of runtime
This required property is requested on the Chain Identification panel. Do *not* select this option.

Module Instances and modes
These required properties are requested on the Chain Assembly panel.

The Chain Assembly panel prompts you to enter values for the Module Instances in the chain. For each module instance, you must select a mode. You will then click a button to add the instance-mode pair to the chain.

For Kerberos constrained delegation, you want to configure a specific sequence of trust service modules:

1. The first Module Instance is **Default IVCred Token**. Choose a mode of **validate**.
2. The second Module Instance is the Kerberos delegation module instance that you created, as named in the **Module Instance Name** property within the module instance wizard. For example:

   ```
   MyKerberosDelegationInstance
   ```

   Select the **issue** mode.

Note: The wizard will warn you that your chain does not contain a module in **map** mode. For a Kerberos constrained delegation, the map mode is not required.

You can add a map mode if your deployment requires it. A map module would be needed if the Tivoli Access Manager user name needs to be mapped to a different user name in the Active Directory registry.

In a typical deployment, this mapping is not required. For example, in many deployments, Tivoli Access Manager will be installed to use the Active Directory registry. In these cases, there is only one identity for each user.

Enable signature validation
This property is requested on the Access Manager Credential (IVCred) Module Configuration panel. Do *not* select this option.

Default target Service Principal Name
This property is requested on the Kerberos Delegation module configuration panel, as Partner property.

In a typical deployment, you can leave this value blank.

This value can be used for WS-Trust clients that do not send the target Service Principal Name (SPN) in the AppliesTo/ServiceName element of the RequestSecurityToken (RST). The clients would also not have a mapping rule to configure the target SPN as a security token service universal user (STSUU) context attribute.

Options for adding a Tivoli Access Manager username for Kerberos authentication
The options allow you to specify whether the module will auto-append a suffix to the user name in the STSUniversalUser. The options are useful when deploying the Kerberos delegation module with a Tivoli Access Manager WebSEAL deployment. Options include:

- Do not add a suffix to the username.

  This option leaves the user name unmodified.
- Add the machine DNS domain as a suffix to the username.

  This option auto-appends the DNS domain suffix for the Tivoli Federated Identity Manager runtime machine to the principal name in the STSUniversalUser before calling the Windows API to obtain a Kerberos ticket. The DNS domain is read from the Windows Registry Key:

  `SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Domain`

  This option optimizes the module behavior for use in Tivoli Access Manager configurations using Kerberos junctions. The addition of the DNS domain enables the Windows API to successfully match the user name against the user record in the Active Directory user registry.

  Note that the module auto-appends the DNS domain name when the STSUniversalUser principal name does *not* already contain the @ character. This means that if a mapping rule was used to append a suffix containing the @ character to the user principal name, or if the Tivoli Access Manager username contains the @ character, this setting has no effect.
- Add the configured suffix to the username

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

  This option allows the administrator to manually specify the suffix. This option is for special cases where the userPrincipalName attribute for the user does not match the DNS domain name of the Windows machine running the Tivoli Federated Identity Manager Runtime. This option has no effect when the principal name already contains an @ character.

  `The suffix to add if using a configured suffix`
  For example:

  `@mydomain.com`

Worksheet for trust chain configuration

Complete the worksheets before configuring the trust chain.

The properties on the worksheets are described in "Planning configuration of the trust chain" on page 513.

Kerberos module instance worksheet

The following tables correspond to the panels presented by the module instance creation wizard.

Table 131. Module identification panels properties

Property	Value
Module type	com.tivoli.am.fim.trustserver.sts.modules.KerberosDelegationSTSModule
Module Instance name	
Module Instance description	

Table 132. Kerberos Delegation Module Configuration panel property

Property	Your value
Maximum size of the user credential	Default: 100

Kerberos module trust chain worksheet

The trust chain wizard presents a series of configuration panels. The following tables correspond to each panel.

Table 133. Chain mapping identification properties

Property	Your value
Chain Mapping Name	
Chain Description	
Create a Dynamic Chain	*This option must be deselected*

Table 134. Chain Mapping Lookup properties

Property	Your value
Request Type	Issue Oasis URI
Lookup Type	Use Traditional WS-Trust Elements (AppliesTo, Issuer, and TokenType)
(AppliesTo) Address	
(AppliesTo) Service Name	*Two fields* Use asterisk (*) for each field
(AppliesTo) Port Type	*Two fields* Leave both fields blank

Table 134. Chain Mapping Lookup properties (continued)

Property	Your value
(Issuer) Address	
(Issuer) Service Name	*Two fields* Leave both fields blank
(Issuer) Port Type	*Two fields* Leave both fields blank
Token Type	Kerberos GSS V5

Table 135. Chain identification panel

Property	Your value
Initialize the chain upon startup of runtime	*Do not select this option*

Table 136. Chain assembly panel

Property	Your value
First module instance	Default IVCred Token
First module mode	validate
Second module instance	*The name of your Kerberos module instance*:
Second module mode	Issue

Table 137. Access Manager Credential Module Configuration property

Property	Your value
Enable signature validation	*Deselect this option*

Table 138. Kerberos delegation module (Issue mode) Configuration property

Property	Your value
Default target Service Principal Name	
Options for adding a Tivoli Access Manager username for Kerberos authentication Options: • Do not add a suffix to the username. • Add the machine DNS domain as a suffix to the username. • Add the configured suffix to the username **The suffix to add if using a configured suffix** For example: `@mydomain.com`	

Creating a Kerberos constrained delegation module instance

Learn how to create a module instance for a Kerberos constrained delegation.

About this task

A wizard guides you through the creation of the module instance. For information about each requested property, see "Planning configuration of the trust chain" on page 513.

You can also consult the "Worksheet for trust chain configuration" on page 517.

Procedure

1. Log on to the WebSphere console.
2. Click **Tivoli Federated Identity Manager** > **Configure Trust Service** > **Module Instances**. The Module Instances portlet opens.
3. Click **Create**. The Module Instance wizard starts, and the Module Type panel opens.
4. Select **com.tivoli.am.fim.trustserver.sts.modules.KerberosDelegationSTSModule**.
5. Click **Next**. The Module Instance Name panel opens.
6. Enter a value in the **Module Instance Name** field.

   For example:

   ```
   Kerberos Junction
   ```
7. Optionally, enter a description in the **Module Instance Description** field.
8. Click **Next**. The Kerberos Delegation Module Configuration panel opens.
9. Enter a value in the field **Maximum size of the user credential cache**.
10. Click **Finish**. The Module Instances panel opens. The Current Domain portlet also opens, and prompts you to load the new configuration changes.
11. Click the **Load configuration changes to Tivoli Federated Identity Manager runtime** button.
12. Continue with "Creating a trust chain for Kerberos constrained delegation."

Creating a trust chain for Kerberos constrained delegation

You must create a trust chain and configure its properties for Kerberos constrained delegation using the trust chain wizard.

Before you begin

The domain must contain an instance of a Kerberos constrained delegation trust module before you build the trust chain. If you have not already created an instance, do so now. See "Creating a Kerberos constrained delegation module instance."

About this task

To configure the trust chain correctly, you must ensure that the properties align with WebSEAL configuration properties. Before running the trust chain wizard, you should:

- Review the topic "Planning configuration of the trust chain" on page 513
- Complete the "Worksheet for trust chain configuration" on page 517

Procedure

1. Log on to the WebSphere console.
2. Click **Tivoli Federated Identity Manager** > **Configure Trust Service** > **Trust Service Chains**. The Trust Service Chains portlet opens.
3. Click **Create**. The configuration wizard opens.
4. Click **Next**. The Chain Mapping Identification panel opens.
5. Enter the requested values.
   a. Enter a name in the **Chain Mapping Name** field.
   b. Optionally enter a description in the **Description** field.
   c. Do *not* select the field **Create a dynamic chain**
   d. Click **Next**. The **Chain Mapping Lookup** panel opens.
6. Enter the requested values.
   a. Set **Request Type** to **Issue Oasis URI**.
      The corresponding value for Request Type URI is automatically entered by the wizard.
   b. Set **Lookup Type** to **Use Traditional WS-Trust Elements (AppliesTo, Issuer, and TokenType)**.
   c. Enter values in the **AppliesTo** section.
      - Enter an **Address**.
        For example:
        `http://websealhost.example.com/krbjct`
      - Enter the **Service Name**.
        For example, set both fields to the asterisk character (*).
      - Leave the **Port Type** fields blank.

      For help, see "Planning configuration of the trust chain" on page 513.
   d. Enter values in the **Issuer** section.
      - In the **Address** field, enter:
        `amwebrte-sts-client`
      - Leave the **Service Name** field and **Port Type** field blank.
   e. For **Token Type**, select **Kerberos GSS V5**.
   f. Click **Next**.
      The Chain Identification panel is displayed.
7. Do *not* select **Initialize the chain upon startup of runtime**. Click **Next**.
   The **Chain Assembly** panel opens.
8. Build the trust chain:
   a. For Module Instance, select **Default IVCred Token**.
   b. For Mode, select **validate**.
   c. Click **Add selected module to chain**.
   d. For Module Instance, select the Module Instance Name you specified in "Creating a Kerberos constrained delegation module instance" on page 519. For example,
      `Kerberos Junction`
   e. For Mode, select **issue**.
   f. Click **Add selected module to chain**.
9. Click **Next**.

Note: You will see a warning stating that your chain lacks a module in map mode. You can ignore this warning. For more information, see "Planning configuration of the trust chain" on page 513.

The Access Manager Credential (IVCred) Module Configuration panel is displayed.

10. Do *not* select **Enable signature validation**. Click **Next**.

    The Kerberos Delegation Module Configuration panel opens.

11. If necessary, specify the Default target Service Principal Name or change the options for adding a suffix to the Tivoli Access Manager user name for Kerberos Authentication.

    Note: In most cases, you can leave this field blank and leave the default selection for the options. See "Planning configuration of the trust chain" on page 513.

12. Click **Next**. The Summary panel opens.
13. Click **Finish**.
14. In the Current Domain portlet, click **Load configuration changes to the Tivoli Federated Identity Manager runtime**.

Results

The trust chain configuration is now complete.

Tivoli Federated Identity Manager configuration notes

Configuring a Kerberos junction scenario might require you to verify some configuration settings. This section provides notes on what you need to verify.

Verify Tivoli Federated Identity Manager trust chain configuration

Verify that the WebSphere Deployment manager can communicate with the WebSphere Application Server that hosts Tivoli Federated Identity Manager.

To do this, access the URL:

```
http://<IHS_server>/TrustServerWST13/RequestSecurityToken
```

You will see a template response similar to the following:

```
RequestSecurityToken ... Hi there this is an AXIS service!
 Perhaps there will be a form for invoking the service here...
```

Verify WebSphere module mappings

Ensure that the WebSphere Application Server module mappings and virtual host mappings were propagated. To do this, access the URL:

```
http://<IHS_server>/Info/InfoService
```

You will see a template response similar to the following:

```
Hi there this is a Web service!
```

High availability in a cluster configuration

Multiple WAS servers will be deployed in a WAS cluster for high-availability. The individual WAS nodes in the cluster will receive their configuration instructions from a deployment manager.

Most administration tasks will be performed by communicating to the deployment manager. However, all of the protocol flows necessary to service requests to the TFIM trust service are served by individual WAS nodes. Failure of the deployment manager does not impact those protocol flows. Failure of the WAS nodes, however, will impact the protocol flow.

Chapter 39. WebSEAL configuration

You must install and configure the Tivoli Access Manager policy server before you install WebSEAL.


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.

These instructions assume that you have successfully installed and configured the policy server.

Complete a standard installation of WebSEAL. The exact steps to take depend upon your deployment environment. See the *IBM Tivoli Access Manager Installation Guide* for instructions.

Task overview:

1. "Verifying a WebSEAL installation"
2. "Planning WebSEAL Kerberos junction configuration" on page 524
3. Completing a "Kerberos junction configuration worksheet" on page 528
4. "Configuring a WebSEAL Kerberos junction" on page 529

Verifying a WebSEAL installation

This topic shows you how to verify that the basic WebSEAL server configuration is correct, so that you extend the configuration to support Kerberos junctions.

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.

These instructions assume that you have installed and configured IBM Tivoli Access Manager for e-business. The instructions also assume that you have successfully installed a WebSEAL server.

About this task

To verify the basic configuration, create a regular WebSEAL junction and verify that Tivoli Access Manager correctly prompts for a user login.

Procedure

1. Obtain the WebSEAL server name.

   The server name is based on the host name. For example, with a host name of `websealhost`:

   ```
   pdadmin sec_master> server list
     default-webseald-websealhost
   ```

2. Create a simple junction.

   For example, when the protected server is `mydataserver`, the following command creates a junction at `/jct`:

   ```
   pdadmin sec_master> server task default-webseald-websealhost
    create -t tcp -h mydataserver/jct
   ```

3. Obtain the list of the /WebSEAL object.

   This value is needed in order to correctly attach an access control list (ACL):

   ```
   pdadmin sec_master> object list /WebSEAL
     /WebSEAL/websealhost-default
   ```

4. Attach an ACL to the new junction.

   The ACL is used to control the actions that can be taken by specified users within the Tivoli Access Manager protected object space. This step assumes the existence of an ACL named `testacl`.

   ```
   pdadmin sec_master> acl attach /WebSEAL/websealhost-default/testacl
   ```

5. To confirm that the junction and ACL are configured correctly, complete the following steps:

   a. Place a test file under the documentRoot on the protected Web server.

      For example, in the documentRoot for `mydataserver`, create a test directory and add an index.html that displays some content. For example, under the junction point, add the file:

      ```
      /testdir/index.html
      ```

   b. Access the protected content:

      ```
      https://websealhost.example.com/jct/testdir/index.html
      ```

   c. WebSEAL prompts you to log in. Log in with a valid Tivoli Access Manager user identity and password.

      When successful, you can view the contents of `testdir/index.html`.

Planning WebSEAL Kerberos junction configuration

Before you can configure WebSEAL for Kerberos junctions, you must determine the values required by your deployment for each property.

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.

WebSEAL configuration properties are specified in the WebSEAL configuration file. The default configuration file is webseald-default.conf. For example, on UNIX or Linux systems:

```
/opt/pdweb/etc/webseald-default.conf
```

The configuration file contains properties that support the deployment of Kerberos junctions.

The properties are grouped into two stanzas:

```
[tfimsso:jct-id]
[tfim-cluster:cluster]
```

In some cases, the introduction of a Kerberos single signon for junctioned servers can impact performance. Each Kerberos token is valid only for one Kerberos authentication.

WebSEAL must request a new Kerberos token for each separate transaction. Performance can also be impacted by the communication channel, which requires WebSEAL to obtain tokens through a SOAP request to Tivoli Federated Identity Manager.

[tfimsso:*jct_id*] stanza

[tfimsso:*jct_id*] stanza

Use the [tfimsso:<jct-id>] stanza to specify configuration options for using Kerberos single signon. This stanza contains the Tivoli Federated Identity Manager single sign-on configuration information for a single junction.

- For standard junctions, the stanza name must be qualified with the name of the junction point, including the leading forward slash. For example:

  ```
  [tfimsso:/kerbjct]
  ```

- For virtual host junctions, the stanza name must be qualified with the virtual host label, for example:

  ```
  [tfimsso:www.example.com]
  ```

always-send-tokens

Boolean property. This property can be used to optimize performance when the back-end (junctioned) server is capable of maintaining session state. In this case, you can specify whether WebSEAL should send a Kerberos token for every HTTP request, or if WebSEAL should wait for a 401 response before requesting the token.

A 401 response means that authorization is required. When session state is maintained, it not necessary to authorize prior to each request. To limit the retrieval of Kerberos tokens to only those times when authorization is required, set

```
always-send-tokens = false
```

When the backend server cannot maintain session state, and a security token should be sent for every HTTP request, set:

```
always-send-tokens = true
```

applies-to

This property specifies the search criteria to use when locating the correct security token service module within Tivoli Federated Identity Manager.

The value is typically a path consisting of the format:

```
http://webseal_server_host/junction_name
```

For example:

```
http://websealhost.example.com/kerbjct
```

service-name

This important property is used for two purposes:

1. To specify the service principal name that is used when generating a Kerberos token.

This value is used by Tivoli Federated Identity Manager when it searches for a matching trust chain. The Tivoli Federated Identity Manager chain configuration includes an Applies-to section that contains a Service Name property. The value of the WebSEAL service-name setting is compared against the Service Name property.

To ensure a successful match, `service-name` should match the Service Name property in the Tivoli Federated Identity Manager configuration.

Note: One way to ensure a successful match is to use, within the Tivoli Federated Identity Manager configuration, a wildcard character such as asterisk (*).

2. To specify the service principal name of the delegating user when creating the Kerberos token. The service principal name (SPN) is set on the Microsoft Windows system.

   To determine the SPN, go to the Windows server, and use the **setspn** command. For example:

   `setspn -L user_name`

   The junctioned Web server runs with the identity *user_name*. For example, iisuser.

The syntax for this property is:

`service-name=service_principal_name`

The format is:

`HTTP/IIS_server_name.domain_name`

For examples:`service-name = HTTP/B16INTEL3.tamad.com`

renewal-window

The length of time, in seconds, by which the expiry time of a security token is reduced. This entry is used to accommodate differences between system times, and to allow for transmission times for the security tokens.

```
renewal-window = 15
```

tfim-cluster-name

The name of the WebSphere cluster where the Tivoli Federated Identity Manager service is deployed. This value should be matched by another stanza entry [tfim-cluster:<*cluster*>], where *cluster* matches **tfim-cluster-name**.

For example:

```
tfim-cluster-name = STSCluster2
```

token-collection-size

To optimize performance, WebSEAL can request multiple Kerberos tokens from Tivoli Federated Identity Manager within one SOAP request. This is done through use of the WS-Trust Web service specification. The tokens are cached in the user's session and used on subsequent requests.

WebSEAL requests additional tokens from Tivoli Federated Identity Manager only after all of the cached tokens have been used or have expired.
You can specify the number of tokens to retrieve from Tivoli Federated Identity Manager. When this number is increased, the number of requests to Tivoli Federated Identity Manager is decreased, but the size of (and processing time for) each request is increased. The Kerberos tokens can be quite large.

If you specify a large value for this property, you can significantly increase the session size and memory usage for WebSEAL.
The default value is 10:

```
token-collection-size = 10
```

token-type
The only supported token type is `kerberos`. This is the default value. Use this value. Do not change it.

tfim-cluster cluster stanza

[tfim-cluster:*cluster***]**

This value defines the name of the WebSphere cluster for the Tivoli Federated Identity Manager service. The *cluster* name for this stanza must match the **tfim-cluster-name** option in a **[tfimsso:*jct-id*]** stanza.

server Specifies the priority level and URL for a single Tivoli Federated Identity Manager server that is a member of the cluster identified for this stanza.

You can have multiple **server** entries in the stanza. This enables you to specify multiple server entries for failover and load balancing purposes between WebSEAL and the WebSphere Application Server proxy.

When the Tivoli Federated Identity Manager cluster is configured, WebSEAL checks the status of the Tivoli Federated Identity Manager proxy Web server once every minute.

When you have multiple servers, you can use the priority level to specify the order in which the servers are accessed to perform processing. The priority level is an integer in the range [0-9].

When you have only one server, you can omit the priority level. When the priority level is not specified, the level is assumed to be 9 (highest).

Syntax:

```
server = [0-9],server_URL
```

Example:

```
9,http://mydataserver.example.com/TrustServerWST13/services
/RequestSecurityToken
```

handle-pool-size
Specifies the maximum number of cached handles to use when communicating with Tivoli Federated Identity Manager.

Default: 10

handle-idle-timeout
The length of time, in seconds, before an idle handle is removed from the handle pool cache.

Default: 240 seconds

timeout
The length of time, in seconds, to wait for a response from Tivoli Federated Identity Manager.

Default: 240 seconds

ssl-keyfile
The name of the key database file which houses the client certificate to be used.

This SSL entries, and the ones following, are optional and are only required when:

- At least one server entry indicates that SSL (HTTPS) is to be used.
- A certificate is required other than that which is used by this server when communicating with the policy server.

Note: This value, and the following SSL entry values, must be shared for all server variables that use HTTPS. When deploying into a WebSphere cluster, the values must be the same for each server in the cluster that uses HTTPS.

ssl-keyfile-stash
The name of the password stash file for the key database file.

ssl-keyfile-label
The label of the client certificate within the key database.

ssl-valid-server-dn
This configuration entry specifies the DN of the server (obtained from the server SSL certificate) which will be accepted. When no entry is configured, all DN's will be considered to be valid. Multiple DN's can be specified by including multiple configuration entries of this name.

ssl-fips-enabled
This entry controls whether FIPS communication is enabled with Tivoli Federated Identity Manager or not. When no configuration entry is present, the global FIPS setting, as determined by the TAM policy server, will take effect.

Note: For a complete description of each stanza property, see the *IBM Tivoli Access Manager WebSEAL Administration Guide*. See also the comments within the WebSEAL configuration file.

Kerberos junction configuration worksheet

Use this worksheet to assemble the values that you must add to the WebSEAL configuration file.

Table 139. tfimsso and tfim-cluster stanza properties

Property	Your value
[tfimsso:*junction_id*]	
always-send-tokens	default: false
applies-to	
service-name	
renewal-window	default: 15
tfim-cluster-name	
token-collection-size	default: 10
token-type	kerberos
[tfim-cluster:*cluster*]	
server	

Table 139. tfimsso and tfim-cluster stanza properties (continued)

Property	Your value
handle-pool-size	default: 10
handle-idle-timeout	default: 240
timeout	default: 240
ssl-keyfile	
ssl-keyfile-stash	
ssl-keyfile-label	
ssl-valid-server-dn	
ssl-fips-enabled	

Configuration tips:

- Ensure that the **service-name** property matches the Tivoli Federated Identity Manager trust chain configuration.
- Ensure that the **tfim-cluster-name** property matches the *cluster* property in the stanza [tfim-cluster:*cluster*].
- Ensure that the *cluster* property in [tfim-cluster:*cluster*] matches the name of the WebSphere cluster.

Configuring a WebSEAL Kerberos junction

Configure the WebSEAL Kerberos junction by editing the WebSEAL configuration file, and using the pdadmin command to create the junction, and attach access control lists (ACLs).

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

Configuration of a WebSEAL Kerberos junction consists of two tasks:

- Edit the WebSEAL configuration file.

  You must specify properties in the WebSEAL configuration file to support the specific junctions for Kerberos single-signon before you can use the pdadmin command to create the junction.
- Use the pdadmin command to create the junction and attach the necessary access control lists (ACLs).

  To create a standard junction that is enabled for Kerberos single signon, use the junction create command (server task create) with option -Y. The -Y option specifies that SPNEGO/Kerberos single sign-on is required for the junction.

  To create a virtual host junction that is enabled for Kerberos single signon, use the virtualhost create command (server task create) with the -Y option.

WebSEAL supports many options for creating junctions. You can combine the -Y option with other options, as required for your deployment. For complete information on WebSEAL junction options, see the *IBM Tivoli Access Manager WebSEAL Administration Guide*.

Procedure

1. Use a text editor to edit the WebSEAL configuration file.

   Use the values that you assembled in the worksheet for Kerberos junction support.

   For more information, see "Planning WebSEAL Kerberos junction configuration" on page 524
2. Use the pdadmin command to create the Kerberos junction and attach the necessary ACLs.

   You can create either regular Kerberos junctions or virtual host Kerberos junctions.

   Note:
   - The name of the junction must match the *jct_id* value for the [tfimsso:*jct_id*] stanza in the WebSEAL configuration file.
   - Ensure that you have configured the WebSEAL configuration file for the type of junction that you want to use. If you have not edited the WebSEAL configuration file, the administration command will not succeed, and will return an error message.

   Regular Kerberos junctions

   a. Create the junction:

   ```
   pdadmin sec_master>  server task default-webseald-websealhost
     create -t tcp -h mydataserver.example.com -Y /kerbjct
   ```

   The host mydataserver.example.com is the IIS backend server.

   b. Attach the ACL:

   ```
   pdadmin sec_master> acl attach /WebSEAL/websealhost-default/kerbjct testacl
   ```

   Virtual host Kerberos junctions

   a. Create the junction:

   ```
   pdadmin sec_master> server task default-webseald-websealhost virtualhost
     create -t tcp -h mydataserver.example.com -v website.example.com
     -Y kerbvirtjct
   ```

   b. Attach the ACL:

   ```
   pdadmin sec_master> acl attach /WebSEAL/websealhost-default/kerbvirtjct
     testacl
   ```

Results

Error messages are logged in the WebSEAL configuration log file. For example, on UNIX or Linux:

```
/opt/pdweb/log/msg__webseald-default.log
```

WebSEAL configuration notes

Use the following WebSEAL configuration notes for your communication between WebSEAL and the client.

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.

Configuration notes for communication between WebSEAL and the client

- High availability for the WebSEAL server is typically done by placing a load-balancer in front of the WebSEAL server. See the IBM Developer Works article *Load Balancers for Tivoli Access Manager*:

  http://www-128.ibm.com/developerworks/tivoli/library/t-tlb/index.html
- Security of the communication path between the client and WebSEAL is typically provided by purchasing an SSL server certificate for the WebSEAL server.
- Clients may authenticate to WebSEAL through any supported method.
- No change to these standard configurations will be necessary for Kerberos junction support.

Configuration notes for communication between WebSEAL and the junction

- High availability for the junctioned server is typically done by configuring multiple junction servers for the junction point. See the IBM Developer Works article *Load Balancers for Tivoli Access Manager*:

  http://www-128.ibm.com/developerworks/tivoli/library/t-tlb/index.html
- Security of the communication path between WebSEAL and the junction is typically guaranteed by mutually authenticated SSL certificates.
- No change to these standard configurations is required for Kerberos junction support.

Time synchronization between WebSphere and WebSEAL

Verify that time settings are synchronized between the system that hosts the WebSphere Application Server that runs Tivoli Federated Identity Manager and the system that hosts Tivoli Access Manager WebSEAL.

To view the settings:

1. On the WebSphere system, select **Default Domain Security Settings** > **Account Policies** > **Kerberos Policy**.
2. Review the Maximum tolerance for computer clock synchronization.

When the time difference is large between the WebSphere Application Server and the WebSEAL server, the security tokens generated by Tivoli Federated Identity Manager might expire before they can be used.

Configuration error messages

The following error messages are displayed when one of the following conditions are true:

- The service-name property does not match the Tivoli Federated Identity Manager trust chain configuration.
- WebSEAL retrieves tokens from Tivoli Federated Identity Manager, but the tokens have expired. This can happen, for example, when the time settings on each of the servers are not synchronized.

- The browser returns an error. For example:

```
Server Error
Access Manager WebSEAL could not complete your request due to an
unexpected error.
Diagnostic Information
Method: GET
URL: /kjct/index.html
Error Code: 0x38cf027c
Error Text: DPWWA0636E No TFIM single sign-on tokens were available.
```

- The WebSEAL log contains errors. For example (some lines split for formatting purposes):

```
DPWWA2852E   An error occurred when attempting to communicate with the SOAP
  server URL
http://d06win13.testlab.example.com/TrustServerWST13/services/
  RequestSecurityToken: +JNI:
Error running InitializeSecurityContext for HTTP/d02jlnx.testlab.example.com:
-2146893042 (No credentials are available in the security package).
File h:\fim620\src\kerberoswin32\KerbUserState.cpp,
 line 641  (error code: 71/0x47).
2008-03-04-13:08:10.080-06:00I----- 0x38CF027C
webseald ERROR wwa sso ThirdPartyJunction.cpp 4124 0x00000070
DPWWA0636E   No TFIM single sign-on tokens were available.
```

Debugging a Kerberos junction

To debug a Kerberos junction deployment, turn on tracing for Tivoli Federated Identity Manager and Tivoli Access Manager. A relevant trace point for Tivoli Access Manager and WebSEAL is pdweb.sso.tfim.

For example, in a Linux or UNIX environment:

```
pdadmin> server task default-webseald-c1sun1 trace set pdweb.sso.tfim 9
  file path=/var/pdweb/log/debug.log
```

Set the trace level to 0 to turn off tracing.

Configuring WebSEAL to manage cookies

By default, WebSEAL does not delete cookies upon logout. If you plan to configure WebSEAL to manage cookies, the list of managed cookies should not include the WebSphere session cookie.

Chapter 40. SSL configuration task for a Kerberos junctions deployment

For optimal security, configure SSL communication between servers in a Kerberos junction deployment.

This topic provides an overview of the steps to configure a WebSphere cluster environment to use SSL to communicate between WebSEAL, IBM HTTP Server (IHS), WebSphere Application Server Plug-in, WebSphere Application Server and Tivoli Federated Identity Manager. These steps do not address SSL communication between the client and WebSEAL or to the back-end Web server. No changes to these standard SSL configurations are necessary for Kerberos junction support.

Tip: Consider deploying a working configuration without SSL prior to adding SSL.

For each component, create a public/private key pair, and extract the public key to a known location.

On the WebSEAL server:

1. Copy the IHS public key to the WebSEAL system.
2. Use the **ikeyman** utility to add the IHS public key. When there is more then one IHS proxy in the environment, complete this task for each IHS server.
3. Configure appropriate values for the following `[tfim-cluster:cluster]` variables: server, ssl-keyfile, ssl-keyfile-stash. Optionally, configure the ssl-valid-server-dn variable if applicable.

   For more information, see "Planning WebSEAL Kerberos junction configuration" on page 524.
4. Restart WebSEAL to activate the changes made to the WebSEAL configuration file.

On the IBM HTTP Server:

1. Copy the WebSEAL public key to the IHS system.
2. Use the **ikeyman** utility on IHS to add the WebSEAL public key.
3. Copy the WebSphere public key from the WebSphere Deployment Manager (dmgr) system to the IHS system.
4. Use the **ikeyman** utility on IHS to add the WebSphere public key.
5. Update the `httpd.conf` file to configure or add a virtual host to support SSL connections.
6. Restart IHS to activate the changes.
7. When your deployment includes multiple IHS proxies, repeat the above steps for each IHS proxy.

On the WebSphere plug-in located on the IHS server:

1. Copy the WebSphere public key to the plug-in system.
2. Use the **ikeyman** utility for the plug-in to add the WebSphere public key.
3. Copy the WebSphere node public key from the WebSphere node to the plug-in server.
4. Use the **ikeyman** utility for the plug-in to add the WebSphere node public key.

5. When your deployment includes multiple plug-ins, repeat the above steps for each plug-in.

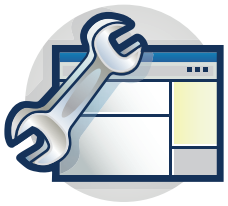
On the WebSphere Network Deployment Manager (dmgr):

1. Ensure that the public key for the plug-in is located in a file path that can be accessed through the WebSphere administration console.
2. Use the WebSphere console to add the public key for the plug-in to the CellDefaultTrustStore.
3. When your deployment includes multiple plug-ins, repeat the above steps for each plug-in.
4. Ensure that the public key for Node is located in a file path that can be accessed through the WebSphere administration console.
5. Use the WebSphere console to add the public key for the Node to the CellDefaultTrustStore.
6. When your deployment includes multiple nodes, repeat the above steps for each nodes.
7. Configure client authentication if appropriate for your deployment.

On the WebSphere Node:

1. Ensure that the public key for the Deployment Manager (dmgr) is located in a file path that can be accessed through the WebSphere administration console.
2. Use the WebSphere console to add the dmgr public key to the NodeDefaultTrustStore.
3. When your deployment includes multiple nodes, repeat the above steps for each nodes.

Part 6. Configuring User Self Care



The topics in the Configuration section provide a step-by-step guide to configuring User Self Care.

This section describes the deployment of User Self Care. First read the overview of the User Self Care feature:

Chapter 41, “Understanding User Self Care,” on page 537

Chapter 41. Understanding User Self Care

User Self Care provides a method by which users can be provisioned into business-to-consumer environments.

User Self Care accomplishes this provisioning by supplying a set of operations that users can use to create and administer their own accounts. The operations include:

- Creating an account
- Creating and updating attributes associated with the account
- Changing passwords
- Recovering forgotten user IDs and passwords
- Deleting accounts

User Self Care is based upon the Tivoli Federated Identity Manager secure token service (STS) technology.

Note: IBM deprecated the Tivoli Federated Identity Manager Security Token Service (STS) Client in this release.

If you use WebSphere 6.X, you can still use the Tivoli Federated Identity Manager Security STS client while Tivoli Federated Identity Manager supports WebSphere 6.X. When Tivoli Federated Identity Manager discontinues its support for WebSphere 6.X, use WebSphere Application Server version 7 Update 11 and later. See WS-Trust client API and WS-Trust Clients for details.

With the STS framework, administrators can plug their own token creation and consumption modules in. User Self Care uses the STS framework and the HTTP components of Tivoli Federated Identity Manager, but it is not used for token creation and consumption.

Users access User Self Care operations through an HTTP interface. Users interact with web pages that prompt for input, collect data, and provide feedback. User Self Care provides a small set of URLs that serve as endpoints for accessing operations.

You can customize User Self Care. STS modules plug-ins that are started sequentially in a chain implement business logic. To provide additional capability for each chain, you can replace individual modules or add new ones. You can modify or replace the HTML forms as necessary.

User Self Care uses the clustering, distribution, scaling, and configuration capabilities provided by WebSphere. User Self Care also uses the WebSphere Federated Repositories component for making registry adapters available to the operating environment. Administrators can add or replace registries.

User Self Care also integrates with Tivoli Access Manager WebSEAL. WebSEAL provides authentication and authorization for business-to-consumer transactions.

The figure shows the software pieces that comprise the User Self Care solution.

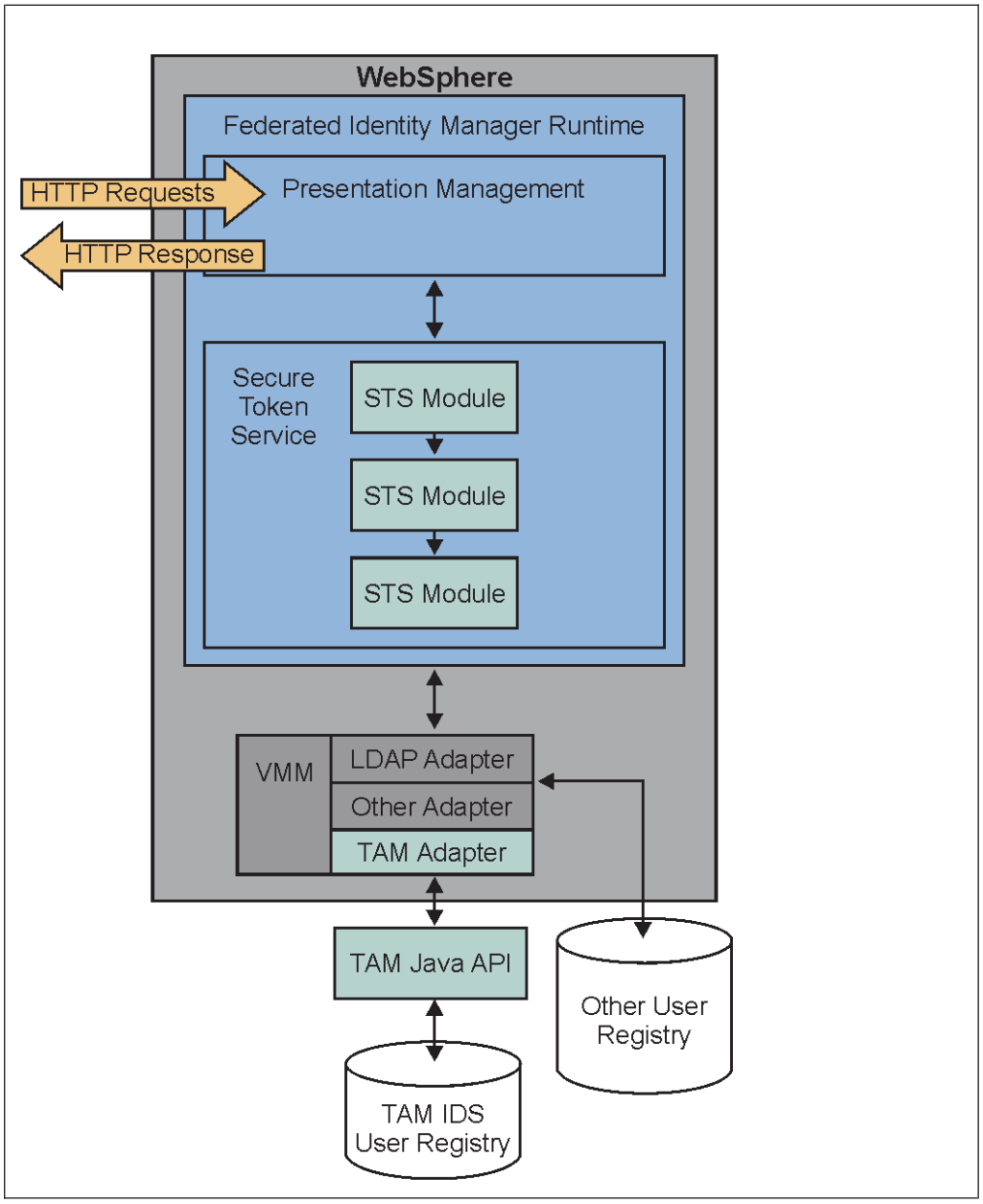


Figure 66. User Self Care solution

- WebSphere provides the framework for most of the software pieces.
- The Tivoli Federated Identity Manager run time provides two components that support User Self Care:

  User Self Care presentation management
  Provides a set of default pages. Users interact with these pages by requesting User Self Care URLs. The management framework supports customization and replacement of these pages. This support includes the ability to substitute (customize) macros on the pages.

  Secure token service (STS) trust chains
  Supports the building of dynamic chains of plug-in modules for performing business logic. User Self Care support includes a number of STS chains. Each chain maps to a User Self Care operation. You can extend the chains. You can replace or modify the component modules in

each chain. Validating user input and sending a confirmation email is an example of a User Self Care chain operation.

- The STS modules use the WebSphere Federated Repository to communicate with the user registry. When the target user registry is Tivoli Access Manager, User Self Care uses the product adapter to communicate to the Tivoli Access Manager registry through the Tivoli Access Manager Registry Direct Java API.

User Self Care works with various user registries. Each registry has a unique syntax for performing management operations. The WebSphere Federated Repositories component allows User Self Care to issue a management command, such as **user create**, using a consistent syntax. The Federated Repositories component then passes the request to the appropriate registry adapter, which translates the command into the registry-specific syntax.

Since WebSphere Federated Repositories provides a plug-in interface for adapters, you can add new registries without modifying the User Self Care.

Effectively customizing User Self Care

Deployments of User Self Care are generally customized for specific business needs. You can most effectively customize your deployment when you understand how the User Self Care pieces work together.

1. Understand the User Self Care technology.
   - User Self Care is based on a series of operations. See "Understanding User Self Care operations."
   - Users interact with User Self Care features through HTTP request and response exchanges. HTML pages as URL drive the exchanges The default HTML pages are templates for the information you want to exchange with your users. You can (and should) customize the HTML pages to reflect your business needs. For more information on default HTML pages, see "User Self Care URLs" on page 547.
   - Many Internet sites use Captcha (Completely Automated Public Turing test to tell Computers and Humans Apart) challenge-response tests to protect against machine-generated attacks. This technology is part of many User Self Care deployments. The User Self Care product provides a Captcha demonstration module. See "Captcha demonstration" on page 550
2. Deploy Tivoli Federated Identity Manager and configure User Self Care.

   This document provides configuration steps that you must do in a specific order. See Chapter 42, "Deploying User Self Care," on page 553.
3. Understand the methods for tuning distributed caches to optimize performance. See Chapter 43, "Tuning User Self Care," on page 617.

Understanding User Self Care operations

A User Self Care *operation* is the series of steps required for a user to accomplish a task.

An example of a task is when a user recovers a forgotten password. To perform this task, the user must take several steps:

1. Submit a web form with their user ID.
2. Submit a second form that asks them to answer their secret question and to provide a new password.
3. Click a link in an email that is sent to them.

The combination of steps comprises an *operation*.

Every user-initiated action performed as part of User Self Care is in the form of an HTTP request. Example requests are requesting a page, submitting a form, or clicking a link in an email. Every HTTP request has a corresponding HTTP response. Some example responses are providing the user with the input form or informing them that an email has been sent. Each User Self Care operation consists of one or more request-response exchanges.

In most cases, each request-response exchange is an atomic event. For example, when a user requests the Profile Management page, User Self Care finishes a discrete operation by returning the page. User Self Care does not retain any state or knowledge that the user has made the initial request. There are exceptions to this treatment of user state, which are described in the individual operation topics of this documentation.

This documentation groups the request-response exchanges based on their association with an operation. Each operation is associated with a particular secure token service trust chain. The STS trust chains do the bulk of the work in processing a User Self Care operation.

Note: IBM deprecated the Tivoli Federated Identity Manager Security Token Service (STS) Client in this release.

If you use WebSphere 6.X, you can still use the Tivoli Federated Identity Manager Security STS client while Tivoli Federated Identity Manager supports WebSphere 6.X. When Tivoli Federated Identity Manager discontinues its support for WebSphere 6.X, use WebSphere Application Server version 7 Update 11 and later. See WS-Trust client API and WS-Trust Clients for details.

A typical request-response exchange

The typical flow when a user submits a request to User Self Care is:

1. The user requests a User Self Care URL that specifies an HTML form.
2. The User Self Care presentation management component returns the appropriate HTML form.

   If the Captcha module is used, the Captcha STS chain is started to obtain the image shown to the user for validation.
3. The user supplies data for the form and submits the form.
4. The presentation management component sends the resulting HTTP request to the appropriate STS trust chain.
5. The User Self Care STS trust modules in the chain are started in a specified order, to perform tasks such as:
   - Validating data
   - Mapping attribute
   - Interacting with registries
   - Sending email
6. The STS modules return the process results to the presentation management component.
7. The presentation management component returns an HTTP response to the user. See the following list of typical responses:
   - Another form

- The same form with a message
- An error page
- An informational page

Depending upon the operation, the user task is either finished or requires another step. If another step is required , the preceding or similar sequence is repeated.

Operations and STS chains

Each user self care operation maps to a single STS chain. During the operation, the STS chain might be started multiple times. User Self Care determines what stage of the operation is being performed and controls behavior accordingly.

For example, Captcha validation might be performed when a user submits the initial enrollment form. However, it is not performed when the user clicks the link in the email. In both cases, the same STS chain is started, and the Captcha STS module is present at the start of the chain. In the second case, the Captcha module is not supposed to do anything, and passes the request to the next STS module in the chain.

You can use the administration console to view each trust chain. Trust chains correspond to one or more User Self Care operations. When you view the trust chains, you see the STS modules that accomplish the operation. You can then customize the modules and chains for your deployment.

Note: For information about how to customize User Self Care, see the Tivoli Federated Identity Manager Wiki:

http://www.ibm.com/developerworks/wikis/display/tivolifederatedidentitymanager/Home

User ID existence check operation

On the initial enrollment page, the user enters a user ID in a specified field. User Self Care provides an icon that the user can click to check if the ID exists in the registry.

The user ID existence operation is an exception to the rule of one STS chain per operation. This operation maps to the same STS trust chain as the enrollment operation. However, it is conceptually different and uses a different URL.

Operation task flow:

1. User enters their requested user ID in a form field.
2. User clicks the icon.
3. The Create Account STS Chain is started.
   - The registry is queried to determine whether the user ID exists.
   - The internal cache is also queried.

     The check of the internal cache is described in “Enrollment operation.”

Enrollment operation

The enrollment operation takes place in two request-response exchanges: when you obtain user information in preparation for sending a validation email; and when the user validates the operation by clicking a link in the email.

Initial enrollment request

Operation task flow:

1. User requests and receives an Enrollment Request form. User supplies data for the form fields with enrollment details such as:
   - User ID
   - E-mail address
   - Password
   - Choice of profile attributes, including the secret question attribute.
2. User submits the Enrollment Request form.
3. The Create Account STS Chain is started.
   - If any errors are encountered, they are returned to the user. The errors are shown as a message on the form that the user initially processed.
   - If no errors are encountered, an email is sent to the user for validation. User Self Care shows a page to the user, advising them of the email.
4. An entry is created in an internal cache that preserves the user enrollment information during the validation. This internal cache also preserves the user ID so that no other user can use it for enrollment. You can configure the time limits for how long data is retained in the internal cache.

Enrollment validation

The email that was sent during the initial enrollment request contains a link with a query string appended. The query string contains a key to the internal cache entry so that the data that the user initially submitted can be recovered and enrollment finished.

Task flow:

1. User clicks a link in the validation email.
2. The Create Account STS Chain is started.
3. If any errors are encountered, they are shown in a page that is sent to the user. If no errors are encountered, User Self Care:
   a. Creates an entry in the registry for the new user account.
   b. Removes the internal cache entry.
   c. Sends a success message to the user.

Password management operations

There are two password management operations: a user-initiated change of password, and a password change required by expiration of an existing password

User-initiated change password

Task flow:

1. The user requests the Change Password Form URL.
2. User Self Care provides the user with a form in which they enter their old password and a new password twice.
3. The user submits the form to the Change Password URL.
4. User Self Care starts the Change Password STS Chain.
   - If any errors are encountered, User Self Care sends the user on an informational page containing the errors.

- If no errors are encountered, the password is changed. User self care then sends a success page to the user.

Change password following password expiration

The task flow is the same as in the User-initiated change password topic, with the exceptions that:

- The user makes an initial request for a protected resource
- The point of contact server requires the user to change their password.

The initial request from the user is intercepted by the authenticating point of contact server, such as WebSEAL or WebSphere Application Server. The point of contact server handles the communications flow and must direct the user to User Self Care in order to change their password.

User self care provides deployment suggestions and enhancements for accomplishing this using WebSEAL as a point of contact server. For more information, see "Integrating User Self Care with WebSEAL" on page 609.

User self care can function as a callable component for a capability such as the Tivoli Access Manager Local Response Redirect feature. This feature redirects the user to the User Self Care handler to perform a change password operation. The user is then redirected back after the operation succeeds.

Profile management operations

You can use the Profile Management to manage extended information specific to their account.

Examples of such information are:

- Address
- Phone number
- Secret question

Initial Profile Management Request

Task flow:

1. User submits request for the Profile Management Form URL. This URL must be a protected resource.
2. The user identity is obtained from the authenticated context.
3. User Self Care starts the profile management STS Chain, and provides the user identity.
   - If errors are encountered, they are shown in an informational page that is sent to the user.
   - If no errors are encountered, the STS retrieves the attributes from the registry.
4. User Self Care presents the user with the Profile Management Form containing their existing attributes. The user can then update profile information, including their secret question.

Submit Profile Update

Task flow:

1. User modifies the wanted fields and submits the form.

2. The user identity is obtained from the authenticated context.
3. User Self Care starts the profile management STS Chain.
   - If errors are encountered, they are shown in an informational page that is sent to the user.
   - If no errors are encountered, the registry is updated. User self care sends a success page to the user.

Forgotten user ID operation

A forgotten ID can still be retrieved by following the steps in this procedure.

Operation task flow:

1. User clicks on the Forgotten ID URL. This URL must not be a protected resource.
2. The Forgotten ID form is returned to the user.
3. User enters their e-mail address.

   A custom solution can use a different registry attribute, such as a customer account number, for example. The default User Self Care form uses the e-mail address.
4. User submits the form.
5. User Self Care passed the form contents to the Forgotten ID STS Chain. The modules in this chain retrieve all the user IDs associated with the e-mail address from the registry, and then e-mail them to the user.
   - If errors are encountered, they are shown in an informational page sent to the user.
   - If no errors are encountered, User Self Care sends the Forgotten ID Acknowledgement informational page to the user. The page informs the user that the user IDs have been sent to their e-mail address.

Forgotten Password operation

The forgotten password operation takes place in several request-response exchanges.

Task flow:

1. The user requests the Forgotten Password URL. This URL must not be a protected resource.
2. User Self Care sends the Forgotten Password Form to the user.
3. The user enters their user ID and submits the form.
4. User Self Care passes the form contents to the Forgotten ID STS Chain to retrieve the secret question.
5. The STS module sends the Forgotten Password Secret Question Form to the user. The form has the secret question and a field in which the user must enter the answer. The form also provides two fields for capturing a new password.
6. The user edits and submits the Secret Question form.
7. User Self Care passes the form contents to the Forgotten ID STS Chain to perform Secret Question Validation.
   a. The STS tracks the number of failed attempts in an internal cache. If the number exceeds the configured limit, the STS sends an error to the user.
   b. The STS stores the password change request in an internal cache.
   c. The STS sends an e-mail to the user containing a link to the Forgotten Password Validation Form URL. The e-mail contains a link with a query

string appended. The query string contains a key to the internal cache entry. The key is used so that the data that the user submitted can be recovered and the password change finished.

8. The user requests the link in the e-mail.
9. User Self Care passes the request to the Forgotten ID STS Chain. The chain modules recover the data from the internal cache and attempt to change the password.
   - If errors occur, they are shown in an informational page sent to the user.
   - If no errors occur, User Self Care sends to the user the Forgotten Password Acknowledgement informational page. This page tells the user that the password has been changed.

Account deletion operation

Account deletion operation follows an operation task flow.

Operation task flow:

- The user requests the Account Deletion page. This page must be a protected resource.
- The user clicks a link on the page.
- The user identity is obtained from the authenticated context.
- The Account Deletion STS chain is started.
- The Account Deletion STS trust chain finishes the deletion of the user account.
- User Self Care returns the Account Deletion success informational page to the user.

Captcha operation

Captcha is not a separate User Self Care operation. Instead, the Captcha operation is implemented as a Captcha STS module.

You can place the Captcha STS module first in any of the secure token service trust chains used by User Self Care.

Note: IBM deprecated the Tivoli Federated Identity Manager Security Token Service (STS) Client in this release.

If you use WebSphere 6.X, you can still use the Tivoli Federated Identity Manager Security STS client while Tivoli Federated Identity Manager supports WebSphere 6.X. When Tivoli Federated Identity Manager discontinues its support for WebSphere 6.X, use WebSphere Application Server version 7 Update 11 and later. See WS-Trust client API and WS-Trust Clients for details.
When the Captcha module is present, Captcha validation is performed before execution of any other operations.

For more information, see “Captcha demonstration” on page 550.

Registry attributes operations

User Self Care does not provide the capability to modify user registry schema. You must modify your registry schema as required to create the registry attributes required for supporting profiles. You must also modify the schema to support the *secret question* attribute.

User Self Care provides an example function. User Self Care uses the LDAP attribute `businessCategory` to store the secret question profile attribute. The example implementation also uses the LDAP attribute `mobile` to store a mobile phone number for the user.

When you deploy User Self Care, you must create a schema that can contain the profile attributes you must provide for your users. When you have identified and defined these attributes, you must customize the HTML forms and STS modules to work with them.

In a full deployment, it is necessary to create a schema that can contain the profile attributes you must provide for your users. When these attributes are selected, you must customize the HTML forms and the STS modules in order to work with the new attributes.

For more information, see the Tivoli Federated Identity Manager Wiki:

http://www.ibm.com/developerworks/wikis%2Fdisplay%2Ftivolifederatedidentitymanager%2Fhome.

Secret question operation

The *secret question* is a secondary password and hint stored in the user registry as a user attribute. User Self Care treats the management of the secret question as another profile element.

User Self Care provides an example implementation of the secret question by using the LDAP attribute businessCategory to store the secret question profile. You can customize this implementation to best fit your business needs.

The following topics describe how the example implementation works.

Selection of secret question during enrollment

A User Self Care enrollment form provides a menu that permits a user to select one of the following questions:

- Maiden name of mother
- Town where you were born
- Name of first pet

The selection of one of these items populates a form field with a numeric value that corresponds to the index of the entry in the list. The name of this field on the HTML forms provided with user self care is `usc.form.profile.secret.question`.

A separate form field is used to specify the answer to the question in text. The name of this attribute on the HTML forms provided with User Self Care is `usc.form.profile.secret.question.answer`.

When the user submits the enrollment form, each of these parameters is passed to the Enrollment STS trust chain. The index and the answer are concatenated together and stored in the LDAP attribute `businessCategory`.

Showing the secret question during profile management

When the user requests the profile management form, User Self Care retrieves the user attributes, including the secret question, from the registry. The profile

management STS module parses the attribute and determines the index specifying the secret question that the user has previously selected. User Self Care then uses this index value to show the appropriate value from the menu in the profile management form.

Using the secret question to validate the user identity

When the user submits the forgotten password form, User Self Care uses the user ID to retrieve the `businessCategory` registry attribute. The Forgotten Password STS module then parses the value of the attribute and returns the index to the presentation management component. This component uses the index to perform a macro substitution. The substitution provides a value to JavaScript that drives the selection of the matching secret question.

Secret question implementation tip

The security of the secret question approach is improved when users can create their own secret question. The convenience offered by a menu list is more than offset by the risk of providing pieces of identifying information. The information is often reused across many Internet sites.

The default values provided by User Self Care are for example use only. They include commonly used values such as maiden name of mother, favorite color, and name of first pet. As a preferred security practice, do not use these values in an enterprise deployment.

User Self Care URLs

User Self Care provides a set of default HTML pages for communicating with the user. The HTML pages facilitate the exchange of HTTP requests and responses.

- "User Self Care HTTP requests"
- "User Self Care HTTP responses" on page 549

User Self Care HTTP requests

The following table lists the URLs that are requested by users when interacting with User Self Care. Some URLs are listed for more than one request. Each URL is unique to a User Self Care operation and maps to an STS chain. User Self Care determines what phase of the operation is performed by examining the contents of the request.

Note: User authentication is required for some URLs. If the description does not mention user authentication, no authentication is required.

Table 140. HTTP Requests

Name	HTTP Method	Request URI and Description
Master Page	GET	Optional custom page not hosted by User Self Care. You might want to create a page that contains links to User Self Care operations but is not hosted by User Self Care.
Enrollment Request Form	GET	/sps/*federation_name*/usc/self/account/create Requests the enrollment form.
Enrollment Request Submit	POST	/sps/*federation_name*/usc/self/account/create Submits the enrollment form.

Table 140. HTTP Requests (continued)

Name	HTTP Method	Request URI and Description
Retrieve user ID	POST	/sps/*federation_name*/usc/global/userid/search Maps to a separate User Self Care operation that determines if a user ID exists. This page results from clicking a link on the Enrollment Request Form.
Enrollment Validation	POST	/sps/*federation_name*/usc/self/account/create/validate Specifies the base URL in the e-mail sent to the user during enrollment validation. The final URL has a query string appended to it.
Change Password Form	GET	/sps/*federation_name*/usc/self/password/update Authentication required Requests the change password form.
Change Password Submit	POST	/sps/*federation_name*/usc/self/password/update Authentication required Submits the change password form.
Forgotten ID Form	GET	/sps/*federation_name*/usc/self/account/recover/userid Requests the forgotten ID form.
Forgotten ID Submit	POST	/sps/*federation_name*/usc/self/account/recover/userid Submits the forgotten ID form.
Forgotten password Form	GET	/sps/*federation_name*/usc/self/account/recover/password Requests the forgotten password form.
Forgotten password Form	POST	/sps/*federation_name*/usc/self/account/recover/password Submits the forgotten password form.
Forgotten Password Secret Question Form	POST	/sps/*federation_name*/usc/self/account/recover/password/secretquestion Submits the secret question validation form. This form is presented to the user after they submit the forgotten password form.
Forgotten Password Validation Form	POST	/sps/*federation_name*/usc/self/account/recover/password/validate Specifies the base URL in the e-mail sent to the user during forgotten password validation. The final URL has a query string appended to it.
Profile Update Form	GET	/sps/*federation_name*/usc/self/profile/update Authentication required Requests the profile update form.
Profile Update Submit	POST	/sps/*federation_name*/usc/self/profile/update Authentication required Submits the profile update form.

Table 140. HTTP Requests (continued)

Name	HTTP Method	Request URI and Description
Account Delete Form	GET	/sps/*federation_name*/usc/self/account/delete Authentication required Requests the account delete form.
Account Delete Submit	POST	/sps/*federation_name*/usc/self/account/delete Authentication required Submits the account delete form.

User Self Care HTTP responses

This topic lists the set of pages that are presented by User Self Care to the user.

This set fits into the following categories:

Info
Informational page presenting instructions, errors, or a success statement.

Form
An HTML form for the user to supply data.

Redirect
An HTTP redirect.

Table 141. HTTP Responses

Name	Type	Description
Enrollment Request Form	Form	Gathers the following information: • Requested user ID • E-mail address • Password • Password confirmation • Profile attributes • Captcha input (optional)
Enrollment Validation	Form	Informs the user that an e-mail has been sent for validation purposes or that an error has occurred.
Enrollment Result	Info	Informs the user that their account has been created or that an error has occurred.
Change Password	Form	Gathers the following information: • Old password • New password • New password confirmation
Change Password Result	Info	Informs the user that their password has been changed or that an error has occurred.
Forgotten ID	Form	Gathers the following to help a user recover a forgotten user ID: • E-mail address • Captcha input. This value is optional.

Table 141. HTTP Responses (continued)

Name	Type	Description
Forgotten Password	Form	Gathers the following information: • User ID • Captcha input This value is optional.
Forgotten Password Secret Question	Form	Shows the secret question. Gathers the following information: • Answer to secret question • New password • New password confirmation • Captcha input This value is optional.
Post Forgotten ID	Info	Presents an error or success statement following attempted recovery of a forgotten ID.
Profile Update	Form	Presents the user with their current profile details and gathers modifications to the fields.
Post Profile Management	Info	Presents an error or success statement following profile management operations.
Account Delete	Form	Presents an icon for the user to click to delete their account.
Post Account Delete	Info	Presents an error or success statement following account deletion.

Validating form contents

Consider providing client-side input validation to verify that form fields contain data appropriate for their intended type. The provided User Self Care HTML pages contain several examples.

Captcha demonstration

The Captcha demonstration STS Module provides an example of how to integrate Captcha with User Self Care.

User Self Care provides HTML pages that support the user self care operations. Several of these pages are good candidates for the type of input validation that Captcha provides. You can configure these pages to include a macro for Captcha.

The User Self Care application can replace the macro value with the HTML source necessary to support the Captcha demonstration. When Captcha is not configured, the macro is not substituted and the Captcha elements are not shown on the page.

When a user initially requests a page that contains a Captcha challenge, the Captcha STS module is contacted. The module randomly selects an image from the set of configured images. This image constructs the macro on the HTML page that is shown to the user.

After the macro substitution occurs, a block of code like the example code in the figure is shown on the page.

```
<label for="demo_captcha">
    Please enter the verification word(s) shown below (required)
</label>
<br />
<img src="http://myserver/public/captcha_test/hello.jpg" border="0" />
<br />
<input type="hidden"
       name="usc.demo.captcha.challenge.field"
       id="usc.demo.captcha.challenge.field"
       value="http://myserver/public/captcha_test/hello.jpg" />
<input style="background-color:#F8F8C8;"
       type="text"
       name="usc.demo.captcha.response.field"
       id="usc.demo.captcha.response.field" />
```

Figure 67. Captcha example

This block provides a `src` tag and two input fields. The `src` tag shows the image to the user. The first input field provides the name of the image. The second gathers the user input, which is the text in the image.

When the form is submitted, the two input fields are provided to the demonstration Captcha STS module. This module compares the user answer with the string that is associated with that image. If a match is correct, the validation is finished.

Note: The first input field specifies a value that is the URL of a server hosting the images that are shown to the user.

The Captcha demonstration package is in the directory:

Federated_Identity_Manager_installation_directory`/examples/demo/captcha`

This directory contains:

- A readme file
- A `com.tivoli.am.fim.demo.sts.captcha.jar` file containing both the compiled code and the source code for the Captcha STS demonstration module.
- A `captchaTestImages` directory containing:
  - A set of six JPEG images
  - A `DemoCaptchaImagesInfo.txt` file that shows the mapping between the image file names and the text string that the user must enter when presented with the associated image.

For configuration instructions, see "Configuring the Captcha demonstration" on page 567.

Chapter 42. Deploying User Self Care

Tivoli Federated Identity Manager automatically installs User Self Care as part of the runtime. You are not required to install any additional software, unless you plan to use Tivoli Access Manager as the target user registry.

Administrations who want to deploy User Self Care must be familiar with the administration of:

- WebSphere Application Server, including the **wsadmin** administration interface.
- Tivoli Federated Identity Manager secure token service (STS) modules and trust chains.
- Tivoli Directory Server LDAP.

Administrations who want to use Tivoli Access Manager as the target user registry or WebSEAL as the point of contact serve must be familiar with Tivoli Access Manager for e-business administration.

The following list summarizes the tasks for deploying User Self Care and the order in which to perform them. Before you start a task, ensure that you have finished any prerequisite tasks.

1. Configure a Tivoli Federated Identity Manager domain. The configuration steps include configuring the runtime management.

   The steps for this task are identical for all Tivoli Federated Identity Manager scenarios. There are no tasks in this topic that are unique to User Self Care. The task links point you to common task topics in the *Tivoli Federated Identity Manager Configuration Guide.*

   "Configuring a Tivoli Federated Identity Manager domain"
2. Integrate User Self Care with the user registry for your deployment. User Self Care supports Tivoli Directory Server and Tivoli Access Manager registries. You are directed to the instructions that match your registry type.

   "Configuring a user registry" on page 557
3. User Self Care configuration relies on values obtained from a response file. In this task, you populate a response file with values applicable to your deployment.

   "Configuring a response file" on page 564
4. Use the response file created in the previous task to configure your User Self Care deployment. This step describes how to view pre-configured trust chains from the administration interface. This step also describes how to use the Tivoli Federated Identity Manager command-line interface to deploy your User Self Care environment. Optionally, you can configure the Captcha demonstration.

   "Configuring User Self Care" on page 566
5. When your deployment includes Tivoli Access Manager WebSEAL server as a point of contact server, you must integrate some User Self Care features with WebSEAL. These tasks instruct you on how to accomplish the integration.

   "Integrating User Self Care with WebSEAL" on page 609

Configuring a Tivoli Federated Identity Manager domain

You must configure a Tivoli Federated Identity Manager domain.

Before you begin

Install the following Tivoli Federated Identity Manager components:

- Runtime management
- Administration console

Procedure

1. Log on to the administration console.
2. Create a domain. Follow the instructions in Chapter 3, "Domain configuration," on page 23.

What to do next

Continue with "Configuring a user registry" on page 557.

Domain configuration

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

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

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

When Tivoli Federated Identity Manager is installed, no domains exist. Use the management console to create a domain. When you installed Tivoli Federated Identity Manager, the management service was deployed to a WebSphere server (single server mode) or WebSphere Deployment Manager (WebSphere cluster mode).

Connect with the management service and choose a WebSphere server or cluster to which you must deploy the Tivoli Federated Identity Manager runtime component. When the runtime is deployed and configured, you are ready to configure additional features such as federated single sign-on or Web services security management.

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

The Tivoli Federated Identity Manager management service uses the application deployment services of the WebSphere Deployment Manager to deploy and configure the runtime application to distributed cluster members.

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

Domain management service endpoints properties

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

```
idp.example.com
```

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

WebSphere global security properties

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

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

```
wsadmin
```

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

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

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

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

On Windows:

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

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

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

On Windows:

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

SSL Trusted Keystore password
The password that is required to access the SSL trusted keystore file.

The default password for the WebSphere key is:

```
WebAS
```

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

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

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

WebSphere server or cluster name

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

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

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

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

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

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

Tivoli Access Manager environment properties

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

The wizard presents the following prompt:

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

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

Administrator Username
The Tivoli Access Manager administrator. The default ID is sec_master. If you chose another administrator ID when you installed Tivoli Access Manager enter the administrator ID in the **Administrator Username** field.

Administrator Password
The password for the Tivoli Access Manager administrator.

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

```
idp.example.com
```

Port The port number used to communicate with the policy server.

This number matches the port number that you specified when you configured Tivoli Access Manager. The default value is 7135.

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

```
idp.example.com
```

Port The port number used to communicate with the authorization server.

This number matches the port number that you specified when you configured Tivoli Access Manager. The default value is `7136`.

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

Configuring a user registry

Integrate User Self Care with the user registry set up for your deployment.

User Self Care supports these registries through WebSphere Federated Repositories configuration:

- IBM Tivoli Directory Server. See "Configuring a Tivoli Directory Server."
- IBM Tivoli Access Manager. See "Configuring a Tivoli Access Manager adapter" on page 558.
- Microsoft Active Directory. See "Configuring an Active Directory server" on page 563.

Configuring a Tivoli Directory Server

Configure WebSphere Federated Repository for Tivoli Directory Server LDAP.

About this task

Do not use this task if you are using Tivoli Access Manager as a user registry. See "Configuring a Tivoli Access Manager adapter" on page 558.

Procedure

1. Log on to the administrative console.
2. Select the **Security** tab, and select **Global Security**.
3. Click **Configure**.
   The icon is located to the right of the **Federated Repositories** menu.
4. Click **Add Base Entry to Realm**.
5. Click **Add Repository**.
6. Enter a name for **Repository Identifier**.
   You can specify an identifier name.
7. Enter values in the following fields:
   - **Directory Type**
   - **Primary Host Name**
   - **Port**
   - **Bind distinguished name**
   - **Bind password**

   You can optionally provide values for additional fields.

8. Click **OK** and save. You now see a page that requests **Distinguished name of a base entry that uniquely identifies this set of entries in the realm**.
9. Enter a base entry name.

   If necessary, see the WebSphere Application Server documentation on WebSphere Federated Repository.

   Note: Remember the base entry name. You must use it when configuring user self care.
10. Click **OK** and save. The configuration page for **defaultWIMFileBasedRealm** is shown.
11. Examine the table labeled **Repositories in the realm**. Verify that you new realm is shown, and that the **Base Entry** is set to the value you entered.
12. Click **OK** and save. The administrative console returns to the **Global Security** page.
13. Click the **Enable Application Security** check box.
14. Click **OK** and save.

What to do next

Continue with "Configuring a response file" on page 564.

Configuring a Tivoli Access Manager adapter for WebSphere Federated Repository

To configure a Tivoli Access Manager adapter for User Self Care, you must configure the adapter and then add it to WebSphere Federated Repository as a custom registry.

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.

Complete the following tasks:

1. "Configuring a Tivoli Access Manager adapter."
2. "Configuring the adapter as a WebSphere Application Server custom registry" on page 560.

If necessary, consult the troubleshooting information in "Troubleshooting WebSphere Application Server login failures" on page 562.

Configuring a Tivoli Access Manager adapter

Configure this adapter when User Self Care manages the Tivoli Access Manager registry.

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 adapter uses the Tivoli Access Manager Registry Direct Java API to perform administration commands such as creating users and groups. The Tivoli Access Manager installation provides this adapter.

Note: If you are not using a Tivoli Access Manager adapter, do not use these instructions. See "Configuring a Tivoli Directory Server" on page 557.

Procedure

1. Ensure that you have installed Tivoli Access Manager.
2. Ensure that you have installed and configured Tivoli Access Manager using Tivoli Directory Server as the user registry.
3. Ensure that you have installed the Tivoli Access Manager 6.1.1 Java run time component.
4. Copy *TAM_installation_directory*/java/export/rgy/com.tivoli.pd.rgy.jar to *WebSphere_installation_directory*/lib.
5. Create a Tivoli Access Manager user identity that runs the Java API.

   For example:

   ```
   pdadmin -a sec_master -p sec_master_password
   pdadmin sec_master> user create -no-password-policy user_name
   cn=user_name,registry_suffix user_name user_name password
   ( SecurityGroup ivacld-servers remote-acl-users )
   pdadmin sec_master> user modify user_name account-valid yes
   ```

   In the example, *user_name* is your choice of name for the user. A good naming scheme would be:

   ```
   tamVMMAdapter-machine_name
   ```

   The value *registry_suffix* is the suffix of the registry where this user must be stored. For example:

   ```
   o=ibm,c=us
   ```

6. Go to the computer where the Tivoli Access Manager adapter is to be configured. Change directory to *WebSphere_installation_directory*/lib . Run the **com.tivoli.pd.rgy.until.RgyConfig** tool.

   Use the IBM Java runtime environment to run this tool. For example:

   ```
   <WebSphere install>/AppServer/java/jre/bin/java
   ```

Table 142. Using the com.tivoli.pd.rgy.util.RgyConfig utility

Syntax:

```
java com.tivoli.pd.rgy.util.RgyConfig properties_file_destination create Default
Default "ldaphostname:389:readwrite:5" "DN" DN_password
```

properties_file_destination
Specifies the full path to an existing directory and the name of a file that is created when this command is run. Place the file in a directory appropriate for your WebSphere Application Server deployment:

- For a non-clustered WebSphere Application Server server:

  WebSphere_application_server/profiles/*server_name*/config/itfim

- For a WebSphere Application Server cluster (replicated) environment, create the file on the DMgr:

  WebSphere_application_server/profiles/*DMgr_server_name*/config/itfim

ldaphostname
The host name of the LDAP server to which Tivoli Access Manager is configured. The host name is specified in the Tivoli Access Manager runtime configuration file:

Tivoli Access Manager_installation_directory/etc/ldap.conf

389
The default LDAP port. Modify as needed for your deployment.

"DN"
The Distinguished Name (DN) specified in the **pdadmin** user creation command. Ensure that the value is surrounded by double quotation marks.

DN_password
The password for the DN.

Example command:

```
java com.tivoli.pd.rgy.util.RgyConfig
WebSphere_application/profiles/<server>/config/itfim/tamVMMAdapter.properties
 create Default Default "myldapsystem:389:readwrite:5"
"cn=tamVMMAdapter-myhost,o=ibm,c=us" mypasswordmypassword
```

7. Update the configuration as needed for your WebSphere Application Server deployment:
   - For a non-clustered WebSphere Application Server server, reload the Tivoli Federated Identity Manager configuration.
   - For a WebSphere Application Server cluster (replicated) environment, perform a full WebSphere Application Server resynchronization, and reload the Tivoli Federated Identity Manager configuration.

What to do next

Continue with "Configuring the adapter as a WebSphere Application Server custom registry."

Configuring the adapter as a WebSphere Application Server custom registry

To integrate with WebSphere, configure the Tivoli Access Manager adapter as a WebSphere Application Server custom registry.

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.

Complete the task "Configuring a Tivoli Access Manager adapter" on page 558.

About this task

After configuring the Tivoli Access Manager adapter with the Tivoli Access Manager runtime environment, you must configure the Virtual Member Manager (VMM) Tivoli Access Manager Adapter into WebSphere Application Server as a custom registry.

Note: For information about configuring WebSphere Federated Repository custom registries, see the WebSphere Application Server documentation. For WebSphere Application Server Network Deployment 6.1, see the WebSphere information center.

Procedure

1. Stop the WebSphere Application Server.
2. Change directory to:

   WebSphere_Installation_directory/profiles/*profile_name*/config/cells/*cell_name*/wim/config
3. Use a text editor to open `wimconfig.xml`.

   Note: Back up `wimconfig.xml` before you change it.
4. Add a new `config:repositories` element to the file. Place this element before the `config:realmConfiguration` element.

   This entry specifies the class name of the adapter, and sets an identifier for the repository. For example, to specify a class name of `com.tivoli.pd.vmm.adapter.tam.TAMRegistryAdapter` and to set the `TAMRegistryAdapter` repository as the identifier:

   ```
   <config:repositories
   adapterClassName="com.tivoli.pd.vmm.adapter.tam.TAMRegistryAdapter"
   id="TAMRegistryAdapter"/>
   ```
5. Save the wimconfig.xml file, and close the text editor.
6. Copy the *TAM_installation_directory*/java/export/vmm_tam_adapter/VMMTamAdapter.jar file to *WebSphere_install_directory*/lib.
7. Start **wsadmin** in **no connection** mode:

   ```
   wsadmin -conntype none
   ```
8. Disable paging in the common repository configuration. Set the `supportPaging` parameter for the `updateIdMgrRepository` command to `false` to disable paging.

   ```
   $AdminTask updateIdMgrRepository {-id TAMRegistryAdapter
    -supportPaging false }
   ```

   Note: A warning is shown until the configuration of the sample repository is finished.
9. Add a custom property for the `TAMRegistryAdapter`.

```
$AdminTask setIdMgrCustomProperty {-id TAMRegistryAdapter
-name tamConfFile -value "properties_file_destination"}
```

properties_file_destination
The properties file that was created as the result of running **com.tivoli.pd.rgy.util.RgyConfig** in the prerequisite task "Configuring a Tivoli Access Manager adapter" on page 558.

10. Add a base entry to the adapter configuration using the **addIdMgrRepositoryBaseEntry** command to specify the name of the base entry for the specified repository:

```
$AdminTask addIdMgrRepositoryBaseEntry {-id TAMRegistryAdapter
-name base_entry_name }
```

base_entry_name
This name must match the suffix used by the Tivoli Access Manager user registry.

11. Use the **addIdMgrRealmBaseEntry** command to add the base entry to the realm. This action links the realm with the repository.

```
$AdminTask addIdMgrRealmBaseEntry {-name defaultWIMFileBasedRealm
-baseEntry base_entry_name }
```

base_entry_name
This name must match the value you specified in the previous command.

defaultWIMFileBasedRealm
The default realm name is `defaultWIMFileBasedRealm`. If this realm name was renamed, use the new realm name instead of `defaultWIMFileBasedRealm`.

12. Save your configuration changes. Enter the following commands to save the new configuration and close the **wsadmin** tool:

```
$AdminConfig save
exit
```

13. Restart the WebSphere Application Server.

What to do next

Select one of the following tasks:

- If you can successfully log on to WebSphere Application Server, continue with "Configuring a response file" on page 564.
- If you cannot log on to WebSphere Application Server, see "Troubleshooting WebSphere Application Server login failures."

Troubleshooting WebSphere Application Server login failures

If you cannot log on to WebSphere Application Server following configuration of the adapter, review these troubleshooting tips.

About this task

If a registry cannot be contacted, WebSphere Application Server prevents you from logging on. This limitation occurs even if the WebSphere Application Server administration account is located in a different registry. Wrong configuration or lack of availability of a required registry can result in WebSphere Application Server preventing you from logging in as the administrator.

If you encounter this problem after configuring the Tivoli Access Manager adapter, follow the troubleshooting steps in this procedure:

Procedure

1. Ensure that the Tivoli Access Manager registry is available. Since Tivoli Access Manager Registry adapter does not maintain an authentication cache, you see a "cannot log in" error immediately when the registry is unavailable.
   a. Use **pdadmin** to connect to the registry and perform a test user creation to confirm.
   b. Restart the registry and correct any connection issues if necessary.
   c. If the problem persists, continue to the next step.
2. Open the `wimconfig.xml` file and verify the settings in the new code that you created.

```
<config:repositories adapterClassName="com.tivoli.pd.vmm.adapter.tam.TAMRegistryAdapter"
id="TAMRegistryAdapter" supportPaging="false">
<config:baseEntries name="o=ibm,c=us"/>
<config:CustomProperties
name="tamConfFile"
value="/opt/IBM/WebSphere/AppServer/profiles/dmgr/config/itfim/tamVMMAdapter.properties"/>
</config:repositories>
```

Figure 68. Sample wimconfig.xml settings

   - Confirm that the location or name of the properties file is correct.
   - Confirm that the suffix is correct for the Tivoli Access Manager registry.

   Note: If you modify the configuration file, you must restart WebSphere Application Server. WebSphere Application Server requires you to log on as the administrator to stop WebSphere Application Server. However, if you cannot log on you must stop the WebSphere Application Server process. You can then restart WebSphere Application Server without logging on.
3. If in the previous step you did not identify any problems with the configuration file, roll back to the backup copy of the `wimconfig.xml` file.
   a. Make a backup of your new `wimconfig.xml` file.
   b. Restore the backup of the original `wimconfig.xml` file.
   c. Restart WebSphere Application Server.

      Note: WebSphere Application Server requires you to log on as the administrator to stop WebSphere Application Server. However, if you cannot log on you must stop the WebSphere Application Server process. You can then restart WebSphere Application Server without logging on.

   If you can log in after restoring the backed up file, there is a problem with the Tivoli Access Manager adapter configuration. Review the configuration and correct any errors.

Configuring an Active Directory server

Configure WebSphere Federated Repository for Microsoft Active Directory.

About this task

Do not use this task if you are using Tivoli Access Manager as a user registry. See “Configuring a Tivoli Access Manager adapter” on page 558.

Procedure

1. Log on to the administrative console.
2. Select the **Security** tab, and select **Global Security**.
3. Click **Configure**.
   The icon is located to the right of the **Federated Repositories** menu.
4. Click **Add Base Entry to Realm**.
5. Click **Add Repository**.
6. Enter a name for **Repository Identifier**.
   You can specify an identifier name.
7. Enter values in the following fields:
   - **Directory Type**
   - **Primary Host Name**
   - **Port**
   - **Bind distinguished name**
   - **Bind password**

   You can optionally provide values for additional fields.
8. On the WebSphere Application Server console, select **Require SSL communications**.

   Note: Configuration of SSL communication between a WebSphere Application Server and a user registry such as Active Directory requires additional steps. See the documentation for your version of WebSphere Application Server for instructions on configuring SSL connections with WebSphere Application Server.
9. Click **OK** and save. You now see a page that requests **Distinguished name of a base entry that uniquely identifies this set of entries in the realm**.
10. Enter a base entry name.
    If necessary, see the WebSphere Application Server documentation on WebSphere Federated Repository.

    Note: Remember the base entry name. You must use it when configuring user self care.
11. Click **OK** and save. The configuration page for **defaultWIMFileBasedRealm** is shown.
12. Examine the table labeled **Repositories in the realm**. Verify that you new realm is shown, and that the **Base Entry** is set to the value you entered.
13. Click **OK** and save. The administrative console returns to the **Global Security** page.
14. Click the **Enable Application Security** check box.
15. Click **OK** and save.

What to do next

Continue with "Configuring a response file."

Configuring a response file

Create a response file and then populate it with values specific to your deployment.

About this task

User Self Care loads configuration from an XML properties file called a *response file*. This file contains the responses to configuration options. In most cases, response file contents are generated by administrator choices during initial deployment. For user self care, loading a properties file is required as part of initial configuration.

Procedure

1. Create a response file as needed for your deployment. Use **wsadmin**:

   ```
   $AdminTask manageItfimUserSelfCare {-operation createResponseFile
   -fileId target_location }
   ```

   The value *target_location* is the fully qualified path to a file that is created.

2. Determine a value of each parameter in the response file, as required by your deployment.

   Optionally, use the following worksheet to plan your response file. The worksheet identifies the required parameters. In the response file, you can search for the string `REQUIRED` to find these parameters.

   For more information about each parameter in this worksheet, see Chapter 44, "Response file parameters," on page 621

Table 143. User Self Care response file parameters

Response file parameter	Required?	Default Value	Your value
AccountCreateLifetime	yes	86400	
AccountRecoveryFailureLifetime	no	86400	
AccountRecoveryFailureLimit	no	3	
AccountRecoveryFailureLockoutTime	no	86400	
AccountRecoveryLookupAttribute	no	mail	
AccountRecoveryLookupField	no	none	This field is deprecated.
AccountRecoveryValidationAttributes	no	mail	
AccountRecoveryValidationLifetime	no	86400	
AttributeMappingFilename	yes	none	
BaseURL	yes	none	
CaptchaSTSModuleId	yes	default-usc-captcha-noop	
DemoCaptchaImageAndKeyList	Yes, if using Captcha	Fixed content.	Do not modify.
DemoCaptchaImageRootURL	Yes if using Captcha	none	
EnrollmentEmailSender	yes	none	
EntitySuffix	yes	o=ibm,c=us	
GroupMembershipGroups	no	none	
PasswordRecoveryEmailSender	yes	none	
ProfileManagementAttributes	yes	businessCategory roomNumber mobile mail	
SecretQuestionMinimumNumber	no	2	
SecretQuestionMaximumNumber	no	3	
SecretQuestionRequiredForValidationNumber	no	2	
SMTPAuthenticatePassword	no, unless your SMTP server requires it	none	
SMTPAuthenticateUsername	no, unless your SMTP server requires it	none	
SMTPServerName	yes	none	

3. Update your response file with the values.
4. Save the file.

What to do next

Continue with the topic: "Configuring User Self Care."

Configuring User Self Care

Follow the steps in this procedure to configure user self care with the Tivoli Federated Identity Manager deployment.

Before you begin

Ensure that you have finished the prerequisite configuration tasks:

1. "Configuring a Tivoli Federated Identity Manager domain" on page 553
2. "Configuring a user registry" on page 557
3. "Configuring a response file" on page 564

About this task

Do the following tasks in order. The instructions for each task provide a link to the next task. The list of tasks is shown here as an overview.

Procedure

1. "Showing trust chains"
2. "Configuring the Captcha demonstration" on page 567
   Skip this step if you do not intend to use the Captcha demonstration.
3. "Using a response file to configure User Self Care" on page 568
4. "Configuring a point of contact server" on page 568
5. "Integrating User Self Care with WebSEAL" on page 609
   Skip this step if you are using WebSphere Application Server as the point of contact server.

Showing trust chains

You can configure Tivoli Federated Identity Manager to show the trust chains that are created by default for User Self Care.

About this task

You can use the administrative console to view each trust chain. Trust chains correspond to one or more User Self Care operations. When you view the trust chains, you see the STS modules that accomplish the operation. You can then customize the modules and chains to fit your deployment.

Note: For information about how to customize User Self Care, see the Tivoli Federated Identity Manager Wiki:

http://www.ibm.com/developerworks/wikis/display/tivolifederatedidentitymanager/Home

Procedure

1. Log on to the administrative console.
2. Go the Runtime Node Management panel.

3. In the custom property part of the panel, select the menu entry for `STS.showUSCChains`.
4. Set the value to `true`.
5. Save the configuration.
6. When prompted, load the configuration changes.
7. Restart WebSphere Application Server to refresh the management commands available to **wsadmin**.

What to do next

Select one of the following steps:

- If you want to use the Captcha demonstration, continue with "Configuring the Captcha demonstration."
- If you do not want to use the Captcha demonstration, continue with "Using a response file to configure User Self Care" on page 568.

Configuring the Captcha demonstration

You can optionally configure the Captcha demonstration as part of your User Self Care deployment.

Before you begin

Ensure that you have finished all of the prerequisite configuration steps:

- "Configuring a Tivoli Federated Identity Manager domain" on page 553
- "Configuring a user registry" on page 557
- "Configuring a response file" on page 564
- "Showing trust chains" on page 566

Procedure

1. Host the provided image files on a web server that is accessible to your users.

   Ensure that you know the location of the root URL of the images that are used during the configuration of the Captcha STS module. The value is stored in the DemoCaptchaImageRootURL parameter in the response file.
2. Activate the plug-in:
   a. Copy the Captcha jar file to the Tivoli Federated Identity Manager plug-ins directory. For example, copy:

      FIM_install_dir`/examples/demo/captcha/com.tivoli.am.fim.demo.sts.captcha.jar`

      to the directory:

      TFIM_install_dir`/plugins`
   b. Using the Runtime Node Management Panel, click the **Publish Plugins** icon.
   c. Click **Load Configuration Changes**.
3. Use the Module Instances Panel to create an instance of the DemoCaptchaSTSModule. Set the Module Instance Name to the value `usc-captcha-demo`.

What to do next

Continue with "Using a response file to configure User Self Care" on page 568.

Using a response file to configure User Self Care

Use the response file that you created previously to supply the necessary properties to the configuration command for user self care.

Before you begin

Ensure that you have finished the prerequisite configuration tasks:

- "Configuring a Tivoli Federated Identity Manager domain" on page 553
- "Configuring a user registry" on page 557
- "Configuring a response file" on page 564
- "Showing trust chains" on page 566
- If you are using the Captcha demonstration, ensure that it is configured. See "Configuring the Captcha demonstration" on page 567

Procedure

1. Obtain your configured response file.
2. Run **wsadmin**.

   ```
   wsadmin.sh -username WebSphere_adminstrator_name -password password
   ```

3. Load the response file:

   ```
   $AdminTask manageItfimUserSelfCare {-operation configure -fimDomainName
   domain_name -federationName federation_name
    -fileId response_file_path }
   ```

   Supply these values:

   domain_name
   : The name of the Tivoli Federated Identity Manager domain that you created.

   federation_name
   : The name of the Tivoli Federated Identity Manager federation that you created.

   response_file_path
   : The location of your User Self Care response file.

4. Reload the Tivoli Federated Identity Manager configuration.

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

   Supply this value:

   domain_name
   : The name of the Tivoli Federated Identity Manager domain that you created.

What to do next

Continue with the topic: "Configuring a point of contact server."

Configuring a point of contact server

You must configure a point of contact server for User Self Care.

Select the instructions for the type of point of contact server that your deployment uses:

- "Configuring WebSphere Application Server as a point of contact server" on page 569
- "Configuring WebSEAL as a point of contact server" on page 569

Configuring WebSphere Application Server as a point of contact server

You can configure WebSphere Application Server as the point of contact server for User Self Care.

Procedure

1. Use **wsadmin** to activate the **WebSphere** point of contact type.

   Use the **wsadmin** commands:

   ```
   $AdminTask manageItfimPointOfContact {-operation activate
   -uuid uuid4f3d17d-0106-w412-r36b-a0d5ecc604ba
   -fimDomainName your_domain_name}
   ```

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

2. Log on to the administrative console.
3. Select **Enterprise Applications > ITFIMRuntime > Security role to user/group mapping**.
4. Update the **FIMUserSelfCareAnyAuthenticated** application role to be **AnyAuthenticated**.
5. Save the WebSphere Application Server configuration.
6. Restart WebSphere Application Server.

What to do next

Review the performance tuning guidelines in Chapter 43, "Tuning User Self Care," on page 617.

Configuring WebSEAL as a point of contact server

You can configure WebSEAL as the point of contact server for User Self Care.

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.

Procedure

1. Determine the location of your **tfimcfg.jar** file.

   This file is located in the hierarchy under the Tivoli Federated Identity Manager installation directory. On UNIX, the path is:

   ```
   /opt/IBM/FIM/tools/tamcfg/tfimcfg.jar
   ```

2. Run the `tfimcfg` tool.

   ```
   java -jar tfimcfg.jar -action tamconfig
   -cfgfile /opt/pdweb/etc/webseald-default.conf
   ```

   Note: If Federal Information Processing Standards (FIPS) is enabled in your environment, the secure socket connection factory must be specified. For example:

   ```
   java -jar /download_dir/tfimcfg.jar -action tamconfig
   -cfgfile webseald-instance_name.conf -sslfactory TLS
   ```

   Usage notes:

   - The file paths might differ for your installation and your WebSEAL instance.

- The default Tivoli Federated Identity Manager HTTP port is 9080. This port is also the WC_defaulthost port for the WebSphere Application Server.
- Do not specify an optional Tivoli Federated Identity Manager administrator user ID or password.
- Answer `no` to the question `Use SSL connection to ITFIM server`.
- Select `uscfed` from the list of Federations to configure.

What to do next

Continue with the topic: "Integrating User Self Care with WebSEAL" on page 609.

Modifying checks on user ID and password

User data can include information about the user such as user name, password, email address, and the secret question and answer. The user data is checked and validated in the HTML pages, mapping rule, and federated repository. The validation of user data in the federated repository is not covered in this document.

User data is checked and validated in three places:

- In the JavaScript in the User Self Care HTML pages

  In the default setup of User Self Care, the JavaScript in the HTML validates the following user data:

  – Missing required field values

  – Invalid characters: `[ ] / < > ( ) , ; : \" = "`

  – The mobile number field for values other than numbers, spaces, and the following characters: `( ) -`

  See "Overview of the HTML validation function" on page 571 for more details.
- In the mapping rule

  In the default setup of User Self Care, the mapping rule validates the user data for the following details:

  – Password length

    - The minimum password length is seven characters
    - The minimum number of alphanumeric characters in a password is four
    - The minimum number of non-alphanumeric characters in a password is one
    - The maximum number of repeating characters in a password is two

  – User name length

    The maximum user name length is 256 characters

  – User name and password must contain only letters or digits

  See "Overview of the mapping file validation function" on page 571 for more details.
- In the federated repository

  The LDAP server can enforce limitations on the values of the user data. One example is Active directory. The user name and password rules that are set in the Account Policies can enforce the user password to be more than seven characters.

To modify the user name or password policy, one or more of the following files must be edited. See "Modifying the validation for user name and password" on page 574 for more details.

HTML page	Page description
enrollment.html	Registration form.
changepassword.html	Change password form.
forgotid.html	Form that retrieves user ID by using the registered email address.
forgotpassword.html	Form that resets the password if a user forgets the password.
secretquestion.html	Page that is displayed to accept the answer to the secret question and the new password in the forgot password flow.
profile.html	Form that updates the profile information of the user.

Overview of the HTML validation function

The JavaScript function `testInput(...)` validates the user name and password in the HTML pages.

This function is in all the HTML pages that validate the user name and password.

The following function is an example from the `enrollment.html` page.

```
function testInput(required, fieldName, fieldVal){
   ...
if ( required && fieldVal == "" ) {
    ...
                    return false;
                }
if ( fieldVal.match(illegalChars) ) {
   ...
                    return false;
               }
    return true;
            }
```

The input parameters that are accepted are:

- `required`: This parameter accepts true or false values. If this parameter is set to true, then the `fieldName` is a required field.
- `fieldName`: This parameter is the name of the field.
- `fieldVal`: This parameter is the value that is supplied by the user for the field `fieldName`.

Table 144. Conditions found in the HTML validation function

Conditions	Description
`if ( required && fieldVal == "" ) { ... return false; }`	Checks for all required fields.
`if ( fieldVal.match(illegalChars) ) { ... return false; }`	Checks for characters that are not valid.

Overview of the mapping file validation function

The mapping rule has different variables and functions that check and validate user data in User Self Care.

Variables	Value
MIN_PASSWORD_LENGTH	7
MIN_PASSWORD_ALPHA	0
MIN_PASSWORD_NON_ALPHA	0
MAX_PASSWORD_REPEAT_CHARS	2

Function	Description
function validUsernameCharacter(cp) {...}	Returns true only when the character in the user name is a letter or number.
function validPasswordCharacter(cp) {...}	Returns true only when the character in the password is a letter or number.

function checkUsername(helper) {
...
if (userid.length() > MAX_USERNAME_LENGTH) {
//Checking for maximum length of username
...
return;
}
...
for (var i = 0; i < cp.length; i++) {
// only allow letters and numbers in usernames
if (validUsernameCharacter(cp[i]) == false) {
...
return;
}
}
}
	Checks the user name for length and allowed characters. It calls
	validUsername Character
	
	It uses the value in the following variable:
	MAX_USERNAME _LENGTH

function checkPassword(helper) {
...
if (password.length() < MIN_PASSWORD_LENGTH) {
//checking for minimum password length
...
return;
}
...
for (var i = 0; i < cp.length; i++) {
if (validPasswordCharacter(cp[i]) == false) {
//checking for valid characters in the password-only
letters and numbers
...
return;
}
...
}
if (alphas < MIN_PASSWORD_ALPHA) {
//password should not have less than
MIN_PASSWORD_ALPHA alphabets
...
return;
}
if (nonalphas < MIN_PASSWORD_NON_ALPHA) {
//password should not have less than
MIN_PASSWORD_NON_ALPHA
...
return;
}
if (repeats > MAX_PASSWORD_REPEAT_CHARS) {
// password cannot have more than
MAX_PASSWORD_REPEAT_CHARS repeat //characters
...
return;
}
}
	Checks for the following password data:
	• Minimum length
	• Valid characters
	• Allowed number of alphabetical and non-alphabetical characters
	• Maximum number of repeat characters

Condition	Description

if (CHECK_PASSWORD[operation] == 1) {
System.out.println("USC DEBUG MAPPING:
enforcing default password policy.");
checkPassword(helper);
}
	These conditions are the entry points to the mapping rule.

if (CHECK_USERNAME[operation] == 1) {
System.out.println("USC DEBUG MAPPING:
enforcing default username policy.");
checkUsername(helper);
}	

CHECK_PASSWORD[operation]=1	By default, all the operations are mapped to this condition.

`var CHECK_PASSWORD = {`
`"/usc/self/account/create/post": 1,`
`"/usc/self/account/recover/password/secretquestion/post": 1,`
`"/usc/self/password/update/post": 1`
`};`
	You can modify this condition by changing the array and adding more conditions to handle them.

`if (CHECK_PASSWORD[operation] == 1) {`
`System.out.println("USC DEBUG MAPPING: enforcing default password policy.");`
`checkPassword(helper);`
`}`	

Modifying the validation for user name and password

You can modify the different validation or checking of user data in User Self Care.

About this task

Modifying the `enrollment.html` is covered in this procedure. You must repeat the steps for all other affected HTML files:

- `enrollment.html`
- `forgotid.html`
- `forgotpassword.html`
- `secretquestion.html`
- `profile.html`

Procedure

1. Edit the HTML page.

   If you want to remove the required field validation for the user ID field

   Modify the input to `testInput(required, fieldName, fieldVal` function in

   ```
   if ( ! testInput(true,"User ID", useridTF.value) ){
                       return true;
                   }
   ```

   to

   ```
   if ( ! testInput(false,"User ID", useridTF.value) ){
                       return true;
                   }
   ```

   If you want to change the label on the user ID field

   Modify

   ```
   <label for="usc.form.userid">
                   User ID (required)
               </label>
   ```

   to

   ```
   <label for="usc.form.userid">
                   User ID
               </label>
   ```

   If you do not want to enable the check for the following characters that are not valid in user ID, email address, password, and secret question answer:

   - [
   -]
   - /

- <
- >
- (
-)
- ,
- ;
- :
- \
- "
- =
- "

Look for function `testInput(required, fieldName, fieldVal)` and comment the section.

```
if ( fieldVal.match(illegalChars) ) {
      var illegalCharsStr = "The following characters are not
   allowed: [ ] \\ / < > ( ) , ; : \" = ";
       var invalidCharsFieldStr = "The following field contains
   illegal characters: " + fieldName + "<br> " + illegalCharsStr;
        printWarning(invalidCharsFieldStr);
        return false;
    }
```

If you do not want to check the mobile number field for characters other than numbers, spaces, ( ) and -

Look for function `doSubmit()` and comment out the section.

```
// var mobileCleaned = mobileTF.value.replace(/[\(\)\-\ ]/g, '');
//    if (isNaN(mobileCleaned)) {
//    var illegalMobileCharsStr = "The mobile number can only contain
//    numbers, spaces and the following characters: ( ) - ";
//    printWarning(illegalMobileCharsStr);
//    return true;
// }
```

2. Publish pages to the Tivoli Federated Identity runtime environment.
3. Modify the mapping rule file.
   a. Back up the JavaScript mapping rule file.
   b. Open the JavaScript mapping rule file with a text editor.
   c. Search for the following variables and edit accordingly.
      - MIN_PASSWORD_LENGTH
      - MIN_PASSWORD_ALPHA
      - MIN_PASSWORD_NON_ALPHA
      - MAX_PASSWORD_REPEAT_CHARS
      - MAX_USERNAME_LENGTH
   d. Edit according to the scenarios detailed in the following table.

      If you want to allow more characters in the user name field other than letters and digits

      Search for `validUsernameCharacter(cp)` and replace the return value `Character.isLetterOrDigit(cp)` appropriately.

      Note: Some special characters might need to be escaped with a backslash (\).

If you want to allow any character in the user name field

Replace `Character.isLetterOrDigit(cp)` with `true`. The federated repository that you are using must allow that special character.

If you want to have different policies for a different workflow

Change the variables `CHECK_USERNAME` and `CHECK_PASSWORD`.

For example, if you want to have a different policy setting when the user want to change the password, do the following steps.

1) Change the value for the variable `CHECK_PASSWORD`.

```
var CHECK_PASSWORD = {
"/usc/self/account/create/post": 1,
    "/usc/self/account/recover/password/secretquestion/post": 1,
    "/usc/self/password/update/post": 2
};
```

2) Introduce a new condition.

```
if (CHECK_PASSWORD[operation] == 2) {
     System.out.println("USC DEBUG MAPPING:
enforcing modified password  policy.");
     modifiedCheckPassword(helper);
 }
```

3) Introduce the new function `modifiedCheckPassword(helper)`.

```
function modifiedCheckPassword(helper) {
     var password = helper.getNewPassword();
     var pwdattr = [ "usc.form.password.new",
  "usc.form.password.new.confirm" ];

     if (helper.getUserRecoverableError()) {
 return;
     }

     if (password.length() < 10) {
       helper.setUserRecoverableError("Password should
be atleast 10  characters", pwdattr);
 helper.setSTSOutputPageID(FORM_PAGE_IDS[operation]);
 return;
     }
 }
```

4. Save the changes to the file.
5. Apply the modified mapping rule to the Trust Service Chains.
   a. Log in to Integrated Solutions Console.
   b. Navigate to **Tivoli Federated Identity Manager** > **Configure Trust Service** > **Trust Service Chains**.
   c. Select **USC** under **Show Chain Types**.
      1) If you are modifying policies that are related to user name alone, modify the following chains.
         - **Default Chain uscCreateAccount**
         - **Default Chain uscDeleteAccount**
         - **Default Chain uscForgottenId**
         - **Default Chain uscProfileManagement**- All the two chains.
      2) If you are modifying policies that are related to password alone, modify the following chains.
         - **Default Chain uscChangePassword**
         - **Default Chain uscCreateAccount**
         - **Default Chain uscDeleteAccount**
         - **Default Chain uscForgottenPassword**- All the two chains.

- **Default Chain uscProfileManagement**- All the two chains.

3) If you are modifying the policies that are related to both user name and password, modify all the chains.

d. Select the chain mapping that you want to modify.

e. Click **Properties**.

f. Under **Trust Service Chain Modules**, select **Default mapping module**.

g. Click **Properties**.

h. Under Identity Mapping Rule for partner module chain **Default Chain uscCreateAccount**, click **Modify Rule**.

i. Click **Import File**.

j. Click **OK**.

k. Click **Load configuration changes to Tivoli Federated Identity Manager runtime**.

Note: Repeat the steps for all the chains that you want to modify.

Enabling multiple secret question

Use the multiple secret question feature to enhance the security of validating user credentials.

Secret Questions in User Self Care

A secret question is a question that the user must answer during enrollment. It is typically personal information that only the user knows. In User Self Care, the secret questions and answers verify the identity of users when they forget their password.

Users are required to answer the secret questions correctly for their identities to be verified. Example secret questions are:

- What is your mother's maiden name?
- What is the name of your first pet?

Every pair of question ID and answer are then concatenated and stored in a multi-valued LDAP attribute. By default, it is the `businessCategory` attribute. If you enable the salt and hash feature, the original answer is never stored in the LDAP. This feature increases the security of storing the answers.

The secret question configuration is stored in the:

- User interface for storing secret questions in User Self Care HTMLs.
- User interface for answering secret questions in User Self Care HTMLs.

- Configuration for storing and fetching secret questions to and from the LDAP in the mapping file.
- Security Token Service (STS) modules configurations.

Note: By default, only the secret question ID is stored in the LDAP attribute. The secret question string is stored in the User Self Care HTML files.

An identity of a user can be compromised when other people know the answer to the secret question. You can configure User Self Care to use more than one secret question to validate the identity of users. Multiple secret questions provide a more secure way of validating users when they request a password reset.

Updating configuration settings for existing User Self Care federations

After you install Tivoli Federated Identity Manager version 6.2.2, fix pack 4, you must update the configuration for the User Self Care federations.

About the User Self Care STS chain migration:

Tivoli Federated Identity Manager version 6.2.2, fix pack 4 automatically updates the STS chains. The migration affects all existing User Self Care federations.

The following changes are applied after you install fix pack 4:

- The STS chains are updated to accommodate multiple secret questions.
- The runtime custom property `USC.SecretQuestion.SaltAndHash.Enabled` is set to `false`.

Chains updated	Modules added	Properties
`uscAccountCreate`	`USCSecretQuestionStoreSTSModule` in `validate` mode	• **Minimum number of secret questions that a user is required to answer**. The default value is **1**. • **Maximum number of secret questions that a user can answer.** The default value is **1**.
`uscProfileManagement`	`USCSecretQuestionStoreSTSModule` in `validate` mode	• **Minimum number of secret questions that a user is required to answer**. The default value is **1**. • **Maximum number of secret questions that a user can answer**. The default value is **1**.
`uscForgottenPassword`	`USCSecretQuestionSTSModule` in `issue` mode	**Minimum required number of secret questions that a user must answer correctly to validate their identity**. The default value is **1**.

Enabling salting and hashing on existing User Self Care federations:

You can enhance the security of storing existing secret question answers by encrypting them.

About this task

In cryptography, salting is a method encrypting inputs such passwords by adding a random string to them. This makes passwords more secure. Hashing is a method of using an algorithm to convert data to a fixed-sized value. In User Self Care you can use these encryption methods to secret question answers. Enabling the salting and hashing feature increases the security of storing the secret question values. To enable the salting and hashing of existing secret question values set the runtime custom property `USC.SecretQuestion.SaltAndHash.Enabled` to `true` in the Integrated Solutions Console.

Migrating the User Self Care secret question answers in LDAP to a new format:

Store the secret question values in a more secure format. Use the LDAP migration tool to migrate existing User Self Care secret question answers to a salted and hashed format.

Before you begin

- Install Tivoli Federated Identity Manager version 6.2.2, fix pack 4
- Configure User Self Care

Note:

You do not need to update the LDAP if you are configuring User Self Care for the first time in Tivoli Federated Identity Manager version 6.2.2, fix pack 4 onwards.

CAUTION:
The migration of secret question answers is irreversible. The hashing of the secret question answer is one way. You must back up the original secret question answers before doing the migration. This backup can restore the original values if the migration is not successful. However, if you restore the secret question entries by using the backup file, the migrated hashed values cannot be retrieved.

About this task

In cryptography, salting is a method encrypting inputs such passwords by adding a random string to them. This makes passwords more secure. Hashing is a method of using an algorithm to convert data to a fixed-sized value. Use this tool to salt and hash existing secret question answers.

LDAP migration tool

The LDAP migration tool:

- Backs up the existing secret question answers to the LDAP Data Interchange Format
- Creates an LDAP Data Interchange Format (LDIF) file for migration with the migrated salted and hashed values. This file can be executed later to do the LDAP migration.
- Migrates the existing LDAP secret question values directly in the LDAP

The parameters for the LDAP migration tool include:

- -h: Specifies the LDAP machine hostname.
- -p: (Optional) Specifies the LDAP port number. The default port number is 389.
- -D: Specifies the bind user distinguished name (DN). For example, `cn=admin,dc=example,dc=com`.
- -w: Specifies the password of the bind user.
- -baseDN: Specifies the base DN to be searched. For example `dc=example,dc=com`.
- -attribute: LDAP attribute used to store secret question. For example, `businessCategory`.
- -newattribute: (Optional) New secret question answer. This attribute must be a multi-valued attribute if you use multiple secret questions. If this attribute is not specified, the destination attribute is the same as the original attribute specified in **-attribute**.
- -ldif: (Optional) Writes the changes to a specified LDIF file instead of doing the migration.
- -deleteOldEntry: (Optional) This parameter works only if **-attribute** and -newattribute are specified. If this parameter is present, the old attribute that was specified in **-attribute** is deleted after the migration is completed. If -ldif is also specified, the attribute is not deleted immediately, but the LDIF file contains the commands to delete those entries.
- -backup: (Optional) Writes a backup of the current value of attribute to the specified LDIF file.
- -Z: (Optional) Specifies whether SSL is used to connect to LDAP.

Procedure

1. Enable salting and hashing on secret question values. See, "Enabling salting and hashing on secret question values" on page 582.
2. Stop the Tivoli Federated Identity Manager runtime application.
   a. In the Integrated Solutions Console, navigate to **Applications** > **Application Types** > **WebSphere enterprise applications**
   b. Select **ITFIMRuntime**.
   c. Click **Stop**.
3. Open the command prompt.
4. Run the following commands for the LDAP migration tool:

   Important: Use the `-backup` parameter to ensure that the backup copy is created.

   ```
   java -classpath <FIM_INSTALL_PATH>\tools\ldap
   com.tivoli.am.fim.ldap.MigrateUSCSecretQuestion [parameters]
   ```

   For example:

   ```
   java -classpath itfim-ldap.jar com.tivoli.am.fim.ldap.MigrateUSCSecretQuestion
   -backup /home/user1/Downloads/usc/backup.ldif -h localhost
   -D cn=root,dc=example,dc=com -w mercury1 -baseDn dc=example,dc=com
   -ldif /home/user1/Downloads/usc/migrate.ldif -attribute description
   -newAttribute businessCategory
   -deleteOldEntry
   ```

5. Start Tivoli Federated Identity Manager run time application.
   a. In Integrated Solutions Console, navigate to **Applications** > **Application Types** > **WebSphere enterprise applications**
   b. Select **ITFIMRuntime**.

c. Click **Start**.

Results

The secret question answers are salted and hashed in the directory.

Using more than one secret question in User Self Care:

Tivoli Federated Identity Manager version 6.2.2, fix pack 4 supports multiple secret questions. You must configure User Self Care to use this feature.

Before you begin

Tivoli Federated Identity Manager version 6.2.2, fix pack 4 supports multiple secret questions to validate the identity of users. In previous versions User Self Care used one secret question. See, "Modifying the number of secret questions used in User Self Care" on page 582 to understand the difference between the old and the new sample mapping rule.

Procedure

1. Edit the following HTML pages:
   - `enrollment.html`
   - `secretquestion.html`
   - `profile.html`

   Tivoli Federated Identity Manager versions 6.2.2, fix pack 4 provides sample template pages. Merge the required information from the sample with your existing HTML pages.

Table 145. HTML pages

Page	Pages path	Sample template pages
`enrollment.html`	`<FIM_INSTALL_DIR>/pages/<LOCALE>/usc/enrollment/enrollment.html`	`<FIM_INSTALL_DIR>/pages_template/<LOCALE>/usc/enrollment/enrollment.html`
`secretquestion.html`	`<FIM_INSTALL_DIR>/pages/<LOCALE>/usc/password/secretquestion.html`	`<FIM_INSTALL_DIR>/pages_template/<LOCALE>/usc/password/secretquestion.html`
`profile.html`	`<FIM_INSTALL_DIR>/pages/<LOCALE>/usc/profile/profile.html`	`<FIM_INSTALL_DIR>/pages_template/<LOCALE>/usc/profile/profile.html`

   See "Modifying the number of secret questions used in User Self Care" on page 582 to understand the difference between the old and the new sample pages.
2. **Publish pages to the Tivoli Federated Identity runtime environment**.
3. Merge the mapping rule. You must merge the information provided in the sample mapping rule with your existing mapping rule. The sample mapping rule in `<FIM_INSTALL_DIR>/examples/js_mappings/usc.js`.

What to do next

You must reconfigure the User Self Care federation.

Configuring new User Self Care federations

After you installTivoli Federated Identity Manager version 6.2.2, fix pack 4 you must configure new User Self Care federations to be able to use the multiple secret feature.

After you install the fix pack 4 existing User Self Care federations do not support multiple secret questions. However, if you create a User Self Care federation after you install the fix pack, that federation supports the multiple secret question function by default. The template HTML pages can accept up to three secret question answers.

Before you configure the new federation, replace the default HTML pages. Replace the default HTML pages in `<FIM_INSTALL_DIR>/pages` with the sample template pages in `<FIM_INSTALL_DIR>/pages_template`.

To configure the User Self Care federations, see *IBM Tivoli Federated Identity Manager Configuration Guide*.

The salting and hashing of secret question answers are disabled by default. To enable the salting and hashing, see "Enabling salting and hashing on secret question values."

Enabling salting and hashing on secret question values

To enhance the security of storing secret question values, you can enable salting and hashing in User Self Care.

About this task

Salting is a method encrypting inputs such passwords by adding a random string to them. This makes passwords more secure. Hashing is a method of using an algorithm to convert data to a fixed-sized value. Enabling the salting and hashing is optional. However, it is suggested that you enable this feature to enhance the security of storing the secret question answers.

Procedure

1. Log in to the Integrated Solutions Console.
2. Navigate to **Tivoli Federated Identity Manager** > **Domain Management** > **Runtime Node Management** > **Runtime Custom Properties**.
3. Change the **USC.SecretQuestion.SaltAndHash.Enabled** parameter to **true**.
4. Click **OK**.
5. Click **Load configuration changes to the Tivoli Federated Identity Manager runtime**.

Modifying the number of secret questions used in User Self Care

User Self Care supports secret questions for user validation. You can configure User Self Care to use more than one secret question to enhance the security of validating users when they forget their passwords.

Before you begin

Use these instructions to change the secret questions presented to users during enrollment and validation and if:

- You created new federations after you installed Tivoli Federated Identity Manager version 6.2.2, fix pack 4

- You have existing User Self Care federations and completed "Using more than one secret question in User Self Care" on page 581

About this task

You can change the number of secret questions used to validate users. The changes include modifications to the user interface, Security Token Service configurations, and the User Self Care mapping rule.

Note: Some special characters cannot be used as LDAP attributes. Administrators must keep enough validation in place to prevent users from entering these characters as values for any fields in the User Self Care HTML forms.

To change the number of secret questions, you must modify the following files:

Table 146. HTML pages

HTML page	Page description
`enrollment.html`	Registration form.
`secretquestion.html`	Page displayed to accept the answer to the secret question and the new password in the forgot password workflow.
`profile.html`	Edit profile form.

Procedure

1. Understand the multiple secret question-related HTML pages.
2. Modify the multiple secret question-related HTML pages.
3. Understand the multiple secret question-related mapping rule.
4. Modify the mapping rule to implement the multiple secret question feature.
5. Apply the changes to the mapping rule for multiple secret questions.
6. Understand the multiple secret question-related STS modules.
7. Modify the multiple secret question related-STS modules.
8. "Reconfiguring the User Self Care federation" on page 593

Results

- The correct number of secret question inputs are displayed. The number of inputs is the same as the new maximum secret question number.
- Users can register with the specified minimum and maximum number of secret questions.
- Users can see all the questions they answered during enrollment when they do a password recovery.
- Users can reset their passwords by answering the multiple secret questions correctly to validate their identity.

About the User Self Care multiple secret question-related HTML pages:

You must edit HTML pages to configure the User Self Care multiple secret question feature. It is important to understand how HTML pages work and how to edit the pages to meet your requirements.

Three sections are related to the secret question in the enrollment and secret question HTML pages. The `enrollment.html` and `secretquestion.html` pages contain all three sections. The `profile.html` page contains only the first two sections.

- User interface input
- Validation of required fields
- Secret question option initialization

User interface input

The user interface input consists of the HTML elements `select` and `input`. The `select` element provides the secret question selection capability. This element is the only place where the secret question string is located. The option list must be the same in both the enrollment and secret question pages.

enrollment.html code:

```
<!-- Secret Question Example Field -->
<label for="usc.form.profile.secret.question0">
    Please select a secret question and enter an answer. (required)
</label>

<br />

<select name="usc.form.profile.secret.question0"
        id="secret_question0"
        tabindex="8" >
    <option value="0">Mother's maiden name.</option>
    <option value="1">Name of town where you were born.</option>
    <option value="2">Name of first pet.</option>
</select>

<input style="background-color:#F8F8C8;"
       type="text"
       name="usc.form.profile.secret.question0.answer"
       id="secret_question_answer0"
       value=""
       size="60"
       maxlength="60"
       tabindex="9" />
<br />
```

Example process:

1. The user submits the enrollment form.
2. The `usc.form.profile.secret.question(index)` and `usc.form.profile.secret.question(index)` are passed as input parameters to Tivoli Federated Identity Manager.
3. The `usc.form.profile.secret.question(index).answer` contains only the index of the secret question, not the full sentence.
4. The `question select` and `input` elements are repeated with different index according to the maximum secret question number that user can enter.

secretquestion.html code:

```
<!-- Question 1 -->
<!-- Secret Question -->
<select name="usc.form.profile.secret.question0"
        id="secret_question0"
        disabled="disabled">
    <option value="0">Mother&#8217;s maiden name.</option>
    <option value="1">Name of town where you were born.</option>
```

```
    <option value="2">Name of first pet.</option>
</select>
<!-- Secret Question Answer Required Field -->
<input style="background-color:#F8F8C8;"
       type="text"
       name="usc.form.profile.secret.question0.answer"
       id="secret_question_answer0"
       value=""
       size="60"
       maxlength="60"
       tabindex="1" />
<input type="hidden"
       name="usc.form.profile.secret.question0.index"
       id="secret_question_hidden0"
       value="@USC_FORM_PROFILE_SECRET_QUESTION0@"
       maxlength="2" />
<br />
<!-- End question 1 -->
```

The `secretquestion.html` file contains similar code sections to `enrollment.html` but `usc.form.profile.secret.question(index).index` provides additional hidden input. There are differences between these HTML pages. The `select` element is disabled. The user cannot change the value provided because `usc.form.profile.secret.question(index)` is disabled. It is not passed as input parameter on the submit form. When the user submits the enrollment form, the secret question parameters that are passed as input parameters to Tivoli Federated Identity Manager are `usc.form.profile.secret.question(index).answer` and `usc.form.profile.secret.question(index).index`.

profile.html code:

```
<!-- Secret Question Example Field -->
<label for="usc.form.profile.secret.question0">
    Secret question (Question and answer not displayed for your security.)
</label>
<br />
<select name="usc.form.profile.secret.question0"
        disabled="true"
        id="secret_question0">
    <option value="0">Mother's maiden name.</option>
    <option value="1">Name of town where you were born.</option>
    <option value="2">Name of first pet.</option>
</select>

<input style="background-color:#F8F8C8;"
       type="text"
       disabled="true"
       name="usc.form.profile.secret.question0.answer"
       id="secret_question_answer0"
       value=""
       size="60"
       maxlength="60" />


...
Check to edit the secret question fields:
<input id="edit_secret_question"
       type="checkbox"
       name="control3"
       onclick="enableEditing(this.checked,
               document.forms[0].secret_question0,
               document.forms[0].secret_question_answer0,
               document.forms[0].secret_question1,
               document.forms[0].secret_question_answer1,
               document.forms[0].secret_question2,
               document.forms[0].secret_question_answer2)" />
```

The profile.html file contains similar code section to enrollment.html. However, it contains a check box to enable and disable the secret question-related inputs. The usc.form.profile.secret.question and usc.form.profile.secret.question.answer elements are disabled initially. If a user changes the secret question and answer, they must select the check box to enable those inputs.

The onclick handler of the control3 check box calls enableEditing to enable or disable the secret question inputs. The enableEditing function takes as many parameters as needed. The first parameter is a boolean value, and the rest of the parameters are the elements that can be enabled or disabled. If the first parameter is true the rest of the parameters is enabled, otherwise they are disabled.

Validation of required fields

Validation of required fields happens in the javascript doSubmit() function. This function is called when the user triggers the submission form. For example, when the user clicks Enroll in the enrollment.html. If the doSubmit() function succeeds, the form is submitted to Tivoli Federated Identity Manager. Otherwise, an error message is displayed to the user, and the form submission is canceled. The relevant portions are the same for both enrollment.html and secretquestions.html.

```
function doSubmit() {

    ...

    var secretQuestionAnswerTF =
            document.getElementById('secret_question_answer');

    ...

    if (!testInput(true,"Secret Question Answer (1st entry)",
            secretQuestionAnswerTF0.value.trim())){
        return true;
    }
    if (!testInput(true,"Secret Question Answer (2nd entry)",
            secretQuestionAnswerTF1.value.trim())){
        return true;
    }
    if (!testInput(false,"Secret Question Answer (3rd entry)",
            secretQuestionAnswerTF2.value.trim())){
        return true;
    }
    ...
}
```

The testInput function makes the secret question fields mandatory. In enrollment.html and profile.html, the first few questions are made mandatory and the rest are optional. The number of mandatory questions must equal the minimum number of questions that a user must answer during enrollment. For secretquestion.html, do not make any of the questions mandatory so that users can choose which question they want to answer.

For the profile.html page, the secret question inputs are validated and submitted only when they are enabled.

```
function doSubmit() {

    ...

    var secretQuestionAnswerTF0 =
```

```
            document.getElementById('secret_question_answer0');
    var secretQuestionAnswerTF1 =
            document.getElementById('secret_question_answer1');
    var secretQuestionAnswerTF2 =
            document.getElementById('secret_question_answer2');

    ...

    if (document.getElementById('edit_secret_question').checked
            == true) {
        if (!testInput(true,"Secret Question Answer (1st entry)",
                secretQuestionAnswerTF0.value.trim())){
            return true;
        }
        if (!testInput(true,"Secret Question Answer (2nd entry)",
                secretQuestionAnswerTF1.value.trim())){
            return true;
        }
        if (!testInput(false,"Secret Question Answer (3rd entry)",
                secretQuestionAnswerTF2.value.trim())){
            return true;
        }
    }
}
```

Secret question option initialization

When a submission fails because of an invalid input, the user is presented with the previous form with pre-filled fields for them to correct. The `setSecretQuestionSelect` sets the `select` element to the previous selected value.

The relevant sections are the same for both `enrollment.html` and `secretquestions.html`. `Profile.html` does not have this section for security reasons.

```
function setSecretQuestionSelect() {
    var secretQuestionValue = new Array();
    secretQuestionValue[0] = "@USC_FORM_PROFILE_SECRET_QUESTION0@";
    secretQuestionValue[1] = "@USC_FORM_PROFILE_SECRET_QUESTION1@";
    secretQuestionValue[2] = "@USC_FORM_PROFILE_SECRET_QUESTION2@";

    for (var j = 0; j < 3; j++) {
        if (secretQuestionValue[j].length > 0) {
            var secretQuestion =
                    document.getElementById('secret_question' + j);
            for (i = 0; i < secretQuestion.options.length; i++) {
                if (secretQuestion.options[i].value ==
                        secretQuestionValue[j]) {
                    secretQuestion.options[i].selected = true;
                    break;
                }
            }
        }
    }
    return true;
}
```

Tivoli Federated Identity Manager replaces the `@USC_FORM_PROFILE_SECRET_QUESTION(index)@` value, which is a macro, with the outgoing `usc.form.profile.secret.question(index)` parameter. For example:

1. `enrollment.html` is requested for the first time.
2. The value of the outgoing parameter `usc.form.profile.secret.question(index)` value is empty.

3. `@USC_FORM_PROFILE_SECRET_QUESTION(index)` is replaced with an empty string.
4. The user submits the enrollment form.
5. Tivoli Federated Identity Manager copies the content of the input parameter `usc.form.profile.secret.question(index)`. It contains the secret question index for the outgoing parameter `usc.form.profile.secret.question(index)`.
6. If the enrollment fails because of wrong input, Tivoli Federated Identity Manager displays the `enrolment.html` again. It adds error messages. It also replaces `@USC_FORM_PROFILE_SECRET_QUESTION(index)@` with the value for the `usc.form.profile.secret.question(index).` output parameter.

Modifying the User Self Care multiple secret question-related HTML pages:

When you change the maximum and minimum number of secret questions, update the input sections for each of the three pages.

Before you begin

To understand the prerequisites and which HTML pages are related to multiple secret questions see, "Modifying the number of secret questions used in User Self Care" on page 582. Modify the multiple secret question-related HTML pages to edit the number of secret questions presented to users during registration and profile updates.

About this task

Edit specific sections in the HTML pages to modify the number of secret questions.

Procedure

1. Open the related HTML pages with a text editor.
2. Edit the relevant sections of the HTML pages. Repeat the number of input sections the same number of times as the maximum number of secret questions.
3. Update the index by starting with 0.
4. Make the first few questions for `enrollment.html` and `profile.html` required values. The number of required questions must be the same as the minimum number of secret questions required.
5. Update the call to: `enableEditing` to reflect the new number of secret questions.

Example

```
<input id="secret_enabled"
       type="checkbox"
       name="control3"
       onclick="enableEditing(this.checked,
               document.forms[0].secret_question0,
               document.forms[0].secret_question_answer0,
               document.forms[0].secret_question1,
               document.forms[0].secret_question_answer1,
               document.forms[0].secret_question2,
               document.forms[0].secret_question_answer2);" />
```

What to do next

Publish pages to the Tivoli Federated Identity runtime environment

The mapping rule for multiple secret questions:

The mapping rule is implemented in the User Self Care multiple secret question feature to transform user inputs from one form to another.

Mapping rules are the rules that are applied by the Security Token Service (STS) module Default Map Module. An example of this rule is transforming user input to another form. In the User Self Care secret question, the mapping rule can be used to concatenate the secret question index and answer to a single string.

The STS module `Default Map Module` is used in both the `Default Chain uscCreateAccount` and `Default Chain uscForgetPassword` STS chains to parse secret question input and output.

The default mapping rule is in `<FIM_INSTALL_DIR>\examples\js_mappings\usc.js`. The mapping rules define how secret questions are stored internally. By default, the secret question is stored in the `businessCategory` LDAP attribute. It is stored in the following format:
`'<secretQuestionIndex>::{SSHA2}<salt><hashedSecretQuestionAnswer>'`.

In the default mapping rule file, two sections relate to secret question:

Incoming request mapping section

This section handles the incoming input parameter from the user. It retrieves the secret question index and secret question answer from the form input and maps it into the STS internal attribute for further processing.

The LDAP attribute stores the secret question and answer uses the `helper.setSTSInternalSecretQuestionAttr` function.

```
helper.setSTSInternalSecretQuestionAttr("businessCategory");

// Set maximum secret question allowed
var MAX_SECRET_QUESTIONS = 3;
var secretQuestions = java.lang.reflect.Array.newInstance(
        java.lang.String, MAX_SECRET_QUESTIONS);
var secretQuestionsAnswer = java.lang.reflect.Array.newInstance(
        java.lang.String, MAX_SECRET_QUESTIONS);
for (var i = 0; i < MAX_SECRET_QUESTIONS; i++) {
    var secretQuestionInput = helper.getUserInputAttributeValues(
            "usc.form.profile.secret.question" + i);
    if (!supplied(secretQuestionInput)) {
        secretQuestionInput = helper.getUserInputAttributeValues(
                "usc.form.profile.secret.question" + i + ".index");
    }
    var secretQuestionAnswerInput = helper.getUserInputAttributeValues(
            "usc.form.profile.secret.question" + i +".answer);
    if (supplied(secretQuestionInput)
            && supplied(secretQuestionAnswerInput)) {
        secretQuestions[i] = secretQuestionInput[0];
        secretQuestionsAnswer[i] = secretQuestionAnswerInput[0];
    }
}
// Set to STS internal attribute
helper.setSTSInternalAttribute(
        USCCAConstants.USC_STS_INTERNAL_SECRET_QUESTIONS,
        secretQuestions);
helper.setSTSInternalAttribute(
        USCCAConstants.USC_STS_INTERNAL_SECRET_QUESTIONS_ANSWER,
        secretQuestionsAnswer);
// Set input to normalize and sanitize STS
var normalizeSanitizeInput =
```

```
        java.lang.reflect.Array.newInstance(java.lang.String, 2);
normalizeSanitizeInput[0] =
        USCCAConstants.USC_STS_INTERNAL_SECRET_QUESTIONS;
normalizeSanitizeInput[1] =
        USCCAConstants.USC_STS_INTERNAL_SECRET_QUESTIONS_ANSWER;
helper.setSTSInternalAttribute(
        USCCAConstants.USC_STS_INTERNAL_NORMALIZE_AND_SANITIZE_INPUT,
        normalizeSanitizeInput);
// Set input to salt and hash STS
helper.setSTSInternalAttribute(
        USCCAConstants.USC_STS_INTERNAL_SALT_AND_HASH_INPUT,
        USCCAConstants.USC_STS_INTERNAL_SECRET_QUESTIONS_ANSWER);
```

Outgoing request mapping section

This section handles the outgoing parameter that replaces the macros in the HTML pages before it sends the pages to the user. It retrieves the stored secret question from the STS output `USC_FORM_SECRET_QUESTION` attribute. It then sets the STS output `USC_FORM_SECRET_QUESTION(Index)` attribute.

```
 //
Get the secret question from the registry
   var secretQuestionRA = helper.getSTSOutputAttributeValues(
           USCCAConstants.USC_FORM_SECRET_QUESTION);
   if (secretQuestionRA != null) {
       for (var i = 0; i < secretQuestionRA.length; i++) {
           if (secretQuestionRA[i] != null) {
               helper.setSTSOutputAttribute(
                       "usc.form.profile.secret.question" + i,
                       secretQuestionRA[i]);
           }
       }
   }
```

Modifying the mapping rule for User Self Care multiple secret questions:

You must modify the mapping rules if you change the maximum secret question number.

Before you begin

To understand how the mapping rule is implemented in the User Self Care multiple secret question feature, see "The mapping rule for multiple secret questions" on page 589.

Procedure

1. Back up the mapping rule file.
2. Open the file.
3. Change the maximum secret question number. For example:
   ```
   var MAX_SECRET_QUESTIONS = 3;
   ```
4. Save the modified file.

What to do next

Apply the changes to the mapping rule.

Applying changes to the mapping rule for multiple secret questions:

Apply the modified mapping rule to the Default map module of three STS chains to use the multiple secret question feature.

Before you begin

Use the mapping rule you modified in "Modifying the mapping rule for User Self Care multiple secret questions" on page 590.

About this task

For the mapping rule to implement the changes, it must be applied to these chains:

- Default Chain **uscCreateAccount**
- Default Chain **uscForgottenPassword** (both instances of this chain)
- Default Chain **uscProfileManagement** (both instances of this chain)

Procedure

1. Log in to the Integrated Solutions Console.
2. Navigate to **Tivoli Federated Identity Manager** > **Runtime Node Management**.
3. Click **Runtime Custom Properties**.
4. Set property **STS.showUSCChains** to `true`.
5. Click **OK**.
6. Logout from the Integrated Solutions Console.
7. Login to the Integrated Solutions Console.
8. Navigate to **Tivoli Federated Identity Manager** > **Configure Trust Service** > **Trust Service Chains**.
9. For each of the chains do the following steps:
   a. Select the check box for the corresponding chain.
   b. Click **Properties**.
   c. Click **Modify Rule**.
   d. Click **Browse** to select the modified mapping rule.
   e. Click **Import File**
   f. Click **OK**.
   g. Repeat the steps for the second `Default Map Module` in the chain.
   h. Click **Load configuration changes to the Tivoli Federated Identity Manager runtime**.

User Self Care multiple secret questions response file parameters:

Use the multiple secret questions response file parameters to configure the response file.

There are three configurable parameters in the response file related to multiple secret questions. They are the following:

SecretQuestionRequiredForValidationNumber
Specifies the number of secret questions a user must answer correctly to validate their identity. During password recovery, users must provide correct answers to the secret questions. The number of questions that they must answer correctly is dependent on this parameter.

Note: You can also find this parameter in the response file, with a default value of 1.

SecretQuestionMaximumNumber
Specifies the maximum number of secret questions to which a user can

provide answers during enrollment. It dictates the number of secret questions that users can provide answers to when they forget their password or during profile update.

Note: You can also find this parameter in the response file, with a default value of 3.

SecretQuestionMinimumNumber
Specifies the minimum number of required secret questions to which a user must provide answers during enrollment. It dictates the number of secret questions that users must answer when they forget their password or during profile update.

Note: You can also find this parameter in the response file, with a default value of 2.

Modifying the STS module configurations for multiple secret questions:

You must modify specific STS modules to utilize the multiple secret question feature of User Self Care.

About this task

The Default USC Secret Question Store STS Module is in `uscAccountCreate` and `uscProfileManagement` chain.

The Default USC Secret Question STS Module is in `uscForgottenPassword` chain.

Modify the Default USC Secret Question STS Module and Default USC Secret Question Store STS Module in the following STS chains:

- Default Chain **uscCreateAccount**
- Default Chain **uscForgottenPassword** (both instances of this chain)
- Default Chain **uscProfileManagement** (both instances of this chain)

Procedure

1. Log in to the Integrated Solutions Console.
2. Navigate to **Tivoli Federated Identity Manager** > **Runtime Node Management**.
3. Click **Runtime Custom Properties**.
4. Set the property **STS.showUSCChains** to `true`.
5. Click **OK**.
6. Log out from the Integrated Solutions Console.
7. Log in to the Integrated Solutions Console.
8. Navigate to **Tivoli Federated Identity Manager** > **Configure Trust Service** > **Trust Service Chains**.
9. For each of the chains do the following steps:
   a. Select the check box for the corresponding chain.
      - Default Chain **uscCreateAccount**
      - Default Chain **uscForgottenPassword** (both instances of this chain)
      - Default Chain **uscProfileManagement** (both instances of this chain)
   b. Click **Properties**.

c. Under **Trust Service Chain Modules**, select the corresponding module and mode. You need to modify the Default USC Secret Question STS Module and Default USC Secret Question Store STS Module.
   - The Default Chain **uscCreateAccount** contains the Default USC Secret Question Store STS Module.
   - The Default Chain **uscForgottenPassword** (both instances of this chain) contains the Default USC Secret Question STS Module.
   - The Default Chain **uscProfileManagement** (both instances of this chain) contains the Default USC Secret Question Store STS Module.

d. Click **Properties**.

e. Modify the corresponding parameter values.

   Default USC Secret Question STS Module

   SecretQuestionRequiredForValidationNumber - modify the parameter value to specify the number of secret questions a user must answer correctly to validate their identity.

   Default USC Secret Question Store STS Module

   - **SecretQuestionMaximumNumber** - modify the parameter value to specify the maximum number of secret questions to which a user can provide answers to during enrollment.
   - **SecretQuestionMinimumNumber** - modify the parameter value to specify the minimum number of required secret questions to which a user must provide answers during enrollment.

f. Click **OK**.

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

Reconfiguring the User Self Care federation:

Reconfigure the User Self Care federation to use a modified response file.

Procedure

1. Export the current User Self Care federation configuration to a response file by using the following command in wsadmin:

   Note: Enter the following syntax on one line.

   ```
   $AdminTask manageItfimUserSelfCare {-operation createResponseFile
   -fileId <filepath> -fimDomainName <domainName> -federationName <federationName>}
   ```

2. Modify the following parameters:
   - **SecretQuestionMinimumNumber**
   - **SecretQuestionMaximumNumber**
   - **SecretQuestionRequiredForValidationNumber**

   See "User Self Care multiple secret questions response file parameters" on page 591 for more details.

3. Modify the parameter **AttributeMappingFilename** to point to the modified mapping rule you edited in "Modifying the mapping rule for User Self Care multiple secret questions" on page 590.
4. Save the file.
5. Unconfigure the User Self Care federation. See Unconfiguring User Self Care.
6. Configure the User Self Care federation with the modified response file. See Using a response file to configure User Self Care.

Custom attribute definition

Define your own custom attribute in User Self Care so that these attributes are collected by User Self Care during enrollment.

User information that is collected during enrollment is stored in the supported enterprise directories. It is then mapped to LDAP attributes on the user entry in a configured enterprise directory.

The default user enrollment provides a limited number of information fields that are mapped to LDAP attributes. You must add fields that are not included. You can use the user profile management to add fields that can be mapped to an existing LDAP attribute in an enterprise directory.

By default, the following fields are provided:

- User ID
- Email Address Password
- Mobile Phone Number
- Secret Question

You must do a few steps if you want to add a field like **Company**. The value of the new field is saved into the mapped LDAP attribute in the configured enterprise directory.

For more information about how you can use a non-default LDAP attribute, see "Creating an attribute for a new custom field in User Self Care" on page 597.

Complete the following tasks to define custom attributes in User Self Care:

1. Modify the HTML file so that the new field displays in the user information form.
2. Modify the Java script so that the new field is mapped to LDAP attribute.
3. Run the wsadmin commands so that your changes are reflected in IBM Tivoli Federated Identity Manager.

After you complete the steps, a user can enroll.

Modifying the HTML file to define a custom attribute

Modify the `enrollment.html` and `profile.html` to define a custom attribute in User Self Care.

About this task

You can modify the following HTML files:

Table 147. HTML files

HTML page	Page description
`enrollment.html`	Registration form.
`profile.html`	Form that updates the profile information of the user.

Procedure

1. Open each of the HTML files with a text editor.

Note: If the HTML file is not in English, use a text editor that supports UTF-8 encoding. Otherwise, some letters might look wrong and saving the file might cause unreadable content.

2. Add the following the lines between the `<form...> ... </form>` tags.

```
<label for="usc.form.profile.company"> Company </label>
<input id="company" type="text" tabindex="8" maxlength="60" size="60"
value="" name="usc.form.profile.company" style="
background-color: rgb(255, 255, 255);" />
```

3. Change the `Company` text to a custom field name. You can customize any attribute, but you must ensure that you use the same identification values.
4. Save the modified files in the same directory as the original files.
5. Log on to the **Integrated Solutions Console**.
6. Select **Tivoli Federated Identity Manager** > **Domain Management** > **Runtime Node Management**.
7. Click **Publish**.
8. Click **Load configuration changes to the Tivoli Federated Identity Manager runtime**.

Results

The new field displays in the account enrollment page.

What to do next

Modify the JavaScript mapping file.

Modifying the JavaScript mapping file

Modify the JavaScript mapping file so that the new field is mapped to LDAP attribute.

About this task

The default JavaScript mapping file is in `<FIMInstallationPath>/examples/js_mappings/usc.js`. This file defines the mapping behavior. Add the new mapping rules for the new custom field in the JavaScript file. If you use another JavaScript file for the mapping rules, ensure that you are accessing the correct file.

Procedure

1. Back up the JavaScript mapping rule file.
2. Open the JavaScript mapping rule file with a text editor.
3. Add the following lines and change the value of `Company` to the field name that you want to add.

```
////////////////////////////////
// INCOMING REQUEST MAPPING
////////////////////////////////
var company = helper.getUserInputAttributeValues("usc.form.profile.company");
if (supplied(company)) {
    // map 'company' to 'company' LDAP attribute
    // change "company" to "organizationName" when using Tivoli Directory Server
    // or as mentioned you can use some other LDAP attribute
    helper.setSTSInternalRegistryInputAttribute("company", company);
}

////////////////////////////////
// OUTGOING RESPONSE MAPPING
////////////////////////////////
```

```
// Get the 'company' from the registry.
 // change "company" to "organizationName" when TDS
      // or as mentioned you can use some other LDAP attribute
var companyRA = helper.getSTSInternalRegistryOutputAttributeValues("company");
if (supplied(companyRA)) {
    // Stick it in the output attribute
    helper.setSTSOutputAttribute("usc.form.profile.company", companyRA);
}
```

4. Save the changes.
5. Log on to the **Integrated Solutions Console**.
6. Select **Tivoli Federated Idenity Manager** > **Configure Trust Service** > **Trust Service Chains**.
7. Open the **Properties** pages for the following files:
   - `USC Chain for uscCreateAccount`
   - `USC Chain for uscProfileManagement` (two instances)

   Note: Depending on the chain that you are modifying, you must modify multiple chains for the mapping rule to be implemented.
8. Select the **USC Chain for uscCreateAccount** and **USC Chain for uscProfileManagement** check boxes.
9. Complete the following steps for all three items.
   - `USC Chain for uscCreateAccount`
   - `USC Chain for uscProfileManagement` (two instances)

   a. Click **Properties**.
   b. Under **Trust Service Chain Modules**, select **Default map module**.
   c. Click **Properties**.
   d. Click **Modify Rule**.
   e. Click **Browse** to select the modified mapping rule.
   f. Click **Import File**.
   g. Click **OK**.
   h. Click **Load configuration changes to the Tivoli Federated Identity Manager runtime**.
   i. Repeat the steps for the following items:
      - The chain mappings with which the policy is associated. Some policies might affect multiple chain mappings.
      - The chains with the same name.
      - The default map modules in a chain.

What to do next

Run the wsadmin commands.

Running wsadmin commands to implement the custom attribute

Run the **wsadmin** commands so that the custom attribute is reflected in Tivoli Federated Identity Manager User Self Care.

Procedure

1. Log in to the wsadmin console with the following commands:

   Windows: **<WASInstallationPath>\WebSphere\AppServer\profiles\ <profileName> \bin\wsadmin.bat\-username username \-password password**

Linux:**/<WASInstallationPath>/WebSphere/AppServer/profiles/<profileName>/bin/wsadmin.sh\-username username \-password password**

2. In the wsadmin console, run the following **$AdminTask** commands.
   a. **$AdminTask addIdMgrPropertyToEntityTypes {-name company -dataType string -entityTypeNames PersonAccount}**

      This command supports the new custom attribute for Federated Repositories Component or Virtual Member Manager of the WebSphere Application Server. By default, the Virtual Member Manager supports only a few basic attributes. To use a new attribute, you must extend the internal entity schema in the Virtual Member Manager. This command adds support for the **company** LDAP attribute in Virtual Member Manager. You can change **company** to any LDAP attribute.

   b. **$AdminTask addIdMgrPropertyToEntityTypes {-name company -dataType string -entityTypeNames PersonAccount -isMultiValued true}**

      Some attributes can have multiple values. You must use attributes that have multiple values. If the attribute must have multiple values, then supply the **isMultiValued** parameter with the value `true`.
3. Stop the WebSphere Application Server.
4. Restart the WebSphere Application Server. Restarting the server ensures that Virtual Member Manager settings are applied.

Results

When you click the link in the email, the user attribute is created in the configured enterprise directory with the new attribute.

What to do next

After you restart the WebSphere Application Server, go to the enrollment page and verify that the new field is added. After you click enroll, an email is sent to the email address that you provided.

Creating an attribute for a new custom field in User Self Care

Create an attribute for a custom field in User Self Care that maps to an LDAP attribute that you want to use.

Before you begin

Ensure that Tivoli Federated Identity Manager and WebSphere Application Server are configured for the Virtual Member Manager. See the Tivoli Federated Identity Manager Configuration Guide for more details.

About this task

Example 1

Create an attribute for a custom field such as such as `badge number` that maps to an attribute such as `badgeNo` in LDAP.

Example 2

You want to add the IBM Tivoli Directory Server attribute `ibm-ismanager`. It cannot be added to an LDAP entity by default. You must first add the `ibm-itdPerson` attribute as one of the `objectClasses` of the LDAP entity. Then, you can add the `ibm-ismanager` to an LDAP entity with a `true` or `false` value.

No changes are required for scenarios where you want to add a custom field that already has an available LDAP attribute. For example, you can add the custom field `Given Name` and map it to the `givenName` LDAP attribute in the IBM Tivoli Directory Server.

Procedure

1. Log on to the **Integrated Solutions Console**.
2. Select **Security** > **Global Security**.
3. Under **User account repository**, click **Configure**.
4. Select **Federated repositories**.
5. Click **Manage Repositories**. A list of repositories is displayed. Complete the following steps for each repository:
   a. Click the name of the repository. Details of the repository are displayed.
   b. Under **Additional Properties**, click **LDAP entity types**.
   c. Click **PersonAccount**.
   d. In the **PersonAccount** page, there is an **Object classes** field. An existing value such as `user` exists if the repository type is Active Directory or IBM Tivoli Directory Server.
   e. In the **Object classes** field, append the new `objectClass` that defines the attribute you want to use. For example, if you want to use the `ibm-ismanager` attribute, the object class that defines it is `ibm-itdPerson`. Append the `ibm-ismanager` attribute into the field with a semicolon. For example, `netOrgPerson;ibm-itdPerson`.
   f. Click **OK**.
   g. Click **Save changes to the master configuration**.
6. Optional: If you configured and are using a Tivoli Access Manager adapter, do the following steps:
   a. Open the file `tamVMMAdapter.properties`.

      ```
      <WASInstallation>\profiles\<tfimprofile>\config\itfim\
        tamVMMAdapter.properties
      ```

      Note: The file name might be different. The name that is shown in this example is the name that is set during the Tivoli Access Manager adapter configuration.
   b. Add the following lines.

      ```
      ldap.user-objectclass=Person;ePerson;inetOrgPerson,organizationalPerson;
      customObjectClass ldap.user-self-care-objectclass=customObjectClass
      ```
7. Stop WebSphere Application Server.
8. Start WebSphere Application Server.

Results

The Virtual Member Manager creates the attribute.

User Self Care session information storage

User Self Care flows store user attributes into the session of a user. Those attributes can then be retrieved by other modules.

An option in the User Self Care response file indicates which flows can store information. You can configure the flows that normally send an email to store information in the user session.

You can configure the following flows to store User Self Care information in the user session:

- User enrollment
- Forgotten password
- Forgotten user ID

To store information in a session, configure the `FlowsWithSessionStorageAndNoEmailDelivery` parameter in the response file.

This parameter is a multi-valued, string parameter and defaults to no flows. It looks for the following values in the response file:

```
<void method="put">
   <string>FlowsWithSessionStorageAndNoEmailDelivery</string>
   <object class="java.util.ArrayList">
      <void method="add">
        <string>USC_ENROLLMENT</string>
      </void>
      <void method="add">
        <string>USC_FORGOT_PASSWORD</string>
      </void>
      <void method="add">
        <string>USC_FORGOT_ID</string>
      </void>
   </object>
</void>
```

Where:

- User enrollment is `USC_ENROLLMENT`
- Forgotten password is `USC_FORGOT_PASSWORD`
- Forgotten user ID is `USC_FORGOT_ID`

The information that is stored in the session is fetched from the incoming User Self Care Security Token Service (STS) chain and placed into a map. The map

- Consists of String keys and String[] values
- Is serialized, converted to a byte[], and Base64 encoded
- Is then stored in the session of the user
- Can be retrieved in a mapping rule by using `IDMappingExtUtils.getSPSSessionData()` function by using the key `usc_attributes_session_key`

The map contains various attributes that are related to the user and the flow. Since attributes are multi-valued, they are stored as an array of String objects in the map. Each flow persists a different set of attributes.

- The user enrollment flow

Table 148. List of attributes that are stored during a user enrollment flow

Key	Description
`usc.flow.id`	The flow that is being run.
`usc.form.userid`	The user who performs the enrollment.
`usc.confirmation.code`	The confirmation code of the enrollment flow.
`usc.validation.url`	The validation URL of the enrollment flow.

Note: The STSUU attributes from `usc.user.input.type` is also contained in the map by using their STSUU attribute name as the key.

- The forgotten password flow

Table 149. List of attributes that are stored during a forgotten password flow

Key	Description
usc.flow.id	The flow that is being run.
usc.user.id	The user who performs the enrollment.
usc.confirmation.code	The confirmation code of the enrollment flow.
usc.validation.url	The validation URL of the enrollment flow.

Note: The STSUU attributes from usc.sts.internal.registry.output.type is also contained in the map by using their STSUU attribute name as the key.

- The forgotten user ID flow

Table 150. List of attributes that are stored during a forgotten user ID flow

Key	Description
usc.user.id	The list of user ID associated with the forgotten account.
usc.flow.id	The flow that is being run.

Note: The STSUU attributes from usc.sts.output.type is also contained in the map by using their STSUU attribute name as the key.

The following JavaScript example shows how a mapping rule can retrieve the stored User Self Care data from the session of the user. The attributes are accessed by retrieving the serialized map from the session with the getSPSSessionData('usc_attribute_session_key') method. This object is deserialized into a map, and individual attributes can be accessed from the map by using the appropriate key.

To access the SPSSessionData, you must import the com.tivoli.am.fim.trustserver.sts.utilities package. To deserialize, you can import the com.ibm.ws.util package. The following mapping rule is an example.

```
//required import statements
importPackage(Packages.com.tivoli.am.fim.trustserver.sts.utilities);
importPackage(Packages.com.ibm.ws.util);

//
// Returns a String->String[] attribute map that was set by USC using the
//     FlowsWithSessionStorageAndNoEmailDelivery parameter.
//
function getUSCAttributeMap() {
    var dmapKey0 = "usc_attributes_session_key";
    var attrMap = null;
    var serializedAttributes = IDMappingExtUtils.getSPSSessionData(dmapKey0);
    if (serializedAttributes != null) {
        IDMappingExtUtils.traceString("Serialized attributes: " + serializedAttributes);
        // deserialize then add to AttributeList with canned "type"
        var bais = new java.io.ByteArrayInputStream((Base64.decode(serializedAttributes)));
        var ois = new java.io.ObjectInputStream(bais);
        attrMap = ois.readObject();
    } else {
        IDMappingExtUtils.traceString("No session attributes are persisted.");
    }
    return attrMap;
}

//
// Pull the attributes from the session stored by USC
//
var flowId = null;
var userId = null;
var confirmationCode = null;
var validationUrl = null;

// get the attribute map
var uscAttributeMap = getUSCAttributeMap();
if (uscAttributeMap != null) {
    // All values are String[] and you must use index operator.
    flowId = uscAttributeMap.get("usc.flow.id")[0];
    userId = uscAttributeMap.get("usc.user.id")[0];
    confirmationCode = uscAttributeMap.get("usc.confirmation.code")[0];
```

```
    validationUrl = uscAttributeMap.get("usc.validation.url")[0];
    IDMappingExtUtils.traceString("Attributes retrieved-> flowId: " + flowId + " userId: " + userId +
        " confirmation code: " + confirmationCode + " validationUrl: " + validationUrl);
}
```

Customizing the User Self Care HTML pages

Customize the User Self Care interface by adding CSS files, background images and by using User Self Care provided macros.

Before you begin

Administrators who configure User Self Care must have knowledge about the following products and concepts:

- WebSphere® Application Server, including the `wsadmin` administration interface
- Tivoli Federated Identity Manager Secure Token Service modules and trust chains
- Tivoli Directory Server LDAP or other supported LDAP
- Knowledge of CSS and HTML

About this task

You need a web server for example, IBM HTTP Server, to host the external files like CSS files and image files.

You can customize the HTML pages with User Self Care macros and Cascading Style Sheets (CSS). For more details, see:

- "User Self Care macros"
- "About User Self Care CSS" on page 606

Procedure

1. Log on to Integrated Solution Console.
2. Take one of the following actions:
   - Use macros to format User Self Care. See "Formatting User Self Care HTMLs by using macros" on page 607 for more details.
   - Use CSS to format User Self Care. See "Formatting User Self Care HTMLs by using Cascading Style Sheets" on page 609 for more details.
3. Test the changes in the HTML files.

User Self Care macros

Use the User Self Care macros to add code or values to the User Self Care HTML pages at run time.

The macros are replaced with their values at run time when the user accesses User Self Care. Not all macros are populated at the same time. The value of the macros depends on the user operation and the current state of the operation.

The following information is provided:

- Which macros are available in the default setup of User Self Care.
- The operation and the state of the operation of those macros.
- The description of the macro with sample values for User Self Care. WebSphere is the Point Of Contact server.

The template has the following replacement macros:

@ACTION@

This macro is replaced with the URL to which the form data is sent on submission. It is also the URL that is initially requested. For example, in the enrollment form, the form action looks like:

```
<form action=" (ACTION)">
```

In page load, this action is replaced with

```
<form action="https://company.com:9443/sps/USCFederation/
usc/self/account/create">
```

This macro is applicable to all workflows.

@VALIDATION_URL@

This macro is replaced with the validation URL. Whenever the system requires the user to send an email verification, VALIDATION_URL is replaced with the URL to be sent in the email. At run time, the system sends an email that contains the URL appended with a unique confirmation code parameter.

Enroll User, Value:

Note: Enter the following syntax on one line.

```
https://<Host_Name>:<Port>/sps/<USC_Federation>/usc/self/
account/create/validate
```

Forgot user ID, Value:

Note: Enter the following syntax on one line.

```
https://<Host_NameName>:<Port>/sps/<USC_Federation>/usc/self/account/recover
/password/validate
```

@USC_SEARCH_USERID_URI@

In the Enrollment page, this macro is the URL to which the request is sent when the user clicks **user ID Available?**

Enroll User, Value:

```
https://<Server_Name>:<Port>/sps/Federation>/usc/global/userid/search
```

@DETAIL@

This macro contains the message, if any, in the response.

Example

If the user tries to access the Validation URL without any validation data in the URL, the value for DETAIL looks like:

```
FBTUSC0>>E
```

The enrollment validation data must be supplied.This macro is applicable to all workflows.

@EXCEPTION_STACK@

This macro is replaced with the required HTML source that supports the `Captcha` demonstration if it is configured.

Example

This macro is replaced with:

```
<label for="demo_captcha">
  Please enter the verification word(s) shown below (required)
  </label>
  <br />
```

```
<img src="http://myserver/public/captcha_test/hello.jpg" border="0" />
<br />
<input type="hidden"
name="usc.demo.captcha.challenge.field"
id="usc.demo.captcha.challenge.field"
value="http://myserver/public/captcha_test/hello.jpg" />
<input style="background-color:#F8F8C8;"
type="text"
name="usc.demo.captcha.response.field"
id="usc.demo.captcha.response.field" />
```

This macro is applicable to all workflows.

@USC_STS_CAPTCHA_HTML_STRING@

This macro is replaced with the required HTML source that supports the Captcha demonstration if it is configured.

Example

This macro is replaced with:

```
<label for="demo_captcha">
  Please enter the verification word(s) shown below (required)
  </label>
  <br />
  <img src="http://myserver/public/captcha_test/hello.jpg" border="0" />
  <br />
  <input type="hidden"
  name="usc.demo.captcha.challenge.field"
  id="usc.demo.captcha.challenge.field"
  value="http://myserver/public/captcha_test/hello.jpg" />
  <input style="background-color:#F8F8C8;"
  type="text"
  name="usc.demo.captcha.response.field"
  id="usc.demo.captcha.response.field" />
```

This macro is applicable in the enroll user workflow.

@USC_FORM_USERID@

This macro is replaced with either the user ID of the user with a valid session or the user ID provided in the previous request.

Example

In the change password workflow, USC_FORM_USERID is replaced with the authenticated user ID after the user authenticates.This macro is applicable to all workflows.

@USC_USERAGENT_IPADDR@

This macro contains the IP address from where the previous request is received.

This macro is applicable to all workflows.

@USC_USERAGENT_HOSTNAME@

This macro contains the host name from where the previous request is received.

This macro is applicable to all workflows.

@USC_USERAGENT_TRANSPORT@

This macro contains the HTTP transport protocol that is used for the previous request.

This macro is applicable to all workflows.

@USC_USERAGENT_METHOD@

This macro is replaced with the HTML submission method that fetches this form. The value for this macro can be `GET` or `POST`.

This macro is applicable to all workflows.

@USC_USERAGENT_URI@

This macro is replaced with the HTML submission method that fetches this form. The value for this macro can be `GET` or `POST`.

This macro is applicable to all workflows.

@USC_USERAGENT_QUERY@

This macro is replaced with the query string in the previous request.

The following example shows the structure when the user accesses the URL in the email for reset password:

Note: Enter the following syntax on one line.

```
https://<server>:<port>/sps/<USC federation>/usc/self/account/recover/
password/validate?usc.confirmation.code=<Random characters>
```

After you successfully reset the password, `USC_USERAGENT_QUERY` has the following value:

```
usc.confirmation.code=<Random characters>
```

This macro is available during:

- User enrollment.
- User password reset.

@USC_USERAGENT_LOCALES@

This macro contains the locale of the browser from where the previous request is received.

Example

When the request is sent from an English locale, the value for this macro is: `en_US`.This macro is applicable to all workflows.

@USC_FORM_PASSWORD@

When the user changes a password, this macro is replaced with the value provided in the previous request for the **Old password** field.

This macro is applicable in the change password workflow.

@USC_FORM_PASSWORD_NEW@

This macro is replaced with the value of the **Password** field that is provided in the previous request.

Example

When the user fills out the password value of in the **Enroll User** form and clicks **Enroll**, the validation page displays `USC_FORM_PASSWORD_NEW_CONFIRM` with the value of the password field that is provided by the user.

This macro is available in the following workflows:

- Enroll user workflow.
- Change password workflow.
- Forgot password workflow.

@USC_FORM_PASSWORD_NEW_CONFIRM@

This macro is replaced with the value of the **Confirm password** field that is provided in the previous request.

Example

When the user fills out the Confirm password value in the **Enroll user** form and clicks **Enroll**, the validation page displays `USC_FORM_PASSWORD_NEW_CONFIRM` with the value of the confirm password field.This macro is available in the following workflows:

- Enroll user workflow.
- Change password workflow.
- Forgot password workflow.

@USC_FORM_EMAIL_ADDRESS@

This macro is replaced with the **email address** value provided by the user in the previous form.

Example

When the user fills out the **email address** value in the **Enroll User** form and clicks **Enroll**, the validation page displays `USC_FORM_EMAIL_ADDRESS` with value of the **Email address** field that is provided in the previous form.This macro is available in the following workflows:

- Enroll user workflow.
- Forgot password workflow.

This macro is available after a user answers the secret question correctly.

@USC_FORM_EMAIL_ADDRESS_CONFIRM@

This macro is replaced with the **Confirm email address** value that is provided by the user in the previous form.

Example

When the user fills out the Confirm email address value in the **Enroll User** form and clicks **Enroll**, the validation page displays `USC_FORM_EMAIL_ADDRESS_CONFIRM` with the value of the **Confirm email address** field that is provided in the previous form.

This macro is available in the following workflows:

- Enroll user workflow.
- Forgot password workflow.

This macro is available after a user answers the secret question correctly.

@USC_FORM_USERID_AVAILABLE@

When the user clicks **Is userID Available?** from the **Enroll** page, the response contains `USC_FORM_USERID_AVAILABLE` set to true or false, depending on whether the user ID is available.

@USC_FORM_SECRET_QUESTION@

This macro is replaced with the secret question selected by the user in the previous request.

Example

When the user selects a secret question in the **Enroll User** form and clicks **Enroll**, the `USC_FORM_SECRET_QUESTION` contains the value of the secret question that is selected in the previous request if the form is returned because of some error.

This macro is available in the enroll user workflow.

@USC_FORM_SECRET_QUESTION_ANSWER@

This macro is replaced with the secret question answer selected by the user in the previous request.

Example

When the user provides a secret question answer in the **Enroll User** form and clicks **Enroll**, the `USC_FORM_SECRET_QUESTION_ANSWER` contains the value of secret question answer that is provided in the previous request if the form is returned because of some error.

This macro is available in the enroll user workflow.

About User Self Care CSS

The Cascading Style Sheet (CSS) is a style sheet language that describes the look and format of an HTML document.

You can format the User Self Care HTML pages with a Cascading Style Sheet. Cascading Style Sheets can:

- Improve content accessibility.
- Provide flexibility and control in the specification of presentation characteristics.
- Enable multiple pages to share formatting.
- Reduce complexity and repetition in the structural content.

You need a web server (HTTP Server) to host the CSS pages. This document provides a sample CSS file, image file, and User Self Care HTML pages for English only. The HTMLs are modified to use the CSS and the image files. Deploying the HTML files modifies the behavior of English and does not affect User Self Care pages in other languages.

Examples

File	Description

enrollment.html	In this file, the CSS file is linked at the beginning of the document.
	`<link href="../../css/usc.css" rel="stylesheet">`
	
	In your deployment, replace `../../css/usc.css` with the location of the CSS file on the web server.
	`<link href="" https://company.com/ css/usc.css"" rel="stylesheet">`

	This entry displays the image at the top of the HTML pages:
	`<div class="header"> <a href="http: //ibm.com" title="Powered by IBM"><img src="@USC_HTTP_SERVER_URL@/images/ IBMlogo.png" alt="IBM"/></a> </div>`
	
	At run time, you can replace this entry with:
	`<div class="header"> <a href="http: //ibm.com" title="Powered by IBM"><img src="https://company.com/images/ IBMlogo.png" alt="IBM"/></a> </div>`

usc.css	`cssname` is an HTML element name to which the attributes are defined under `{...}`.
	
	Every attribute is a key value pair. In this definition, the HTML background color is color code 336699. This style is applied to all the HTMLs that reference the CSS file.
	`html { ; background-color:#336699; }`
	`cssname { key:val; }`

	The following code defines the styling for all elements with `class="container"`. `container{ background:#fff; margin:0 auto <0px auto; width:640px }`

Formatting User Self Care HTMLs by using macros

Use macros to format the User Self Care HTML pages. You can define your own macro from the mapping rule.

Procedure

1. Place the macro in the HTML. For example, place the macro in `enrollment_validation.html` as a comment.

   ```
   <!-- Observe the macro being replaced
   @USC_FORM_PROFILE_TEST_MACRO@
   -->
   ```

2. Publish the pages to theTivoli Federated Identity Manager run time environment.
3. Define the macro in the mapping rule. The default mapping rule is at `<FIM_INSTALL>\examples\js_mappings\usc.js`.
   a. Add the following text at the bottom of the mapping rule.

      ```
      testMacro = "Macro replaced";
      helper.setSTSOutputAttribute("usc.http.profile.test.macro", testMacro);
      ```

      This attribute `helper.setSTSOutputAttribute` sets the value `Macro replaced` to the attribute `usc.http.profile.test.macro` which is represented in the HTML page by `@USC_FORM_PROFILE_TEST_MACRO@`.
4. Apply the changes to the mapping rule.
   a. Log on to the Integrated Solution Console.
   b. Navigate to **Tivoli Federated Identity Manager** > **Domain Management** > **Runtime Node Management**.
   c. Click **Runtime Custom Properties**.
   d. Set the property **STS.showUSCChains** to **true**.
   e. Click **OK**.
   f. Log out of the Integrated Solutions Console.
   g. Log on to the Integrated Solutions Console.
   h. Navigate to **Tivoli Federated Identity Manager** > **Configure Trust Service** > **Trust Service Chains**.
   i. Select **USC Chain** for **uscCreateAccount**.
   j. Click **Properties**. Two entries of **Default Map Module** under **Trust Service Chain Modules** are present.
   k. Complete these steps for both chain mappings:
      1) Select **Default Map Module** and click **Properties**.
      2) Under **Identity Mapping Rule** for partner module chain, click **Modify Rule**.
      3) Click **Browse** to select the modified mapping rule.
      4) Click **Import File**.
      5) Click **OK**.
   l. Click **Load configuration changes to Tivoli Federated Identity Manager runtime**.
      1) Go to `https://myserver/sps/uscfed/usc/self/account/create`.
      2) Complete the user registration.

      In the User Self Care Enrollment Validation page, the macro is replaced with

      ```
      <!-- Observe the macro being replaced
      Macro replaced
      -->
      ```

      Note: The macros that are defined in the mapping rule are replaced with their values only when the request is received by the STS chain and when the default mapping module contains the definition for the macro.

Formatting User Self Care HTMLs by using Cascading Style Sheets

Use Cascading Style Sheets to format the User Self Care HTML pages.

Procedure

1. Host the css and image files on an IBM HTTP Server.
   a. Navigate to the IHS installation directory.
   b. Open the file `httpd.conf` in `<IHS_INSTALL_ROOT>\conf` with any word processor.
   c. Determine the value of `'DocumentRoot'` from this file. This directory is the location of your css and image documents. The default value is `<IHS_INSTALL_ROOT>\htdocs`.
   d. Create two folders, `css` and `images` under the `DocumentRoot(htdocs)` folder.
   e. Place `usc.css` file under `css` folder and the image file `IBMlogo.png` under the `images` folder.
2. Replace the default User Self Care HTMLs with the modified HTMLs.
   a. Make sure that you edited all the HTML files to point to the `usc.css` file and the image hosted on the web server.

```
...
<link href=""https://company.com/css/usc.css"" rel="stylesheet">
...
<div class="header">
<a href="http://ibm.com" title="Powered by IBM">
<img src="https://company.com/images/IBMlogo.png" alt="IBM"/></a>
</div>
...
```

   b. Apply the same changes to the following HTML pages:
      - `captcha.html`
      - `enrollment.html`
      - `generic.html`
      - `password.html`
      - `profile.html`
   c. Back up the `usc` folder.
   d. Replace `<FIM_INSTALL>\pages\C\usc` with the sample files provided in Technote 1614886, *Cascading Style Sheets for Tivoli Federated Identity Manager User Self Care*, in the Tivoli Federated Identity Manager Support Portal.

Testing the changes to the HTML files

After you publish the HTML files to the Tivoli Federated Identity Manager run time, you can access the User Self Care HTML pages to see the changes.

Procedure

1. Go to `https://myserver/sps/USCFED/usc/self/account/create`.
2. Observe the modified HTML pages.

Integrating User Self Care with WebSEAL

User Self Care deployments that have a Tivoli Access Manager registry in most cases use WebSEAL as a point of contact server. In this scenario, you must integrate interactions between two components that accomplish the task of account deletion and password management.

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.

- Account deletion

  When a user deletes an account from the Tivoli Access Manager registry, as part of a user self deployment, ensure that the current session has ended. This restriction is required for best security practice.

  Deletion of the user session includes the WebSEAL session. User Self Care, by default, terminates the WebSEAL session when the account is deleted. However, this termination is dependent on your prior use of the tfimcfg tool to configure WebSEAL as point of contact server. If you have run this tool as instructed earlier, no special configuration is required.

  If you have not configured WebSEAL as a point of contact server, do so now. See "Configuring WebSEAL as a point of contact server" on page 569

- Password management

  Two password management operations are affected when WebSEAL is the point of contact server. The operations are: Change Password and Expired Password. Both of these integrations require that you permit unauthenticated access to the change password page and use a modified User Self Care Change Password form.

Integrating the change password operation with WebSEAL

When WebSEAL is the point of contact server and a user wants to change a password, the user must provide data. There are several ways that the user can do this task.

Two ways to provide the data are:

- The user can directly access the User Self Care Change Password URL.
- The WebSEAL change password form can redirect the user to the User Self Care Change Password form. You can add a meta tag redirect in the WebSEAL change password page to support this action.

Integrating the expired password operation with WebSEAL

WebSEAL as a point of contact server manages authentication, including expired passwords. However, when User Self Care is integrated with WebSEAL, it must manage the handling of expired passwords.

When this scenario is the case, the following steps occur:

1. WebSEAL flags the authenticated session as `expired`.
2. The user is presented with a modified version of the WebSEAL expired password form.
3. The user provides input and submits the expired password form. This action posts the password data to the User Self Care change password URI.

   Note: The password data must meet certain criteria and POST to the correct User Self Care target URL. When the user has submitted the form, User Self Care processes the form contents and handles any errors. This processing can include showing the User Self Care change password form to the user with error details.

4. User Self Care handles the changing of the password.
5. The WebSEAL session is terminated.

   Note: The WebSEAL session is terminated because the session entry managed by WebSEAL is flagged as `expired`. Until this flag is changed, the user is always presented with the WebSEAL change password form. The user cannot continue, even after changing their password in User Self Care. Terminating the session is also a preferred security practice because it requires the user to log on with their new password, in order to continue.
6. The user is shown the User Self Care password change success page. This page can be modified to redirect back to WebSEAL if wanted.

Configuration steps

Do each of the following steps for the operation you want to integrate with WebSEAL:

- To integrate the change password operation with WebSEAL:
  1. "Permitting unauthenticated access to the User Self Care change password form"
  2. "Modifying the user self care WebSEAL change password form" on page 612
- To integrate the expired password operation with WebSEAL:
  1. "Permitting unauthenticated access to the User Self Care change password form"
  2. "Modifying the user self care WebSEAL change password form" on page 612
  3. "Modifying a WebSEAL expired password form" on page 613
  4. "Supporting redirection back to WebSEAL" on page 614

Permitting unauthenticated access to the User Self Care change password form

To support WebSEAL password operation integration, unauthenticated users must be able to access the Change Password URI through a WebSEAL junction. The junction must be configured with SSL for privacy and confidentiality.

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.

Use pdadmin to permit unauthenticated access to the User Self Care change password form located at:

WebSEAL_server/*fim_junction*/sps/uscfed/usc/self/password/update

Consult the Tivoli Access Manager documentation for information about the **pdadmin** command.

When you change this access, you must use a modified User Self Care Change Password form. Continue with "Modifying the user self care WebSEAL change password form" on page 612

Modifying the user self care WebSEAL change password form

The user ID must be supplied in the Change Password form when integrating User Self Care change password operations with WebSEAL.

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

Permitting unauthenticated access means that it is possible for users to access the change password form. From a security perspective, this user action is acceptable because the user must enter their old password on this form before they can change it. However, you must modify to the default User Self Care form to activate this function.

By default, User Self Care does not require users to enter their user ID on the change password form. Instead, User Self Care gathers it from the authenticated context. This mechanism does not work if the user does not authenticate before requesting the form. If the user requests the form without authenticating, User Self Care returns an error message indicating that no authenticated user identity is available.

To avoid this error, the user ID must be supplied in the Change Password form when integrating User Self Care change password operations with WebSEAL.

Procedure

1. Make a backup copy of *FIM_install_dir*/pages/C/usc/password/changepassword.html.
2. Copy the example file changepassword.html to the User Self Care pages repository.
   - The example file is:
     FIM_install_dir/examples/examples/html/usc/password/changepassword.html
   - The destination location is:
     FIM_install_dir/pages/C/usc/password/changepassword.html
3. Log on to the administrative console.
4. Go to the **Runtime Node Management** panel.
5. Click **Refresh Pages**.
6. Save the configuration changes.

What to do next

- If you are integrating the change password operation, you have finished the task.
- If you are integrating the expired password operation, continue with "Modifying a WebSEAL expired password form" on page 613.

Modifying a WebSEAL expired password form

Modify the WebSEAL expired password form to ensure correct handling of passwords with User Self Care.

Before you begin

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.

Ensure that you have finished the prerequisite tasks:

1. "Permitting unauthenticated access to the User Self Care change password form" on page 611
2. "Modifying the user self care WebSEAL change password form" on page 612

About this task

There is more than one way to modify the form.

Procedure

1. Copy the User Self Care `changepassword.html` file to the WebSEAL directory where the management pages are located. Rename it to `usc_changepassword.html`.

   For example:

   ```
   /opt/pdweb/www-default/lib/html/C/usc_changepassword.html
   ```

2. Edit the `usc_changepassword.html` form, as follows:
   a. Add a new hidden field:

      ```
      <input type="hidden" name="usc.form.password.expired.flag" value="true" />
      ```

   b. Add another new hidden field:

      ```
      <input type="hidden" name="usc.form.userid" value="%USERNAME%" />
      ```

   c. Remove or comment out the two lines:

      ```
      <div class="hidden" id="errorDiv"> </div>
      <div class="hidden" id="errorAttrDiv"> </div>
      ```

   d. Replace the form `ACTION` macro with the URL of the User Self Care change password target.

      For example:

      ```
      https://webseal.example.com/fimjct/sps/uscfed/usc/self/password/update
      ```

3. Set the file permissions and ownership of `usc_changepassword.html` to match the permissions of the other WebSEAL management files.
4. Edit the WebSEAL configuration file. Go to the `acnt-mgt` stanza and change `passwd-expired = passwd_exp.html` to `passwd-expired = usc_changepassword.html`
5. Restart WebSEAL.

What to do next

Optionally, continue with "Supporting redirection back to WebSEAL" on page 614.

Supporting redirection back to WebSEAL

Optionally, you can direct users back to WebSEAL after they have changed their password.

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

In some cases, you might want to host a *landing page* with links to destinations from the WebSEAL system rather than the User Self Care system.

Procedure

1. Create a *password change success* page in the WebSEAL docs directory.

   This page is the landing page at WebSEAL. It can say: `Your password has been successfully changed, you will have to login again to access any protected pages.`
2. Modify the User Self Care page located at *`FIM_install_dir`*`/pages/C/usc/password/changepassword_success.html` to add a meta-redirect tag that redirects the client to the new WebSEAL *password change success* page.

Modifying a User Self Care federation

There are some limitations on how you can modify existing User Self Care federations.

- The command line interface does not support modification of User Self Care federations. Use the administration console to set the runtime property `STS.showUSCChains` to `true`. View the User Self Care trust chains and modify the trust chains and properties as needed.

  As an alternative, you can configure User Self Care by repeating the initial deployment steps. In this case, you must create and edit a new response file, and then use the command line interface to deploy the federation.
- You cannot capture, within a response file, configuration settings that are specific to a particular chain. For example, Attribute Mapping STS modules use a mapping rule file. Different chains might have different mapping rules. You cannot specify the different mapping rules when creating a response file from an existing configuration.

  Parameters that can be specific to a particular chain do not have values set in the response file. When different chains have different mapping rules, use the administration console to modify the chain modules to use different rules files.

Unconfiguring User Self Care

Use **wsadmin** to unconfigure User Self Care.

About this task

This task deletes the User Self Care trust chains and the User Self Care federation.

Procedure

1. Start **wsadmin**.
2. Issue the command:

```
$AdminTask manageItfimUserSelfCare {-operation unconfigure
-fimDomainName your_domain_name -federationName uscfed}
```

Chapter 43. Tuning User Self Care

You can improve User Self Care performance by adjusting settings for several distributed caches.

User Self Care maintains three different distributed caches:

- Account Create Cache
- Forgotten Password Cache
- Secret Question Failure Cache

The caches are shared among WebSphere Application Server cluster members to permit a user operation to be properly handled. This sharing is required in case different phases of the operation take place on different nodes.

User Self Care uses the WebSphere Distributed Object Cache technology for implementation of the caches. See the WebSphere Application Server documentation for details on this caching technology.

There are two parameter types that affect each User Self Care distributed cache:

`Entry lifetimes`
: These parameters are set in the User Self Care response file. Cache entries are retained until either the lifetime is hit or the user finishes the operation requiring the cache entry. The names and settings of these cache-specific parameters are described in the individual cache tuning descriptions later in this set of topics.

`Cache sizes`
: These parameters are set in the administrative console by accessing **Resources** > **Cache Instances** > **Object Cache Instances**. The `Cache size` parameter controls how many concurrent entries are retained in the cache. The names and settings of these cache-specific parameters are described in the individual cache tuning descriptions

You must size the caches adequately so users can perform operations that require a distributed cache in the configured time period. If a cache is too small, users might not be able to validate their accounts or recover their passwords during the specified time period. You can specify the time period in the configuration for lifetime of the cache entries.

For example, to give your users two minutes to finish an account recovery validation, configure the entry lifetime for the account recovery validation cache to be two minutes. If you expect two users per second to perform an account recovery operation, set the account recovery validation cache to at least 240.

Determine the appropriate size using the following calculation:

```
120 seconds x 2 users/second = 240
```

The default size of the account recovery validation cache is 1000 entries. This default would be adequate for this example. Other operations, such as account creation, might require an increase in the cache size.

Depending on your expected system usage, you might increase the size of one or more caches. This adjustment can affect your hardware requirements. Cache entries take up memory and must be replicated between systems in the cluster.

A preferred performance tuning practice is to provide a buffer for the expected cache size.

See the following topics:

- "Account create cache"
- "Forgotten password cache"
- "Secret question failure cache" on page 619
- "Notes about tuning caches" on page 619

Account create cache

This cache stores the data from the user inputs during the account creation and e-mail validation process. When the user finishes the validation, the User Self Care recovers the data from the cache to create an account in the registry.

Table 151. Account create cache parameters

Parameter	Description
AccountCreateLifetime	Entry lifetimes are controlled by the `AccountCreateLifetime` parameter described in the topic: Chapter 44, "Response file parameters," on page 621.
itfim-usc_accountcreate	Cache size is controlled by the `itfim-usc_accountcreate` cache size.

Unlike other operations, each account creation operation creates two cache entries. One entry is consists only of the user ID and a key. The second entry consists of all the data that the user enters in the account creation form.

You configure cache entry lifetimes to be 120 seconds. You expect a peak number of users enrolling during a new application provisioning operation to be 10 each/second. You might want to size your cache as follows:

```
10 users/second x 2 entries/user x 120 seconds/entry = 2400 x 20% buffer ~= 3000.
```

Forgotten password cache

This cache stores the user ID during the Forgotten Password validation operation.

Table 152. Forgotten password cache parameters

Parameter	Description
AccountRecoveryValidationLifetime	Entry lifetimes are controlled by the `AccountRecoveryValidationLifetime` parameter described in the topic: Chapter 44, "Response file parameters," on page 621.
itfim-usc_forgottenpassword	Cache size is controlled by the `itfim-usc_forgottenpassword` cache size. This entry is small, consisting of the user ID and a key.

Secret question failure cache

This cache stores the number of failed secret question answer attempts that have been performed.

Table 153. Secret question failure cache parameters

Parameter	Description
AccountRecoveryFailureLifetime	Entry lifetimes are controlled by the `AccountRecoveryFailureLifetime` parameter described in the topic: Chapter 44, "Response file parameters," on page 621
itfim-usc_secretquestionfailures	Cache size is controlled by the `itfim-usc_secretquestionfailures` cache size. This entry is consists only of a number and a key.

Notes about tuning caches

Configuration of WebSphere Application Server operations can improve your tuning of the caches.

- Replication

  WebSphere does not automatically replicate all of the cached data between nodes. Instead, it just replicates the keys between nodes and only retrieves the data when requested by a particular node. If a key is requested on a particular node system that is not found in the cache, User Self Care attempts the cache lookup operation. The attempt provides time for WebSphere Application Server to finish any possible replication.
- Cache flushes

  Restarting WebSphere Application Server clears caches and returns them to a clean state.
- Removal of User Self Care caches

  Cache entries are retained until either the entry lifetime is hit or the user finishes the operation requiring the cache entry.

Chapter 44. Response file parameters

Use the parameters described in this section to configure response files for User Self Care.

AccountCreateLifetime
Specifies the amount of time, in seconds, that User Self Care recognizes the account creation request as valid, and retain the request in the internal cache. If the Create Account trust chain does not finish account creation in the specified time, the request is discarded and account creation terminates.

This property is required.
Type: Integer
Default: 86400
Maximum: none
Minimum: 0

A setting of '0' disables account creations because entries are not retained in the cache. Larger settings can affect memory consumption and potentially affect performance in replicated environments due to increased data being replicated using DynaCache across nodes.

When setting this property, also consider an appropriate size for the `itfim-usc_accountcreate cache`. See: Chapter 43, "Tuning User Self Care," on page 617.

AccountRecoveryFailureLifetime
Specifies how long, in seconds, the program retains record of an unsuccessful account validation attempt. When the specified time period elapses, the record of the unsuccessful attempt is discarded, and the counter is decremented by one.

Type: Integer
Default: 86400
Maximum: none
Minimum: 0. The value 0 means to disable locking.

When setting this property, also consider an appropriate size for the `itfim-usc_secretquestionfailures` cache. This parameter is configured separately as part of tuning User Self Care. See: Chapter 43, "Tuning User Self Care," on page 617.

AccountRecoveryFailureLimit
Specifies the number of times a user can attempt but fail to restore account access before the program locks the account. If the user does not supply a correct answer to the secret question, account access is not restored. When the user fails to restore account access, the value of this property increments by one. When the value equals the specified number, the program locks the account.

Type: Integer
Default: 3
Maximum: none
Minimum: 0

A setting of 0 or 1 for the minimum causes the account to be locked after the first failure.

AccountRecoveryFailureLockoutTime
Specifies how long, in seconds, the program keeps the account locked after the user has exceeded the maximum number of unsuccessful validation attempts. When the program has locked the account, this value specifies the amount of time that must pass before the program unlocks the account.

Type: Integer
Default: 86400
Maximum: none
Minimum: 0. The value 0 disables locking.

AccountRecoveryLookupAttribute
Specifies a user attribute used for user ID lookup. This property specifies a single attribute that the user enters in the Forgotten user ID form to retrieve their user ID (identity). User Self Care uses this registry attribute as a lookup field. User Self Care searches for an entry that contains the attribute supplied by the user, and returns the matching user ID. The attribute value is assumed to be an email address. An email containing all the forgotten user IDs are sent to this address.

Type: string
Default: `mail`

AccountRecoveryLookupField
This field is deprecated. Do not modify.

AccountRecoveryValidationAttributes
This field is deprecated. Do not modify.

AccountRecoveryValidationLifetime
Specifies the amount of time, in seconds, that User Self Care considers the account validation request to be valid.

During password recovery, users must finish a validation step before recovering their password. The validation step consists of responding to a user self care e-mail that specifies a link to access. If the user does not respond within the time period specified by this parameter, the program invalidates the link in the e-mail.

Type: Integer
Default: 86400
Maximum: none
Minimum: 0

A setting of 0 for the minimum disables the ability to recover an account.

When setting this property, also consider an appropriate size for the `itfim-usc_forgottenpassword` cache. This parameter is configured separately as part of tuning User Self Care. See: Chapter 43, “Tuning User Self Care,” on page 617.

AttributeMappingFilename
Specifies the path to the location of a file that contains the transformation rules for use with the Attribute Mapping STS Module. This file can be either a JavaScript or XSLT file.

User Self Care ships with a default JavaScript file named `usc.js`:

Federated_Identity_Manager_installation_dir/examples/js_mappings

This property is required.
Type: String
Default: none

Example:

/opt/IBM/FIM/examples/js_mappings/usc.js

BaseURL

Specifies a fully qualified URL for the root of the User Self Care federation. User Self Care uses the root to construct dynamic HTML elements. The syntax is as follows:

method//*POC_server:port*/*FIM_junction*/sps

Where:

method
Must be either http:or https:

POC_server:port
The fully qualified host name, and optional port number, of the point of contact server.

FIM_junction
The name of the WebSEAL junction. This value is only required when using a WebSEAL point of contact server.

This property is required.
Type: String
Default: none

Example:

https://myWebSEALserver.example.com/myTFIMjct/sps

Note: If you are using WebSEAL as a point of contact server, you likely have not yet created a junction to the Tivoli Federated Identity Manager server. In most cases, you create this junction at the end of the User Self Care configuration steps. However, you must determine the name of the junction now, so that you can set the BaseURL value in the response file now. You must remember the junction name, for use later when running the **tfimcfg** command.

CaptchaSTSModuleId

Specifies either the demonstration Captcha module, or specifies a placeholder module that takes no action. When this value is specified, User Self Care activates the demonstration Captcha module.

This property is required.
Type: String
Default: none

There are two valid values for this field:

- usc-captcha-demo

  Use this value if you want to activate the demonstration Captcha module. If you use this setting, you must set the other Captcha settings in this response file. To use the Captcha demonstration, you must also configure the module. See: "Configuring the Captcha demonstration" on page 567.

- `default-usc-captcha-noop`

  Use this value if you want to use the placeholder module `USCNoOpsSTSModule`. This module takes no action, but serves as a placeholder for a customer-provided validation module that can be used, as an example, for Captcha validation. The USCNoOpsSTSModule makes it easier for customers to provide their own module without redefining the trust chains.

DemoCaptchaImageAndKeyList

This field is required if you are using the Captcha demonstration module.

The contents are fixed and must not be modified.

Note: The `DemoCaptchaImageAndKeyList` parameter has already been set. The program ignores this parameter if you are not using the demonstration Captcha module.

DemoCaptchaImageRootURL

Specifies a URL of a directory that contains the images used for the demonstration Captcha module provided with User Self Care.

You must specify a value for this property if you want to use the Captcha demonstration module.

Example:

```
https://images.example.com/captcha/demo
```

EnrollmentEmailSender

Specifies a fully qualified e-mail address for the account that User Self Care uses to send a message to the user. The message validates the user enrollment. In most cases, this address is an e-mail address that does not receive responses.

This property is required.
Type: string
Default: none

Example:

```
no-reply@example.com
```

EntitySuffix

Specifies a suffix where created users are stored in the registry. This suffix must uniquely identify the registry that User Self Care uses for all operations.

This property is required.
Type: String
Default: `o=ibm,c=us`

GroupMembershipGroups

Specifies a list of groups to which to add newly enrolled users. Specifies one or more groups that are defined in the user registry used by the Create Account trust chain. The group names are specific to the user registry.

Type: String
Default: none

Example:

```
<void method="add">
 <string>Group1</string>
</void>
<void method="add">
 <string>Group2</string>
</void>
```

PasswordRecoveryEmailSender

Specifies a fully qualified e-mail address for the User Self Care account that sends a message to the user. User Self Care uses the message to validate a password recovery operation. In most cases, this e-mail address does not receive responses.

This property is required.
Type: string
Default: none

Example:

```
no-reply@example.com
```

ProfileManagementAttributes

Defines the set of registry attributes that are used for profile information. To provide a working prototype, the user self care solution defines a set of registry attributes for use with the default function. User Self Care does not modify the schema of the target registry. For this reason, the number of profile attributes are limited and use standard LDAP attributes that are present in most cases.

This property is required. The list of attributes used are:

- businessCategory
- roomNumber
- mobile
- mail

The attributes are represented in the configuration file as follows:

```
<object class="java.util.ArrayList">
 <void method="add">
  <string>businessCategory</string>
 </void>
 <void method="add">
  <string>roomNumber</string>
 </void>
 <void method="add">
       <string>mail</string>
     </void>
 <void method="add">
       <string>mobile</string>
     </void>
</object>
```

Figure 69. Profile management attributes in the response file

SecretQuestionMinimumNumber

Specifies the minimum number of required secret questions that a user must provide answers to during enrollment. Depending on the configuration, some or all of the secret questions may be presented to the user for verification purposes when they forget their password.

This property is optional.
Type: Integer
Default: 2
Maximum: none
Minimum: 1

SecretQuestionMaximumNumber
Specifies the maximum number of secret questions that a user can provide answers to during enrollment. Depending on the configuration, all of the secret questions is presented to the user for verification purposes when they forget their password.

This property is optional.

Type: Integer

Default: 3

Maximum: none

Minimum: The maximum value depends on `SecretQuestionMinimumNumber`. The maximum value should at least be the same as the value specified in the `SecretQuestionMinimumNumber` parameter.

SecretQuestionRequiredForValidationNumber
Specifies the number of secret questions a user must answer correctly to validate their identity.

During password recovery, users must provide correct answers to the secret questions presented to them. The number of questions that they must answer correctly is dependent on this parameter.

Example scenario:

During enrollment, a user is presented with 3 secret questions that they must provide answers to.

An administrator configures the parameter to:
`SecretQuestionRequiredForValidationNumber=2`

When the user forgets their password, all 3 secret questions are presented to them. However, since the parameter was set to `SecretQuestionRequiredForValidationNumber=2`, the user only needs to answer 2 out of the 3 questions correctly. They can leave one of the fields blank. If the user chooses to answer all the questions presented to them, they must get all the answers correctly.

In this scenario, if a user chooses to answer 3 questions, they must provide the correct answers for all 3 questions to be validated. The user cannot be validated either if they only answer 1 out of the 3 questions correctly.

This property is optional.

Type: Integer

Default: 2

Maximum: none

Minimum: The maximum value depends on `SecretQuestionMinimumNumber`. The maximum value should at most be the same as the value specified in the `SecretQuestionMaximumNumber` parameter.

SMTPAuthenticatePassword
The password for the account specified by the SMTPAuthenticateUsername parameter if using authentication to the SMTP server. This property is optional.

Type: string
Default: none

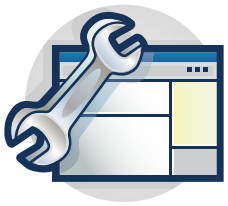
SMTPAuthenticateUsername
The user name that authenticates to the SMTP server. This property is optional.

Type: string
Default: none

SMTPServerName
The fully qualified host name of the Simple Mail Transport Protocol (SMTP) server that sends e-mail for the user. This property is required.

Type: string
Default: none

Part 7. Configuring one-time password



The topics in the Configuration section provide a step-by-step guide to configuring one-time password.

This section describes the deployment of one-time password. First read the overview of the one-time password feature:

"One-time password overview" on page 631

Chapter 45. One-time password

Configure Tivoli Federated Identity Manager to use one-time password as an authentication factor in a federated single sign-on and in an extended authentication scenario.

One-time password overview

Tivoli Federated Identity Manager provides various authentication mechanisms in the point of contact interface.

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

The authentication methods available in a deployment are typically determined by the point of contact technology that is used in the environment. Points of contact technologies usually provide simple authentication such as the use of a user name and password.

A step-up authentication is a type of authentication where users who attempt to access sensitive resources are required to provide a specific type of credential. They might be challenged to authenticate and provide an extra set of credentials to prove that they are allowed to access sensitive resources. The one-time password authentication can be used where increased security is required.

A multi-factor authentication is a type of authentication where users are required to provide more than one type of credential to access a protected resource.

A one-time password is a unique password that validates a login session. A one-time password cannot be reused. These restrictions make it less vulnerable to replay attacks and more secure than static passwords.

The one-time password authentication capability in Tivoli Federated Identity Manager extends the existing point of contact support with the following features:

- Context-based authentication policy determination by using a mapping rule.
- Pluggable one-time password generation and validation with default implementation.
- Pluggable one-time password delivery with email and short message service (SMS) as default implementation.
- Pluggable one-time password store with default support for in memory cache.
- Time-based and counter-based one-time password generation that is created by both client and server so that no delivery mechanism is required.
- Pluggable user information storage and retrieval for one-time password generation and validation that requires user information to be available.

You can implement the use of the one-time password in the federated protocol or extended authentication flow.

Federated single sign-on scenario
This flow consists of allowing multi-factor and step-up authentication operations. This flow relies on a one-time password authentication in the context of a single sign-on protocol.

Extended authentication scenario
This flow consists of allowing multi-factor and step-up authentication operations. This flow relies on a one-time password authentication to extend the authentication capabilities of existing point of contact technologies. This flow is available outside the context of a federated single sign-on.

One-time password configuration overview

The one-time password feature has several components. Understand what you must configure to implement the feature to suit your requirements.

You can implement the one-time password feature in two scenarios:

- Federated single sign-on scenario
- Extended authentication scenario

Each point of contact is configured to use with a federation. Only one point of contact can be active at a time. A single one-time password federation can be used by multiple point of contacts.

There are two required parts that must be configured to enable the one-time password feature:

- Configure the point of contact profile. Configuring the context-based authentication policy might be included.
- Configure the one-time password federation that establishes the runtime web endpoints and supporting Security Token Service (STS) modules and chains. This configuration includes the one-time password generation, validation, and delivery methods.

Configuring the point of contact with the one-time password support can be done in the console. Configuring the one-time password federation component can be done in the command-line interface only.

A Point of Contact profile configuration allows for multiple authentication callbacks to be configured. The execution of an authentication event on Tivoli Federated Identity Manager consists of starting the list of configured authentication callbacks. The execution of the one-time password flows consists of starting the list of configured authentication callbacks. Each configured callback is assigned an authentication level. The configured authentication level represents the level of assurance that each authentication callback provides. The required authentication level that the policy configuration requires dictates which of the configured callbacks are run. If an authenticated session exists when the authentication event happens, the required authentication level determines if a particular token provided for the authenticated session is satisfactory.

Both the federated protocol or User Self Care flow and the extended authentication flow depends on the following parameters:

- **Required authentication level** dictates the level of authentication that is required of a user to be able to access a protected resource. Authentication levels are represented by an integer number. Each authentication callback is assigned an authentication level.

During an authentication event the configured authentication callbacks are used. The required authentication policy is enforced. To evaluate if an authenticated session is satisfactory based on the policy, the Tivoli Federated Identity Manager point of contact retrieves the authentication level of the credential. If there is no valid or satisfactory credential exists, the callbacks are started until a satisfactory level is achieved. An administrator must assign an authentication level to each configured authentication callback.

- **Type of authentication** determines the type of authentication that is required for a user to be able access a protected resource. There are two supported types of authentication.

  Hierarchical authentication type (step-up)
  Executes the authentication callback with an assigned authentication level that is equal to or higher than the required level. It is executed until a satisfactory authentication is achieved.

  Complementary authentication type (multi-factor)
  Executes all the authentication callbacks that are configured until a satisfactory authentication is achieved.

- **Authentication mode** dictates the type of mode where authentication callbacks are run. There are two supported types of authentication mode.

  Group authentication mode
  Runs all the authentication callbacks in the same HTTP exchange.

  Individual authentication mode
  Runs each authentication callback that requires user interaction in a separate HTTP exchange.

You also have the option of using your own mapping rule to determine the required authentication policy that is based on request attributes. This technique is called context-based authentication policy determination. You can upload a JavaScript rule to determine the authentication level, authentication mode, and authentication type in the generic point of contact authentication policy.

An *authentication policy*

- Applies a set of rules to the authentication process and to the verification of authentication data
- Determines enforcement that is based on the request context.
- Consists of the required authentication level, mode, and type.

In the absence of context-based authentication policy determination, the authentication policy is determined statically based on the configuration.

The authentication policy can be set in the following locations:

Authentication policy mapping rule
See Authentication policy mapping rule for more details.

Query string
See the *Modifying stepuplogin.html* step in "Configuring one-time password extended authentication with WebSEAL as point of contact" on page 637.

Point of contact
See the **allow.authentication.policy.request.overrides** parameter in "Creating your own one-time password point of contact" on page 642.

A default implementation of the authentication policy callback is provided. The default implementation is configured to allow for a static policy configuration. It

might also rely on a customer-configured mapping rule to determine the authentication policy from request attributes.

In the extended authentication flow, an endpoint is provided for the external point of contact technology to start the flow. A target URL is provided in a query string parameter. Users are redirected to this URL when the authentication event is completed.

Any user token information of an existing authentication available on the request and request attributes are collected. It is collected for context-based authentication policy determination and one-time password flow processing at the Security Token Service (STS) chain.

Tivoli Federated Identity Manager provides the following pluggable extension points:

- One-time password provider — generates and validates the one-time password value.
- One-time password delivery module — delivers the one-time password value.
- One-time password store module — stores the one-time password value.
- One-time password user information provider module — stores and retrieves the user information that is required to calculate the one-time password value.
- Authentication Policy Callback — responsible for determining the required authentication policy that is based on the request context.
- Dynamic one-time password provider determination that is based on the request context.

Chapter 46. One-time password deployment

You must configure various components, such as one-time password federation and point of contact, to deploy one-time password authentication. One-time password is valid for either the federated single sign-on or extended authentication scenario.

The following list summarizes the tasks for deploying one-time password and the order in which to do them. Before you start a task, ensure that you have finished any prerequisite tasks.

1. Decide on the appropriate scenario for deploying one-time password authentication. See "One-time password overview" on page 631.
2. Configure a one-time password federation. The configuration steps include configuring a response file. Specify the configuration of the one-time password mapping rules, one-time password provider plug-ins, and the one-time password delivery plug-ins.

   "Configuring a one-time password federation"
3. Optional step: customize a one-time password point of contact. You can create your own point of contact profile.

   "Creating your own one-time password point of contact" on page 642
4. Activate the one-time password point of contact. Use the Integrated Solutions Console to active the one-time password point of contact.

   "Activating the one-time password point of contact" on page 636
5. Depending on the scenario where you want to deploy one-time password authentication, complete one of the following tasks:
   - Customizing one-time password for a federated single sign-on scenario.
   - Customizing one-time password for extended authentication scenario.

Configuring a one-time password federation

Use a response file to configure your one-time password federation. Specify the configuration of the one-time password mapping rules, one-time password provider plug-ins, and the one-time password delivery plug-ins.

Procedure

1. Create a response file with the wsadmin tool by entering the following commands on one line.
   - Create a response file

     ```
     $AdminTask manageItfimOneTimePassword  { -operation createResponseFile
     -fimDomainName domainName -fileId fileId }
     ```
   - Create a response file based on an existing one-time password federation:

     ```
     $AdminTask manageItfimOneTimePassword  { -operation createResponseFile
     -fimDomainName domainName -federationName federationName -fileId fileId }
     ```

   Where:

   domainName is the name of your domain.

   federationName is the name of the one-time password federation.

   fileId is the name of the one-time password response file

2. Edit the parameters in the response file. See "One-time password response file" on page 669.

   Important: If you plan to use `WebSEAL with OTP` as your point of contact, ensure that the federation name is `otpfed`.
3. Configure the one-time password federation with your response file by entering the following commands on one line:

   ```
   $AdminTask manageItfimOneTimePassword { -operation configure
   -fimDomainName domainName -fileId fileId }
   ```

   Where:

   domainName is the name of your domain.

   fileId is the name of the one-time password response file you created in 1 on page 635
4. If your mapping rule is syntactically valid, but Tivoli Federated Identity Manager reports that it is not, add `STS.validateMappingRules` and set the value to `false`. The message `FBTADM001I Command completed successfully` is returned.
5. Enter the following command in the wsadmin tool:

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

   Where:

   domainName is the name of your domain.

Activating the one-time password point of contact

Use the Integrated Solutions Console to activate the one-time password point of contact.

Procedure

1. Log on to the Integrated Solutions Console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Point of Contact**.
3. Select **WebSEAL with OTP** or create your own point of contact, see "Creating your own one-time password point of contact" on page 642 for more details.
4. Click **Make Active**.
5. Click **Load configuration changes to the Tivoli Federated Identity Manager runtime.**

Configuring the one-time password in a federated single sign-on flow

Configure one-time password authentication to enable single sign flow in a federation.

Procedure

1. Complete steps 1 - 4 of the task Deploying one-time password.
2. Optional: If the federation where the one-time password feature is used was created before Tivoli Federated Identity Manager 6.2.2, Fix Pack 4, you must rerun `tfimcfg` command against that federation so that the required EAI trigger URL is added into WebSEAL configuration file. You can reuse or re-create the junction. See tfimcfg Reference for more details.

Verifying the one-time password federated single sign-on configuration

Verify your federated single sign-on configuration to ensure that the one-time password implementation works correctly.

About this task

Complete the steps that are provided to verify that your one-time password configuration in the federated single sign-on flow is configured properly.

Procedure

1. Initiate a federated single sign-on flow. Depending on the protocol you are using for your federation, the endpoint URL to initiate the single sign-on might be different. For example: `https://idp.example.com/sps/<federationName>/saml20/logininitial?PartnerId=sp.example.com`
2. You are redirected to the identity provider login page.
3. Enter your user name and password. Depending on your configuration, you might be asked to select the one-time password delivery method.
4. Enter the one-time password that was delivered in the delivery method that you selected. If you authenticate successfully, you are redirected to protected resource in the service provider.

Configuring one-time password extended authentication with WebSEAL as point of contact

Configure WebSEAL to enable the extended authentication flow for one-time password.

Before you begin

Deploy one-time password authentication.

About this task

The `eai-auth` stanza entry, which is in the `[eai]` stanza of the WebSEAL configuration file, enables, and disables the external authentication interface function. The external authentication interface can be implemented over HTTP, HTTPS, or both.

External authentication interface is disabled by default.

Procedure

1. Edit the WebSEAL configuration file to match the Tivoli Federated Identity Manager point of contact profile configuration. For example, `$WEBSEAL_INSTALL_DIRECTORY&/etc/webseald-default.conf`.
   a. For example:

```
[eai]
#----------------------
# EXTERNAL AUTHENTICATION INTERFACE
#----------------------

# Enable EAI authentication. No other EAI parameters will take effect
# if this is set to 'none'.
#
# One of <http, https, both, none>
```

```
# Added by FIM TAM autoconfig: Thu Apr 15 12:33:58 CDT 2010
eai-auth = https

# IMPORTANT
# An appropriate authentication library must be configured to handle
# EAI authentication to complete this configuration.  Please
# refer to the "authentication mechanisms and libraries" subsection
# at the end of the authentication section.

# EAI HEADER NAMES

# If eai-auth is not 'none', and WebSEAL has received a trigger URL
# in a request, WebSEAL will examine the corresponding server response for
# the following headers.  These are the headers that
 will contain authentication
# data used to authenticate the user.

# EAI PAC header names
eai-pac-header = am-fim-eai-pac
eai-pac-svc-header = am-eai-pac-svc

# EAI USER ID header names
eai-user-id-header = am-fim-eai-user-id
eai-auth-level-header = am-eai-auth-level
eai-xattrs-header = am-eai-xattrs
# EAI COMMON header names
eai-redir-url-header = am-fim-eai-redir-url
# RETAIN EAI SESSION
# If an already-authenticated EAI client authenticates via an
EAI a second
# time, the existing session and cache entry are completely replaced by
# default.  If retain-eai-session = yes, then the existing session and
# cache entry will be retained, and the credential and relevant data will
# be updated in the existing cache entry.
retain-eai-session = no
eai-redir-url-priority = yes
```

b. Add the EAI trigger URL. For example:

```
# EAI TRIGGER URLS
[eai-trigger-urls]
trigger = /FIM/sps/auth*
```

c. Add EAI to the authentication-mechanisms stanza. For example:

```
[authentication-mechanisms]
#----------------------
# AUTHENTICATION MECHANISMS AND LIBRARIES
#----------------------
# List of supported authentication mechanisms and
# their associated shared libraries
# Uncomment the line and supply the full path to a library to
# enable a mechanism.

# Username/Password - such as Basic Authentication or Forms
#passwd-cdas       = <passwd-cdas-library>
#passwd-ldap       = <passwd-ldap-library>
#passwd-uraf       = <uraf-authn-library>
passwd-ldap = /opt/PolicyDirector/lib/libldapauthn.so &
-cfgfile [/opt/pdweb/etc/webseald-webseald-ip.conf]
cert-ldap = /opt/PolicyDirector/lib/libcertauthn.so &
-cfgfile [/opt/pdweb/etc/webseald-webseald-ip.conf]

ext-auth-interface = /opt/pdwebrte/lib/libeaiauthn.so
```

Note: The location of the library file might be different in Windows for example, C:\Program Files\Tivoli\PDWebRTE\bin\eaiauthn.dll.

d. Modify the user session ID settings. For example:

```
#----------------------
# USER SESSION IDS
#----------------------
# Enable/disable the creation and handling of user session ids.
user-session-ids = yes

# Include the replica set name in the user session ID.  If set to "yes"
# then the user-session-id will include the replica set.  If set to "no"
# then WebSEAL will not include the replica set in the user-session-id,
# and will assume that all user-sessions specified in
the "terminate session"
# command belong to the standard junction replica set.
user-session-ids-include-replica-set = yes
```

e. Modify the `authentication-levels` stanza. For example:

```
[authentication-levels]
0 = unauthenticated
1 = password
2 = ext-auth-interface
```

f. Modify the `authentication` stanza. For example:

```
##############################
# AUTHENTICATION
##############################

[ba]
#----------------------
# BASIC AUTHENTICATION
#----------------------

# Enable authentication using the Basic Authentication mechanism
# One of <http, https, both, none>
# Added by FIM TAM autoconfig: Tue Nov 27 10:47:33 SGT 2012
# Enabling forms instead of BA for improved user interface
ba-auth = none



[forms]
#----------------------
# FORMS
#----------------------

# Enable authentication using the forms authentication mechanism
# One of <http, https, both, none>
# Added by FIM TAM autoconfig: Tue Nov 27 10:47:33 SGT 2012
# Enabling forms instead of BA for improved user interface
forms-auth = https
```

2. Modify `stepuplogin.html` so that it redirects the authentication request to Tivoli Federated Identity Manager extended authentication endpoint.

   a. Navigate to the folder where `stepuplogin.html` is located. For example, `$WEBSEAL_INSTALL_DIRECTORY$/www-default/lib/html/$LOCALE$/stepuplogin.html`.

   b. Insert the following code in the `Javascript` section of the file. For example:

```
authnlevel="%AUTHNLEVEL%";
 if (authnlevel == "2")
{
    window.location = "https://<HOST>:<PORT>/<JUNCTION>
/sps/xauth?Target=
%HTTPS_BASE%%URL_ENCODED%[&AuthenticationLevel=<RequiredAuthenticationLevel>]
[&AuthenticationType=
HIERARCHICAL|COMPLEMENTARY][&AuthenticationMode=INDIVIDUAL|GROUP]"
}
```

For example:

```
authnlevel="%AUTHNLEVEL%";
 if (authnlevel == "2")
{
    window.location = "https://idp.example.com/FIM/sps
/xauth?Target=%HTTPS_BASE%%URL_ENCODED%&AuthenticationLevel=2"
}
```

In this example, level 2 step-up request is forwarded to Tivoli Federated Identity Manager.

3. Create a Tivoli Federated Identity Manager junction with the following Tivoli Access Manager pdadmin commands:

   ```
   pdadmin sec_master> server task <WEBSEAL_SERVER_NAME> create -t tcp
   -h <BACKEND_SERVER_HOST_NAME> -p <BACKEND_PORT> -c all -f /FIM
   ```

   For example:

   ```
   pdadmin sec_master> server task default-webseald-localhost create -t tcp
   -h localhost -p 9080 -c all -j -r -q /sps/cgi-bin/query_contents -f /FIM
   ```

4. Create a one-time password extended authentication unauthenticated access control list (ACL).
   a. pdadmin sec_master> acl create xauth_unauth
   b. pdadmin sec_master> acl modify xauth_unauth set Group iv-admin TcmdbsvaBRrxl
   c. pdadmin sec_master> acl modify xauth_unauth set Group webseal-servers Tgmdbsrxl
   d. pdadmin sec_master> acl modify xauth_unauth set User sec_master TcmdbsvaBRrxl
   e. pdadmin sec_master> acl modify xauth_unauth set Any-other Tr
   f. pdadmin sec_master> acl modify xauth_unauth set Unauthenticated Tr
   g. pdadmin sec_master> acl show xauth_unauth

      ```
      ACL Name: xauth_unauth
      Description:
      Entries:
      User sec_master TcmdbsvaBRrxl
      Any-other Tr
      Unauthenticated Tr
      Group webseal-servers Tgmdbsrxl
      Group iv-admin TcmdbsvaBRrxl
      ```

5. Attach the one-time password extended authentication access control list (ACL) to the extended authentication endpoint with the following command:

   ```
   pdadmin sec_master> acl attach /WebSEAL/<WEBSEAL_INSTANCE_ROOT>
   /FIM/sps/xauth xauth_unauth
   ```

   For example:

   ```
   pdadmin sec_master> acl attach /WebSEAL/localhost-webseald-ip
   /FIM/sps/xauth xauth_unauth
   ```

6. Create a one-time password extended authentication authenticated access control list (ACL).
   a. pdadmin sec_master> acl create xauth_anyauth
   b. pdadmin sec_master> acl modify xauth_anyauth set Group iv-admin TcmdbsvaBRrxl
   c. pdadmin sec_master> acl modify xauth_anyauth set Group webseal-servers Tgmdbsrxl

d. `pdadmin sec_master> acl modify xauth_anyauth set User sec_master TcmdbsvaBRrxl`

e. `pdadmin sec_master> acl modify xauth_anyauth set Any-other Tr`

f. `pdadmin sec_master> acl modify xauth_anyauth set Unauthenticated T`

g. `pdadmin sec_master> acl show xauth_anyauth`

```
ACL Name: xauth_anyauth
Description:
Entries:
User sec_master TcmdbsvaBRrxl
Any-other Tr
Unauthenticated T
Group webseal-servers Tgmdbsrxl
Group iv-admin TcmdbsvaBRrxl
```

7. Attach the one-time password extended authentication ACL to the authentication endpoint with the following command:

```
pdadmin sec_master> acl attach /WebSEAL/<WEBSEAL_INSTANCE_ROOT>
/FIM/sps/auth xauth_anyauth
```

   For example:

```
pdadmin sec_master> acl attach /WebSEAL/localhost-webseald-ip
/FIM/sps/auth xauth_anyauth
```

8. Restart WebSEAL with the command `pdweb restart`.

What to do next

Verify your extended authentication one-time password configuration.

Verifying the one-time password extended authentication configuration

Verify your extended authentication configuration to ensure that the one-time password implementation works correctly.

About this task

Complete the steps that are provided to verify that your one-time password configuration in the extended authentication flow is configured properly. The commands that are provided in the steps must be used in `pdadmin`. For more information about the `pdadmin` command-line utility, see the Tivoli information center.

Procedure

1. Create a test user account. For example:

```
pdadmin> user create john cn=john,o=ibm,c=us John Doe password
```

2. Activate the account. For example:

```
pdadmin> user modify john account-valid yes
```

3. Create a test resource that is protected with level 2 authentication and place it in the document root of WebSEAL. For example: `/opt/pdweb/www-default/docs/test.html`.
4. Try accessing that resource through WebSEAL. For example: `https://idp.example.com/test.html`. You are presented with a web form to enter the user name and password.
5. Enter the credential that you created in step 1. The contents of the resource is displayed.

6. Create a Protected Object Policy (POP) with a level 2 authentication. For example:

```
pdadmin> pop create level2only
pdadmin> pop modify level2only set ipauth anyothernw 2
```

7. Attach the POP to the protected resource that you created in step 3 on page 641. For example:

```
pdadmin> pop attach /WebSEAL/idp.example.com-default/test.html level2only
```

8. Open a new browser session and try accessing the test resource again. You are presented with a web form where you must enter a user name and password.
9. Enter the credential for the test user. You are forwarded to the extended authentication endpoint of Tivoli Federated Identity Manager. You are now starting the one-time password feature. Depending on your configuration, you might be asked to select the one-time password delivery method.
10. Enter the one-time password that was delivered in the delivery method that you selected. If you authenticate successfully, you are redirected to back to the test resource and you can access the contents of that resource.

Creating your own one-time password point of contact

Tivoli Federated Identity Manager version 6.2.2, fix pack 4 provides a one-time password WebSEAL point of contact profile. However, you can create your own point of contact.

Before you begin

You must do the following steps before you can add your point of contact server to your environment:

- Publish any custom point of contact callback plug-ins.
- Know what type of parameters you must use, if any, and the corresponding values to be passed to these callbacks. See "One-time password configuration overview" on page 632.

Procedure

1. Log on to the Integrated Solutions Console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Point of Contact**.
3. Select your point of contact.
   - **WebSEAL with OTP**
4. Click **Create Like**. The Point of Contact Profile wizard opens.
5. Click **Next**. The Profile Name panel opens.
6. Enter a name for the profile.
7. Optional: Enter a description.
8. Click **Next**. The Sign-in panel opens.
9. Specify the sign-in callbacks to use, the order in which these callbacks are used, and the parameters to use with each callback.

   Because you created a point of contact that is based on an existing point of contact, the callbacks and their parameters are automatically populated.

   To add or remove callbacks, click **Add** or **Remove**. The values in the Callbacks in Use list are the ones that are used with your new point of contact.
10. Click **Next**. The Sign-Out panel opens.

11. Specify the sign-out callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback.

    Because you created a point of contact that is based on an existing point of contact, the callbacks and their parameters are automatically populated.

    To add or remove callbacks, click **Add** or **Remove**. The values in the Callbacks in Use list are the ones that are used with your new point of contact.
12. Click **Next**. The Local ID panel opens.
13. Specify the callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback.

    Because you created a point of contact that is based on an existing point of contact, the callbacks and their parameters are automatically populated.

    To add or remove callbacks, click **Add** or **Remove**. The values in the Callbacks in Use list are the ones that are used with your new point of contact.
14. Click **Next**. The Authentication panel opens.
15. Specify the callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback.

    Because you created a point of contact that is based on an existing point of contact, the callbacks and their parameters are automatically populated.

    To add or remove callbacks, click **Add** or **Remove**. The values in the Callbacks in Use list are the ones that are used with your new point of contact.

    a. Change the default parameters for the **otpAuthenticateCallback**.
       - **authentication.level** - This parameter specifies the authentication level of the callback. The value must be an integer.
       - **config.federation.name**- This parameter specifies the name of the one-time password federation that is used by the callback. This one-time password federation must exist.
16. Click **Next**. The Authentication Policy panel opens.
17. Specify the callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback.

    Because you created a point of contact that is based on an existing point of contact, the callbacks and their parameters are automatically populated.

    To add or remove callbacks, click **Add** or **Remove**. The values in the Callbacks in Use list are the ones that are used with your new point of contact.

    a. Change the default parameters for the **genericPocAuthenPolicyCallback**.
       - **allow.authentication.policy.request.overrides** - This parameter specify whether the callback must use the authentication policy, which includes the authentication level, authentication mode, and authentication type that is specified in the query string instead of its own authentication policy. The value must be a boolean. If this parameter is not specified, the default value, which is false, is used.
       - **authentication.level** - This parameter specifies the authentication level of the callback. The value must be an integer. If this parameter is not specified, the default value, which is 2, is used.
       - **authentication.mode** - This parameter specifies the authentication mode of the callback. The value must be either *INDIVIDUAL* or *GROUPAL*. If this parameter is not specified, the default value, which is *INDIVIDUAL*, is used.
       - **authentication.type** - This parameter specifies the authentication type of the callback. The value must be either *COMPLEMENTARY* or

HIERARCHICAL. If this parameter is not specified, the default value, which is *COMPLEMENTARY*, is used.

b. Optional: Upload the **AuthenticationPolicyCallback** mapping rule. See "Authentication policy mapping rule customization" on page 663 for more details.

   To upload the mapping rule:

   1) Click **Add Rule**.
   2) Click **Modify Rule**.
   3) Click **Browse** to find the file on the system.
   4) Click **Import File**.
   5) Click **OK**. The identity mapping rule file is applied.

18. Click **Next**. The summary panel is displayed.
19. Click **Finish**.
20. Click **Load configuration changes to the Tivoli Federated Identity Manager runtime**.

What to do next

You must activate your newly-created point of contact.

HTTP request claims for authentication policy callback

Configure the authentication policy callback so that HTTP request information is available during the mapping rule execution for the context-based authentication policy determination.

To enable the HTTP request parameters for the context-based authentication policy, you must configure the following parameter:

SPS.http.request.claims.enabled

When set to true, this parameter enables the Secure Protocol Service (SPS) to include a WS-Trust claims element. The WS-Trust claims element is included on the WS-Trust request to the Security Token Service. The claims element contains all the HTTP request information that is received at the SPS that causes the call to the Security Token Service. To avoid XML parsing problems, the values from the request are XML encoded before they are included as values to the claims element structure. The following HTTP request information is included in the claims element:

- Cookies
- HTTP headers
- HTTP request attributes
- HTTP request parameters

Configuration example: `SPS.http.request.claims.enabled=true`
Default value: False

- Value type: Boolean
- Example value: True

The request cookies, headers, and parameters in an HTTP request might be numerous and result in a large claims element. You can filter for request cookies, headers, and parameters by using a custom property. Use the following custom property to avoid including information that cannot not be processed by the Authentication Policy callback:

SPS.http.request.claims.filter.spec

Use this callback parameter to specify the request cookies, headers, and parameters to include in the claims element.

For each data type, you can choose to add all values or filter the values that are based on the item name.

The default filter: **cookies=*:headers=***

The default filter causes all cookies and headers to be included and excludes all parameters.

The format for the filter specification syntax:

cookies=[*|cookieName1,cookieName2]:

headers=[*|header1,header2]: parameters=[*|param1,param2]

Note:

- To filter for a specific element, define the custom property with the specific element on the data type to which it belongs. For example, if you want to receive a cookie that is called `MyCookie`, specify the filter as:

  ```
  cookies=MyCookie
  ```

  To retrieve all cookies in the request but exclude all parameters and headers, set the custom property to `cookies=*`.
- The header, cookies, and parameters can be multi-valued.
- The cookie value includes the actual cookie value, the domain, and the path that is separated by `;`. For example, a cookie named `MyCookie` with value of `MyValue`, path of `/`, and domain of `my.domain` is formatted on the XML document as follows:

  ```
  <Cookie Name="MyCookie" Type="urn:ibm:names:ITFIM:httprequest:cookies">
      <Value>MyValue; %2F; my.domain</Value>
  </Cookie>
  ```

An example of using the custom property to enable all the cookies, headers, and parameters:

```
cookies=*:headers=*:parameters=*
```

The resulting `HTTPRequestClaims` element:

```
<HTTPRequestClaims xmlns="urn:ibm:names:ITFIM:httprequest">
 <Attributes>

  <Attribute Name="remoteAddress"
   Type="urn:ibm:names:ITFIM:httprequest:remoteAddress">
   <Value>127.0.0.1</Value>
  </Attribute>
  <Attribute Name="remoteHost" Type="urn:ibm:names:
      ITFIM:httprequest:remoteHost">
   <Value>fim620</Value>

  </Attribute>
  <Attribute Name="protocol" Type="urn:ibm:names:ITFIM:
    httprequest:protocol">
   <Value>HTTP</Value>
  </Attribute>
  <Attribute Name="method" Type="urn:ibm:names:ITFIM:
      httprequest:method">

   <Value>GET</Value>
  </Attribute>
```

```
 <Attribute Name="pathInfo" Type="urn:ibm:names:ITFIM:
     httprequest:pathInfo">
  <Value>/xauth</Value>
 </Attribute>

 <Attribute Name="queryString"
  Type="urn:ibm:names:ITFIM:httprequest:queryString">
  <Value>Target=https://idp.fim.demo.com</Value>
 </Attribute>
 <Attribute Name="requestURI" Type="urn:ibm:names:
     ITFIM:httprequest:requestURI">
  <Value>/sps/xauth</Value>

 </Attribute>
 <Locales>
  <Locale Name="locales" Type="urn:ibm:names:
      ITFIM:httprequest:locales">
   <Value>en_US</Value>
   <Value>en</Value>

  </Locale>
 </Locales>
</Attributes>
<Headers>
 <Header Name="iv-creds" Type="urn:ibm:names:ITFIM:
   httprequest:headers">
  <Value>Version=1,
   BAKs3DCCBO0MADCCBOcwggT....WgQA
  </Value>

 </Header>
 <Header Name="keep-alive" Type="urn:ibm:names:ITFIM:
   httprequest:headers">
  <Value>115</Value>
 </Header>
 <Header Name="accept-charset" Type="urn:ibm:names:
   ITFIM:httprequest:headers">

  <Value>ISO-8859-1,utf-8;q=0.7,*;q=0.7</Value>
 </Header>
 <Header Name="accept" Type="urn:ibm:names:ITFIM:
  httprequest:headers">
  <Value>text/html,application/xhtml+xml,
     application/xml;q=0.9,*/*;q=0.8
  </Value>
 </Header>

 <Header Name="host" Type="urn:ibm:names:ITFIM:
   httprequest:headers">
  <Value>fim620:9081</Value>
 </Header>
 <Header Name="iv-user" Type="urn:ibm:names:
   ITFIM:httprequest:headers">
  <Value>Unauthenticated</Value>

 </Header>
 <Header Name="referer" Type="urn:ibm:names:ITFIM:
   httprequest:headers">
  <Value>https://saml20ip/FIM/sps/saml20ip/saml20/
      login?SAMLRequest=nVNdT8IwFP0rS....d%2FmV928%3D
  </Value>
 </Header>
 <Header Name="via" Type="urn:ibm:names:ITFIM:
   httprequest:headers">

  <Value>HTTP/1.1 fim620:444</Value>
 </Header>
```

```
  <Header Name="content-type" Type="urn:ibm:names:
    ITFIM:httprequest:headers">
   <Value>application/x-www-form-urlencoded</Value>
  </Header>

  <Header Name="iv-groups" Type="urn:ibm:names:ITFIM:
    httprequest:headers">
   <Value />
  </Header>
  <Header Name="iv_server_name" Type="urn:ibm:names:
    ITFIM:httprequest:headers">
   <Value>webseald-sp-webseald-localhost</Value>

  </Header>
  <Header Name="content-length" Type="urn:ibm:names:
    ITFIM:httprequest:headers">

   <Value>6245</Value>
  </Header>
  <Header Name="accept-language" Type="urn:ibm:names:
    ITFIM:httprequest:headers">
   <Value>en-us,en;q=0.5</Value>
  </Header>

  <Header Name="connection" Type="urn:ibm:names:ITFIM:
    httprequest:headers">
   <Value>close</Value>
  </Header>
 </Headers>
 <Cookies>
  <Cookie Name="jsessionid" Type="urn:ibm:names:ITFIM:
    httprequest:cookies">
   <Value>0000ZOelYEj9RH1aQVymcofXoKc:-1</Value>
  </Cookie>
  <Cookie Name="iv_jct" Type="urn:ibm:names:
     ITFIM:httprequest:cookies">

   <Value>%2FFIM</Value>
  </Cookie>
 </Cookies>
 <Parameters>
  <Parameter Name="Target"
   Type="urn:ibm:names:ITFIM:httprequest:query:param">
   <Value>https://idp.fim.demo.com</Value>
  </Parameter>
 </Parameters>
</HTTPRequestClaims>
```

Note: The parameter attribute type value indicates whether the parameter was received in the query string or as part of the request body. For query string parameters, the type is set to `urn:ibm:names:ITFIM:httprequest:query:param`. For parameters received as part of the request body, the value is set to `urn:ibm:names:ITFIM:httprequest:body:param`.

In the example, the cookies, headers, and parameters are filtered according to the specified values.

This example filters the `jsessionid` cookie, host header, and `RelayState` parameter:

```
cookies=jsessionid:headers=host:parameters=Target
```

Note: The values that are specified for parameters are case-sensitive. The values for cookies and headers are not case-sensitive.

The resulting HTTPRequestClaims element:

```
<HTTPRequestClaims xmlns="urn:ibm:names:ITFIM:httprequest">
 <Attributes>
  <Attribute Name="remoteAddress"
   Type="urn:ibm:names:ITFIM:httprequest:remoteAddress">

   <Value>127.0.0.1</Value>
  </Attribute>
  <Attribute Name="remoteHost"
    Type="urn:ibm:names:ITFIM:httprequest:remoteHost">
   <Value>fim620</Value>
  </Attribute>

  <Attribute Name="protocol"
   Type="urn:ibm:names:ITFIM:httprequest:protocol">
   <Value>HTTP</Value>
  </Attribute>
  <Attribute Name="method"
    Type="urn:ibm:names:ITFIM:httprequest:method">
   <Value>GET</Value>

  </Attribute>
  <Attribute Name="pathInfo"
    Type="urn:ibm:names:ITFIM:httprequest:pathInfo">
   <Value>/xauth</Value>
  </Attribute>
  <Attribute Name="queryString"
   Type="urn:ibm:names:ITFIM:httprequest:queryString">

   <Value>Target=https://idp.fim.demo.com</Value>
  </Attribute>
  <Attribute Name="requestURI"
    Type="urn:ibm:names:ITFIM:httprequest:requestURI">
   <Value>/sps/xauth</Value>
  </Attribute>

  <Locales>
   <Locale Name="locales"
      Type="urn:ibm:names:ITFIM:httprequest:locales">
    <Value>en_US</Value>
    <Value>en</Value>
   </Locale>

  </Locales>
 </Attributes>
 <Headers>
  <Header Name="host"
   Type="urn:ibm:names:ITFIM:httprequest:headers">
   <Value>fim620:9081</Value>

  </Header>
 </Headers>
 <Cookies>
  <Cookie Name="jsessionid"
    Type="urn:ibm:names:ITFIM:httprequest:cookies">
   <Value>0000sOnmzkbGcYdIcevoYRuxq0m:-1</Value>

  </Cookie>
 </Cookies>
 <Parameters>
  <Parameter Name="Target"
   Type="urn:ibm:names:ITFIM:httprequest:query:param">
   <Value>https://idp.fim.demo.com</Value>
  </Parameter>
 </Parameters>
</HTTPRequestClaims>
```

An example HTTPRequestClaims as shown in the STSUUSER during the execution of the authentication policy mapping:

```
<stsuuser:ContextAttributes>

.......
    <stsuuser:Attribute name="HTTPRequestClaims"
type="urn:ibm:names:ITFIM:httprequest">

        <stsuuser:Value>
            <HTTPRequestClaims xmlns="urn:ibm:names:ITFIM:httprequest">
                    ........
            </HTTPRequestClaims>
        </stsuuser:Value>
    </stsuuser:Attribute>
.......
</stsuuser:ContextAttributes>
```

One-time password resend support

The one-time password login template page provides a regenerate and a reselect button. Clicking the regenerate button allows users to regenerate a new one-time password if the one-time password value is lost or not received. Clicking the reselect button allows users to reselect their preferred delivery method.

Clicking the regenerate button causes the following actions:

1. Invalidate the old one-time password,
2. Generate a new one-time password value, and
3. Use the same delivery mechanism that is used in the previous delivery attempt to deliver the new value.

Clicking the reselect button causes the following actions:

1. Invalidate the old one-time password.
2. Regenerate the list of methods to generate, deliver, and verify the one-time password.
3. Redisplay the one-time password method selection page. The user can then reselect the method to generate, deliver, and verify the one-time password.

Configuring an unauthenticated one-time password flow

You can run the one-time password flow without requiring prior user authentication.

About this task

Before Tivoli Federated Identity Manager Fix Pack 5, the one-time password that is relied on the /sps/auth URL to move from a phase of the flow to the other. The /sps/auth URL is also used to force authentication on environments that use the WebSEAL point of contact. This approach made it impossible to run a one-time password flow where the user is not authenticated before the flow is run.

With the Tivoli Federated Identity Manager Fix Pack 5, you can run the one-time password flow without requiring prior user authentication.

The default behavior is that this feature is enabled.

To disable this feature, set the custom property `SPS.POC.use.legacy.auth.url` to `true`. This configuration sets Tivoli Federated Identity Manager to behave as it did before Fix Pack 5.

This feature requires several configuration steps on WebSEAL. The `tfimcfg` tool is enabled to do the necessary configuration when a single sign-on federation is configured.

You must rerun the tool to make the necessary configuration changes.

For environments that use the extended authentication mode the following steps must be done manually.

Procedure

1. Configure a one-time password federation.
2. Add the `/sps/authservice/authentication` URL to the EAI trigger URLs.

   ```
   # EAI TRIGGER URLS
   [eai-trigger-urls]
   trigger = /FIM/sps/authservice/authentication*
   ```

3. Attach the unauthenticated ACL to the `/sps/authservice/authentication` URL.

   ```
   pdadmin sec_master> acl attach /WebSEAL/<WEBSEAL_INSTANCE_ROOT>/FIM/sps/authservice/authentication xauth_unauth
   ```

   For example:

   ```
   pdadmin sec_master> acl attach /WebSEAL/localhost-webseald-ip/FIM/sps/authservice/authentication xauth_unauth
   ```

Migrating one-time password files into an existing environment

Update one-time password files to use time-based and counter-based one-time passwords. Make this migration after you upgrade to Tivoli Federated Identity Manager, version 6.2.2, Fix pack 7, from version 6.2.2, Limited Availability (LA) 5, or later.

About this task

Tivoli Federated Identity Manager, version 6.2.2, Fix pack 7, installation does not overwrite one-time password pages and mapping rules when you upgrade from LA 5, or later.

If you upgrade from Fix pack 4, one-time password files are overwritten. Therefore, you do not need to complete this procedure to use time-based or counter-based one-time passwords.

Procedure

1. Update HTML pages for a time-based and counter-based one-time password:
   a. Browse to *FIM_INSTALL_DIR*`/pages_template/`*LOCALE*`/otp/`.
   b. Locate the following HTML page samples and check if updates apply to your environment:

Updated HTML pages	Updates in 6.2.2.7
login.html	• @OTP_METHOD_TYPE@ macro can hide the **Regenerate** button. • If login attempt limits are enabled and exceeded, a @OTP_LOGIN_DISABLED@ macro that is set in the otp_verify.js mapping rule can disable **Submit** when it is used inside an HTML input tag.
delivery_selection.html	Minor descriptive text change.

c. Update the file of the same name in *FIM_INSTALL_DIR*/pages/*LOCALE*/otp/ by taking one of the following actions:
   - Copy the new HTML file over the existing file if no customization was done to the file.
   - Merge changes from the new HTML file into the existing HTML file to retain custom content.

2. Update your existing mapping rules with the information in the sample mapping rules:
   a. Browse to *FIM_INSTALL_DIR*/examples/js_mappings/.
   b. Locate the following mapping rule files and check if updates apply to your environment:

Updated mapping rules	Updates in 6.2.2.7
otp_get_delivery_methods.js	• The userInfoType attribute specifies the User Info Provider for time-based and counter-based passwords. If unspecified or set to an empty string, no provider is used. • New counter-based, one-time password method was added. • New time-based, one-time password method was added.
otp_deliver.js	When the delivery method is no_delivery, a check is done to provide no one-time password hint.
otp_verify.js	• Different classes are now imported. • A native login attempt limit exception is used, not one with IDMappingExtUtils. • A user-defined @OTP_LOGIN_DISABLED@ macro can disable **Submit** in login.html when the login attempt limit is exceeded.

   c. Merge changes from the new mapping rules into the existing mapping rules to retain any custom content.

Customizing one-time password

Edit the mapping rules and template pages to customize one-time password.

Customizing one-time password mapping rules

Edit the one-time password mapping rules to customize the processing of one-time passwords.

The sample mapping rules are in $FIM_INSTALL_DIR$/examples/js_mappings. The following one-time password mapping rules are available:

- otp_get_delivery_methods.js - This file contains the OTPGetDeliveryMethodsMappingRule.
- otp_generate.js- This file contains the OTPGenerateMappingRule.
- otp_deliver.js - This file contains the OTPDeliverMappingRule.
- otp_verify.js This file contains the OTPVerifyMappingRule.

You can customize the following one-time password mapping rules:

- "OTPDeliver mapping rule"
- "OTPGenerate mapping rule" on page 653
- "OTPGetDeliveryMethods mapping rule" on page 653
- "OTPVerify mapping rule" on page 655

OTPDeliver mapping rule

The OTPDeliver mapping rule is a mapping rule that is executed when Tivoli® Federated Identity Manager delivers the one-time password to the user.

Use the following OTPDeliver mapping rules:

Generate the one-time password hint

The one-time password hint is a sequence of characters that is associated with the one-time password. Tivoli Federated Identity Manager uses the one-time password hint to inform which one-time password the user must submit. The one-time password hint is displayed in the One-Time Password Login page. It is also sent to the user together with the one-time password.

You can customize the way the one-time password hint is generated by modifying the following section in the default OTPDeliver mapping rule:

```
var otpHint = Math.floor(1000 + (Math.random() * 9000));
```

Note: See the comments in the mapping rule for more details.

Generate the formatted one-time password

The formatted one-time password is the formatted version of the one-time password. Tivoli Federated Identity Manager sends the formatted one-time password, instead of the actual one-time password, to the user. For example, for one-time password hint abcd, and one-time password 12345678, you can set the formatted one-time password as abcd-12345678. For one-time time password hint efgh, and one-time password87654321, you can set the one-time password as efgh#8765#4321.

You can customize the way that the one-time password is generated by modifying the following section in the sample OTPDeliver mapping rule:

```
var otpFormatted = otpHint + "-" + otp;
```

Note: See the comments in the mapping rule for more details.

Modify the delivery type of the selected method for delivering the one-time password

Tivoli Federated Identity Manager uses the delivery type to determine the one-time password Delivery plug-in that delivers the one-time password to the user.

Modify the delivery attribute of the selected method to deliver the Tivoli Federated Identity Manager

The delivery attribute is an attribute that is associated with delivery type. The meaning of the delivery attribute depends on the one-time password provider plug-in for the delivery type. For example, for SMS delivery type, the delivery attribute is the mobile number of the user. For email delivery type, the delivery attribute is the email address of the user.

Note: See the comments in the mapping rule for more details.

OTPGenerate mapping rule

`OTPGenerate` mapping rule is a mapping rule that is executed when Tivoli Federated Identity Manager generates the one-time password for the user.

You can use the `OTPGenerate` mapping rule in the following configuration:

Modify the one-time password type of the selected method to generate the one-time password

Tivoli Federated Identity Manager uses one-time password type to determine the one-time password Provider plug-in that generates the one-time password for the user.

Note: See the comments in the mapping rule for more details.

OTPGetDeliveryMethods mapping rule

`OTPGetDeliveryMethods` is a mapping rule that runs when the one-time password response file retrieves the methods for delivering the one-time password to the user.

This sample mapping rule sets password delivery conditions for the following delivery methods:

- By email
- By SMS
- No delivery

Each delivery method includes the following attributes and their corresponding value:

id Specifies a unique delivery method ID. This value replaces the `@OTP_METHOD_ID@` macro in the OTP Method Selection page. Use a unique value across different methods. For example, `sms`.

deliveryType

Specifies the delivery plug-in that delivers the one-time password. The value must match one of the types in the `DeliveryTypesToOTPDeliveryModuleIds` parameter of the OTP response file. For example, `sms_delivery`.

deliveryAttribute

Specifies an attribute that is associated with the delivery type. The value depends on the one-time password provider plug-in for the delivery type. For example:

- For SMS delivery, the value is the mobile number of the user. For example, `mobileNumber`.
- For email delivery, the value is the email address of the user. For example, `emailAddress`.
- For no delivery, the value is an empty string.

label Specifies the unique delivery method to the user. For time-based and counter-based one-time password, use this attribute to specify the secret key of the user. If `label` is not specified, the time-based and counter-based one-time password code retrieves the key by invoking the user information provider plug-in. This parameter replaces the `@OTP_METHOD_LABEL@` macro in the OTP Method Selection page.

otpType
Specifies the one-time password provider plug-in that generates and verifies the password. The value must match one of the types in the `OTPTypesToOTPProviderModuleIds` parameter of the OTP response file. For example, `mac_otp`.

userInfoType
Specifies which user information provider plug-in to use to retrieve user information that is required to calculate the one-time password. This parameter is only required if user information is used for calculation of the one-time password.

To customize one-time password delivery, you can do one of the following actions:

- Create your own mapping rules that are based on the sample `OTPGetDeliveryMethods` mapping rule.
- Modify the sample `OTPGetDeliveryMethods` mapping rule.

Sample OTPGetDeliveryMethods mapping rule

```
var methods = [];

if (useSMSDelivery) {
    var mobileNumber = new java.lang.String("12345678");
    //var mobileNumber = attributeContainer.getAttributeValueByName("tagvalue_phone");
    var fomattedMobileNumber = mobileNumber;
    if (mobileNumber != null && mobileNumber.length() > 4) {
        formattedMobileNumber = mobileNumber.substring(0, mobileNumber.length() - 4) + "XXXX";
    }
    var method = {
        id: "sms",
        otpType: "mac_otp",
        deliveryType: "sms_delivery",
        deliveryAttribute: mobileNumber,
        label: "SMS to: " + formattedMobileNumber,
        userInfoType: ""
    };
    methods.push(method);
}

if (useEmailDelivery) {
    var emailAddress = new java.lang.String("username@tfim.ibm.com");
    //var emailAddress = attributeContainer.getAttributeValueByName("tagvalue_email");
    var fomattedEmailAddress = emailAddress;
    if (emailAddress != null && emailAddress.length() > 4) {
        fomattedEmailAddress = "XXXX" + emailAddress.substring(4);
    }
    var method = {
        id: "email",
        otpType: "mac_otp",
        deliveryType: "mail_delivery",
        deliveryAttribute: emailAddress,
        label: "Email to: " + fomattedEmailAddress,
        userInfoType: ""
    };
    methods.push(method);
}

if (useTOTP) {
    var method = {
        id: "totp",
        otpType: "totp_otp",
        deliveryType: "no_delivery",
        deliveryAttribute: "SECRET_KEY_GOES_HERE",
        label: "Time Based OTP",
```

```
        userInfoType: "file_userinfo"
    };
    methods.push(method);
}

if (useHOTP) {
    var method = {
        id: "hotp",
        otpType: "hotp_otp",
        deliveryType: "no_delivery",
        deliveryAttribute: "SECRET_KEY_GOES_HERE",
        label: "Counter Based OTP",
        userInfoType: "file_userinfo"
    };
    methods.push(method);
}
```

Note: See the comments in the mapping rule for more details.

OTPVerify mapping rule

`OTPVerify` is a mapping rule that runs when Tivoli Federated Identity Manager verifies the one-time password that is submitted by the user.

You can customize the sample `OTPVerify` mapping rule to modify the following verification rules:

Modify the one-time password type of the user

Tivoli Federated Identity Manager uses the one-time password type to determine the one-time Provider plug-in that verifies the one-time submitted by the user.

Set the authentication level of the user

After one-time password authentication completes, a credential is issued that contains the authentication level of the user. You can customize the authentication level by modifying the following section in the mapping rule:

```
var authenticationLevel = contextAttributesAttributeContainer.getAttributeValueByNameAndType
      ("otp.otp-callback.authentication-level", "otp.otp-callback.type");
var attributeAuthenticationLevel = new Attribute("AUTHENTICATION_LEVEL",
      "urn:ibm:names:ITFIM:5.1:accessmanager", authenticationLevel);
attributeContainer.setAttribute(attributeAuthenticationLevel);
```

Enforce the number of times the user can submit the one-time password in the one-time password login page

If a user exceeds the permitted number of times to submit a one-time password, an error message displays. You can customize the number of times that the user can submit the one-time password in the one-time password login page by modifying the following section in the mapping rule:

```
var retryLimit = 5;
```

By default, this option is set to `false`.

Identify the secret key of a user

When a user registers with a time-based one-time password application, they are assigned a secret key. Store the secret key in this mapping rule for verification of the user by modifying the following code:

```
var secretStr = new java.lang.String(SECRET_KEY_GOES_HERE);
```

By default, this option is set to `false`.

To customize one-time password verification, you can do one of the following actions:

- Create your own verification rules that are based on the sample `OTPVerify` mapping rule.
- Modify the sample `OTPVerify` mapping rule.

Customizing one-time password template pages

Edit the one-time password template pages to customize the look of the pages that Tivoli Federated Identity Manager displays to the user and to customize the content of the SMS and email that the `SMSOTPDelivery` and `EmailOTPDelivery` sends to the users. The `SMSOTPDelivery` and `EmailOTPDelivery` are the one-time password delivery plug-ins that are shipped with Tivoli Federated Identity Manager.

The following one-time password template pages are available in `$FIM_INSTALL_DIR$/pages/$LOCALE$/otp`:

- `login.html`- The one-time password template page for login.
- `delivery_selection.html`- The one-time password template page for delivery selection.
- `errors/allerror.html`- The one-time password template page for general errors.
- `errors/error_could_not_validate_otp.html`- Thee one-time password template page for one-time password validation error.
- `errors/error_generating_otp.html`- The one-time password template page for generating one-time password error.
- `errors/error_get_delivery_options.html`- The one-time password template page for get delivery error.
- `errors/error_otp_delivery.html`- The one-time password template page for delivery error.
- `errors/error_sts_invoke_failed.html`- The one-time password template page for security trust service operation error.
- `delivery/sms_message.xml`- The one-time password template page for SMS.
- `delivery/email_message.xml`- The one-time password template page for email.

After you modify the pages, you must publish the pages for the changes to take effect. See Publishing pages to the Tivoli Federated Identity runtime environment.

One-time password template page for login

This template page is used by Tivoli Federated Identity Manager to display the form where the user can enter the one-time password. The page is also called One-Time Password Login page.

The template has the following replacement macros:

@ERROR_MESSAGE@
: This macro is replaced with a message that indicates that the submitted one-time password contains errors. Examples of these errors are that the submitted one-time password is not valid and the one-time password is submitted after it expires.

@MAPPING_RULE_DATA@
: If the submitted one-time password contains an error, this macro is replaced with the value of the STS Universal User context attribute whose name is `@MAPPING_RULE_DATA@` and type is `otp.sts.macro.type`. This context attribute can be set in the "OTPVerify mapping rule" on page 655.

@OTP_HINT@
This macro is replaced with one-time password hint. The one-time password hint is a sequence of characters that is associated with the one-time password. It is used by Tivoli Federated Identity Manager to inform which one-time password that the user must submit.

@REGENERATE_ACTION@
This macro is replaced with the URL where the **Generate** button posts the form to regenerate and deliver the new one-time password value.

@RESELECT_ACTION@
This macro is replaced with the URL where the **Reselect** button posts the form to reselect the method for generating, delivering, and verifying the one-time password value.

@OTP_METHOD_TYPE@
This macro is replaced by the type of the currently selected method for generating, delivering, and verifying the one-time password. This type is generated by `OTPGetDeliveryMethods` mapping rule and was selected by the user.

One-time password template page for delivery selection

This template page displays the list of methods for generating, delivering, and verifying the one-time password. It is also called One-Time Password Method Selection page.

@OTP_METHOD_ID@
This macro is replaced by the ID of the method for generating, delivering, and verifying the one-time password. This ID is generated by `OTPGetDeliveryMethods` mapping rule.

@OTP_METHOD_LABEL@
This macro is replaced by the label of the method for generating, delivering, and verifying the one-time password. This label is generated by `OTPGetDeliveryMethods` mapping rule.

@OTP_METHOD_CHECKED@
For the first method, this macro is replaced with an HTML radio button attribute that causes that radio button to be selected. For the remaining methods for generating, delivering, and verifying, this macro is replaced with an empty string.

One-time password template page for general errors

This template page is used by Tivoli Federated Identity Manager to display general errors that happen during the one-time password flow. General errors are errors that are not displayed in other template pages.

The template has the following replacement macros:

@REQ_ADDR@
This macro is replaced with the URL into which the request from the user is sent.

@TIMESTAMP@
This macro is replaced with the timestamp when the error occurred.

@DETAIL@
This macro is replaced with the error message.

@EXCEPTION_STACK@
This macro is replaced with the stack trace of the error.

Figure 70. Template for allerror.html

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>One-Time Password Error</title>
</head>

<body style="background-color: #ffffff">
  <div>
    <h2 style="color: #ff8800">An error has occurred.</h2>
    <div id="infoDiv" style="background-color: #ffffff; color: #000000">
      <em>@REQ_ADDR@</em><br />
      <em>@TIMESTAMP@</em> <br />
    </div>
    <br />
    <div id="detailDiv" style="background-color: #999999;
     border-style: solid; border-width: 1px; border-color: #000000">
      <h4>Error details</h4>
      @DETAIL@
    </div>
    <br />
    <div id="stackDiv" style="background-color: #999999;
     border-style: solid; border-width: 1px; border-color: #000000">
      <h4>Stack trace</h4>
      @EXCEPTION_STACK@
    </div>
  </div>
</body>
</html>
```

One-time password template page for generating one-time password error

This template page is used by Tivoli Federated Identity Manager to display errors that happen when Tivoli Federated Identity Manager generates one-time password.

The template has the following replacement macros:

@REQ_ADDR@
: This macro is replaced with the URL into which the request from the user is sent.

@TIMESTAMP@
: This macro is replaced with the timestamp when the error occurred.

@DETAIL@
: This macro is replaced with the error message.

@EXCEPTION_STACK@
: This macro is replaced with the stack trace of the error.

Figure 71. Template for error_generating_otp.html

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
```

```
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>One-Time Password Error</title>
</head>

<body style="background-color: #ffffff">
  <div>
    <h2 style="color:#ff8800">An error occurred while
     generating the one-time password.</h2>
    <div id="infoDiv" style="background-color: #ffffff; color: #000000">
      <em>@REQ_ADDR@</em><br />
      <em>@TIMESTAMP@</em><br />
    </div>
    <br />
    <div id="detailDiv" style="background-color: #999999;
      border-style: solid; border-width: 1px; border-color: #000000">
      <h4>Error details</h4>
      @DETAIL@
    </div>
    <br />
    <div id="stackDiv" style="background-color: #999999;
      border-style: solid; border-width: 1px; border-color: #000000">
      <h4>Stack trace</h4>
      @EXCEPTION_STACK@
    </div>
  </div>
</body>
</html>
```

One-time password template page for get delivery error

This template page is used by Tivoli Federated Identity Manager to display errors that happen when Tivoli Federated Identity Manager retrieves the list of methods for delivering one-time password to the user.

The template has the following replacement macros:

@REQ_ADDR@
: This macro is replaced with the URL into which the request from the user is sent.

@TIMESTAMP@
: This macro is replaced with the timestamp when the error occurred.

@DETAIL@
: This macro is replaced with the error message.

@EXCEPTION_STACK@
: This macro is replaced with the stack trace of the error.

Figure 72. Template for `error_get_delivery_options.html`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>One-Time Password Error</title>
</head>

<body style="background-color: #ffffff">
  <div>
    <h2 style="color: #ff8800">An error occurred while
    obtaining the one-time password delivery options.</h2>
```

```
    <div id="infoDiv" style="background-color: #ffffff; color: #000000">
      <em>@REQ_ADDR@</em> <br />
      <em>@TIMESTAMP@</em> <br />
    </div>
    <br />
    <div id="detailDiv" style="background-color: #999999;
     border-style: solid; border-width: 1px; border-color: #000000">
      <h4>Error details</h4>
      @DETAIL@
    </div>
    <br />
    <div id="stackDiv" style="background-color: #999999;
     border-style: solid; border-width: 1px; border-color: #000000">
      <h4>Stack trace</h4>
      @EXCEPTION_STACK@
    </div>
  </div>
</body>
</html>
```

One-time password template page for delivery error

This template page is used by Tivoli Federated Identity Manager to display errors that happen when Tivoli Federated Identity Manager delivers one-time password to user.

The template has the following replacement macros:

@REQ_ADDR@
: This macro is replaced with the URL into which the request from the user is sent.

@TIMESTAMP@
: This macro is replaced with the timestamp when the error occurred.

@DETAIL@
: This macro is replaced with the error message.

@EXCEPTION_STACK@
: This macro is replaced with the stack trace of the error.

Figure 73. Template for `error_otp_delivery.html`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>One-Time Password Error</title>
</head>

<body style="background-color:#ffffff">
  <div>
    <h2 style="color:#ff8800">An error occurred while
    delivering the one-time password value.</h2>
    <div id="infoDiv" style="background-color: #ffffff; color: #000000">
      <em>@REQ_ADDR@</em><br />
      <em>@TIMESTAMP@</em><br />
    </div>
    <br />
    <div id="detailDiv" style="background-color: #999999;
     border-style: solid; border-width: 1px; border-color: #000000">
      <h4>Error details</h4>
```

```
      @DETAIL@
    </div>
    <br />
    <div id="stackDiv" style="background-color: #999999;
     border-style: solid; border-width: 1px; border-color: #000000">
      <h4>Stack trace</h4>
      @EXCEPTION_STACK@
    </div>
  </div>
</body>
</html>
```

One-time password template page for security trust service operation error

This template page is used by Tivoli Federated Identity Manager to display errors that happen when Tivoli Federated Identity Manager invokes the Security Token Service.

The template has the following replacement macros:

@REQ_ADDR@
: This macro is replaced with the URL into which the request from the user is sent.

@TIMESTAMP@
: This macro is replaced with the timestamp when the error occurred.

@DETAIL@
: This macro is replaced with the error message.

@EXCEPTION_STACK@
: This macro is replaced with the stack trace of the error.

Figure 74. Template for `error_sts_invoke_failed.html`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>One-Time Password Error</title>
</head>

<body style="background-color: #ffffff">
  <div>
    <h2 style="color: #ff8800">An error occurred while
    invoking the trust service to perfom a one-time password operation.</h2>
    <div id="infoDiv" style="background-color: #ffffff; color: #000000">
      <em>@REQ_ADDR@</em><br />
      <em>@TIMESTAMP@</em><br />
    </div>
    <br />
    <div id="detailDiv" style="background-color: #999999;
     border-style: solid; border-width: 1px; border-color: #000000">
      <h4>Error details</h4>
      @DETAIL@
    </div>
    <br />
    <div id="stackDiv" style="background-color: #999999;
    border-style: solid; border-width: 1px; border-color: #000000">
      <h4>Stack trace</h4>
      @EXCEPTION_STACK@
```

```
    </div>
  </div>
</body>
</html>
```

One-time password template page for one-time password validation error

This template page is used by Tivoli Federated Identity Manager to display errors that happen when Tivoli Federated Identity Manager validates the one-time password that the user submits.

The template has the following replacement macros:

@REQ_ADDR@
: This macro is replaced with the URL into which the request from the user is sent.

@TIMESTAMP@
: This macro is replaced with the timestamp when the error occurred.

@DETAIL@
: This macro is replaced with the error message.

@EXCEPTION_STACK@
: This macro is replaced with the stack trace of the error.

Figure 75. Template for `error_could_not_validate_otp.html`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
  <title>One-Time Password Error</title>
</head>

<body style="background-color:#ffffff">
  <div>
    <h2 style="color: #ff8800">The one-time password
     value could not be validated.</h2>
    <div id="infoDiv" style="background-color: #ffffff; color: #000000">
      <em>@REQ_ADDR@</em> <br />
      <em>@TIMESTAMP@</em> <br />
    </div>
    <br />
    <div id="detailDiv" style="background-color: #999999;
    border-style: solid; border-width: 1px; border-color: #000000">
      <h4>Error details</h4>
      @DETAIL@
    </div>
    <br />
    <div id="stackDiv" style="background-color: #999999;
    border-style: solid; border-width: 1px; border-color: #000000">
      <h4>Stack trace</h4>
      @EXCEPTION_STACK@
    </div>
  </div>
</body>
</html>
```

One-time password template page for Short Message Service (SMS)

This template page is used by `SMSOTPDelivery` as the content of the SMS that it sends to the user.

The template has the following replacement macro:

@OTP_STRING@
: This macro is replaced with the one-time password generated by the one-time password provider plug-in.

Figure 76. Template page for `sms_message.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
<Message>
  <Value>
    This is your one-time password @OTP_STRING@.

    Thank you,
    OTP Test
  </Value>
</Message>
</root>
```

One-time password template page for email

This template page is used by `EmailOTPDelivery` as the content of the email that it sends to the user.

The template has the following replacement macro:

@OTP_STRING@
: This macro is replaced with the one-time password generated by the one-time password provider.

Figure 77. Template for `email_message.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
<Subject>
  <Value>
    One-time password
  </Value>
</Subject>

<Message>
  <Value>
    This is your one-time password @OTP_STRING@.

    Thank you,
    OTP Test
  </Value>
</Message>
</root>
```

Authentication policy mapping rule customization

The authentication policy mapping rule determines the authentication policy that is based on the request context.

An authentication policy is a set of rules that are applied to the authentication process and to the verification of authentication data. The authentication policy enforcement is determined based on the request context.

The authentication policy consists of the required authentication level, authentication mode, and authentication type. Use a JavaScript mapping rule to determine the authentication level, authentication type, and authentication mode policy settings. See "One-time password configuration overview" on page 632 for more details.

The sample authentication policy mapping rules are in `$FIM_INSTALL_DIR$/examples/js_mappings`.

You can modify the following code in the `otp_authnpolicy.js` mapping rule.

```
// Override the authentication mode, level, and type
based on the ip address of the user
var ipAddress = stsuu.getAttributeValueByName("AZN_CRED_NETWORK_ADDRESS_STR");
if (ipAddress != null && ipAddress.indexOf("YOUR_IP_ADDRESS_PREFIX") == 0) {
    var contextAttributes = stsuu.getContextAttributes();

    var authModeAttr = new Attribute("AuthenticationMode",
     null, "INDIVIDUAL");
    contextAttributes.setAttribute(authModeAttr);

    var authLevelAttr = new Attribute("AuthenticationLevel", null, "2");
    contextAttributes.setAttribute(authLevelAttr);

    var authTypeAttr =
    new Attribute("AuthenticationType", null, "HIERARCHICAL");
    contextAttributes.setAttribute(authTypeAttr);
```

Note: See the comments in the mapping rules for more details.

Related concepts:

"One-time password overview" on page 631
Tivoli Federated Identity Manager provides various authentication mechanisms in the point of contact interface.

Creating user-defined macros

The one-time password method selection page and one-time password login page contain macros that you can use for various purposes.

About this task

In addition to these macros, you can define your own macros. Defining your own macros is a two-step process.

Procedure

1. Specify the name and the value of the macros in the one-time password mapping rules.
   a. To define a macro in OTP mapping rule, add an attribute into STS Universal User context attribute.
   b. Use `otp.sts.macro.type` as the type of the attribute.
   c. Use the name of the macro as the name of the attribute.
   d. Use the value of the macro as the value of the attribute.

   The value of the macro must be a string. The following is an example of JavaScript code.

```
var contextUserMacro = new Attribute("@OTP_MAPPING_RULE_DATA@",
"otp.sts.macro.type", "Data from OTP mapping rule");
stsuu.getContextAttributesAttributeContainer().setAttribute(contextUserMacro);
```

In this example, the name of the macro is `@OTP_MAPPING_RULE_DATA@`, and the value of the macro is `Data from OTP mapping rule`.

2. Add the name of the macros in the one-time password Method Selection page and one-time password Login pages.
   a. To display macros in the one-time password Method Selection page, define them in `OTPGetDeliveryMethods` mapping rule.
   b. To display macros in the one-time password Login page, define them in `OTPGenerate` mapping rule, `OTPDeliver` mapping rule, or `OTPVerify` mapping rule, depending on the operation that causes the one-time password Login page to be displayed.

      If the one-time password Login page is displayed after a one-time password is generated and delivered, the macros that are defined in `OTPGenerate` and `OTPDeliver` mapping rule are used. If the one-time password Login page is displayed after the one-time password provider plug-in determines that the submitted one-time password is incorrect, the macros that are defined in `OTPVerify` mapping rule are used.

manageItfimOneTimePassword

Use the **manageItfimOneTimePassword** command to list, view, configure, modify, and unconfigure one-time password federations.

Purpose

The **manageItfimOneTimePassword** command does the following operations:

- Listing all one-time password federations
- Viewing a one-time password federation
- Configuring a one-time password federation
- Modifying a one-time password federation
- Unconfiguring a one-time password federation
- Creating a one-time password response file

Syntax

```
$AdminTask manageItfimOneTimePassword {-operation operation
-fimDomainName domainName [optional_parameters]}
```

Parameters

The `operation` parameter and `fimDomainName` parameters are required. The following are optional parameters:

`-federationName` *federationName*

`-fileId` *fileId*

`-humanReadable` *humanReadable*

The use of these parameters depend on the operation.

The following parameters are available for use with the **manageItfimOneTimePassword** command:

-operation *operation*

Specifies the operation that you want to do. Table 1 lists the operations that are supported by this command.

Table 154. Values for the **-operation** *parameter*

Value	Description and requirements
list	Lists all one-time password federations.
view	Displays the details of a one-time password federation. When you use this operator, you must also use the following parameters: **federationName** *federationName* The name of the one-time password federation that you want to view. **humanReadable** *humanReadable* The parameter that produces a human-readable display of the one-time password federation.
configure	Configures a one-time password federation. When you use this operator, you must also use the following parameter: **fileId** *fileId* The file name of the response file that is based on which the new one-time password federation is configured. The name of the newly configured one-time password federation is specified in the response file by using the property **FedName**.
unconfigure	Unconfigures a one-time password federation. When you use this operator, you must also use the following parameter: **federationName** *federationName* The name of the one-time password federation that you want to unconfigure.

Table 154. Values for the **-operation** *parameter (continued)*

Value	Description and requirements
modify	Modifies a one-time password federation. When you use this operator, you must also use the following parameters: **federationName** *federationName* The name of the one-time password federation that you want to modify. **fileId** *fileId* The name of the one-time password response file that is based on which the one-time password federation is modified. To modify a one-time password federation, follow these steps: 1. Create a response file that is based on the one-time password federation that you want to modify. 2. Open the one-time password response file with a text editor. 3. Modify the parameters that you want to change. 4. Save and close the file. 5. Run the *modify* operation by specifying the file name of the response file in the **fileId** parameter. If you want to modify the name of the one-time password federation, use the parameter **FedName** in the one-time password response file.

Table 154. Values for the **-operation** *parameter (continued)*

Value	Description and requirements
createResponseFile	Creates a one-time password response file. You can create a sample response file or create a response file that is based on an existing one-time password federation. **Creating a sample response file** When you use this operator, you must also use the following parameter: **fileId** *fileId* The name of the one-time password response file. **Creating a response file that is based on an existing one-time password federation** When you use this operator, you must also use the following parameters: • **federationName** *federationName* The name of the one-time password federation that is based on which the one-time password response file is created. • **fileId** *fileId* The name of the one-time password response file.

-fimDomainName *fimDomainName*
Specifies the name of the domain where the operation is executed. The domain must exist.

-federationName *federationName*
Specifies the name of the one-time password federation. The federation must exist.

-fileId *fileId*
Specifies the name of the one-time password response file.

-humanReadable *humanReadable*
Specifies whether a human-readable display of the one-time password federation is shown.

Examples

The following examples show the correct syntax for several of the tasks for this command:

List all existing one-time password federations:

```
$AdminTask manageItfimOneTimePassword {-operation list
-fimDomainName otpdomain}
```

View a human-readable display of the details of a one-time password federation:

```
$AdminTask manageItfimOneTimePassword {-operation view
-fimDomainName otpdomain -federationName otpfed}
```

View a machine-friendly display of the details of a one-time password federation:

```
$AdminTask manageItfimOneTimePassword {-operation view
-fimDomainName otpdomain -federationName otpfed -humanReadable false}
```

Configure a one-time password federation

```
$AdminTask manageItfimOneTimePassword {-operation configure
-fimDomainName otpdomain
-fileId /home/user/otp_response.xml}
```

Modify a one-time password federation:

```
$AdminTask manageItfimOneTimePassword {-operation modify
-fimDomainName otpdomain
-federationName otpfed -fileId /home/user/otp_response.xml}
```

Unconfigure a one-time password federation:

```
$AdminTask manageItfimOneTimePassword {-operation unconfigure
-fimDomainName otpdomain -federationName otpfed}
```

Create a sample response file:

```
$AdminTask manageItfimOneTimePassword {-operation createResponseFile
-fimDomainName otpdomain
-fileId /home/user/otp_response.xml}
```

Create a response file that is based on an existing one-time password federation:

```
$AdminTask manageItfimOneTimePassword {-operation createResponseFile
-fimDomainName otpdomain
-federationName otpfed -fileId /home/user/otp_response.xml}
```

One-time password response file

Create a one-time password response file with the **manageItfimOneTimePassword** command to configure a new one-time password federation or modify an existing one-time password federation. Edit it with the appropriate values for your environment.

The one-time password response file is an XML file that is used by the **manageItfimOneTimePassword** command for configuring and modifying one-time password federations. You can use the same command to create a sample one-time password response file or a one-time password response file that is based on an existing one-time password federation. If you create a sample one-time password response file, the values of the parameters in the response file are populated with sample values. If you create a one-time password response file that is based on an existing one-time password federation, the values of the parameters in the response file are populated with the values used in that one-time password federation. See “manageItfimOneTimePassword” on page 665 for more details.

Sample one-time password response file

Create a sample one-time password response file with the following command:

```
$AdminTask manageItfimOneTimePassword {-operation createResponseFile
-fimDomainName otpDomain   -fileId /home/user/otp.response}
```

Existing one-time password federation

Create a response file that is based on an existing one-time password federation with the following command:

```
$AdminTask manageItfimOneTimePassword {-operation createResponseFile
-fimDomainName otpDomain  -federationName otpFederation
-fileId /home/user/otp.response}
```

Examples of the response file are in the following directories:

AIX, Linux or Solaris:

```
/opt/IBM/FIM/examples/responsefiles
```

Windows:

```
C:\Program Files\IBM\FIM\examples\responsefiles
```

Parameters

You must specify the parameters in the one-time response file before it can be used by the **manageItfimOneTimePassword** command. The following information lists all the parameters in the one-time password response file.

FedName

The name of the one-time password federation if the one-time password response file is used for configuring one-time password federation.

The *new* name of the one-time password federation if the one-time password response file is used for modifying an existing one-time password federation.

The name of the one-time password federation must contain alphanumeric characters only and must not be used by another one-time password federation.

Required: Yes

Example: `otpfed`

OTPGetDelivery MethodsMappingRule

The XML escaped content of the **OTPGetDeliveryMethods** mapping rule.

You must specify this parameter or **OTPGetDelivery MethodsMappingRuleFileName**. If both parameters are specified, this parameter is used.

See "OTPGetDeliveryMethods mapping rule" on page 653 for more details.

Required:
Yes, when the parameter **OTPGetDelivery MethodsMappingRuleFileName** is not specified.
No, when the parameter **OTPGetDelivery MethodsMappingRuleFileName** is specified.

Example:
See `$FIM_INSTALL_DIR$/examples/js_mappings/otp_get_delivery_methods.js`.

OTPGetDelivery MethodsMapping RuleFileName

The name of the file that contains the **OTPGetDelivery Methods** mapping rule.

You must specify this parameter or **OTPGetDelivery MethodsMappingRule**. If both parameters are specified, this parameter is not used.

See "OTPGetDeliveryMethods mapping rule" on page 653 for more details.

Required: Yes, when the parameter **OTPGetDelivery MethodsMappingRule** is not specified.
No, when the parameter **OTPGetDelivery MethodsMappingRule** is specified.

Example: `/home/user/otp_get_delivery_methods.js`

OTPGetDelivery MethodsMapping RuleType

The type of the **OTPGetDelivery Methods** mapping rule.

The type must be `JAVASCRIPT` or `XSLT`.

See "OTPGetDeliveryMethods mapping rule" on page 653 for more details.

Required: Yes

Example: `JAVASCRIPT`

OTPGenerate MappingRule

The XML escaped content of the **OTPGenerate** mapping rule.

You must either specify this parameter or **OTPGenerateMapping RuleFileName**. If both parameters are specified, this parameter is used.

See "OTPGenerate mapping rule" on page 653 for more details.

Required:
Yes, when the parameter **OTPGenerateMapping RuleFileName** is not specified.
No, when the parameter **OTPGenerateMapping RuleFileName** is specified.

Example:
See `$FIM_INSTALL_DIR$/examples/js_mappings/otp_generate.js`.

OTPGenerate MappingRuleFileName

The name of the file that contains the **OTPGenerate** mapping rule.

You must either specify this parameter or the parameter **OTPGenerate MappingRule**. If both parameters are specified, this parameter is not used.

See "OTPGenerate mapping rule" on page 653 for more details.

Required: Yes, when the parameter **OTPGenerate MappingRule** is not specified.
No, when the parameter **OTPGenerate MappingRule** is specified.

Example: `/home/user/otp_generate.js`

OTPGenerate MappingRuleType

The type of the **OTPGenerate** mapping rule.

The type must be `JAVASCRIPT` or `XSLT`.

See "OTPGenerate mapping rule" on page 653 for more details.

Required: Yes

Example: JAVASCRIPT

OTPDeliverMappingRule

The XML escaped content of the **OTPDeliver** mapping rule.

You must either specify this parameter or the parameter **OTPDeliver MappingRuleFileName**. If both parameters are specified, this parameter is used.

See "OTPDeliver mapping rule" on page 652 for more details.

Required: Yes, when the parameter **OTPDeliver MappingRuleFileName** is not specified.
No, when the parameter **OTPDeliver MappingRuleFileName** is specified.

Example:
See $FIM_INSTALL_DIR$/examples/js_mappings/otp_deliver.js.

OTPDeliver MappingRuleFileName

The name of the file that contains the **OTPDeliver** mapping rule.

You must either specify this parameter or the parameter **OTPDeliver MappingRule**. If both parameters are specified, this parameter is not used.

See "OTPDeliver mapping rule" on page 652 for more details.

Required: Yes, when the parameter **OTPDeliver MappingRule** is not specified.
No, when if the parameter **OTPDeliver MappingRule** is specified.

Example: /home/user/otp_deliver.js

OTPDeliver MappingRuleType

The type of the **OTPDeliver** mapping rule.

The type must be JAVASCRIPT or XSLT.

See "OTPDeliver mapping rule" on page 652 for more details.

Required: Yes

Example: JAVASCRIPT

OTPVerifyMappingRule

The XML escaped content of the **OTPVerify** mapping rule.

You must either specify this parameter or the parameter **OTPVerify MappingRuleFileName**. If both parameters are specified, this parameter is used.

See "OTPVerify mapping rule" on page 655 for more details.

Required: Yes, when the parameter **OTPVerify MappingRuleFileName** is not specified.
No, when the parameter **OTPVerify MappingRuleFileName** is specified.

Example:
See $FIM_INSTALL_DIR$/examples/js_mappings/otp_verify.js.

OTPVerifyMapping RuleFileName

The name of the file that contains the **OTPVerify** mapping rule.

You must either specify this parameter or the parameter **OTPVerify MappingRule**. If both parameters are specified, this parameter is not used.

See "OTPVerify mapping rule" on page 655 for more details.

Required: Yes, when the parameter **OTPVerify MappingRule** is not specified.
No, when the parameter **OTPVerify MappingRule** is specified.

Example: `/home/user/otp_verify.js`

OTPVerifyMapping RuleType

The type of the **OTPVerify** mapping rule.

The type must be `JAVASCRIPT` or `XSLT`.

See "OTPVerify mapping rule" on page 655 for more details.

Required: Yes

Example: `JAVASCRIPT`

OTPTypesTo OTPProviderModuleIds

The list of mappings from one-time password type to one-time password provider module IDs.

Each mapping specifies the one-time password provider plug-in that generates and verifies the one-time password for users with the specified one-time password type.

Each user can be associated with one-time password type. The one-time password provider module ID is an extension ID of the one-time password provider plug-in.

Required: No

Example: See OTPTypesToOTPProviderModuleIds.

DeliveryTypesTo OTPDeliveryModuleIds

The list of mappings from delivery type to one-time password delivery module IDs.

Each mapping specifies the one-time password delivery plug-in that delivers the one-time password for users with the specified delivery type.

Each user might be associated with delivery type. The one-time password delivery module ID is an extension ID of the one-time password delivery plug-in.

Required: No

Example: See DeliveryTypesToOTPDeliveryModuleIds.

OTPProvider ModuleConfigs

The list of mappings from one-time password type or one-time password provider module IDs to configurations.

The one-time password type or the one-time password provider module ID must correspond to a one-time password type or a one-time password provider module ID that is specified in the property **OTPTypesTo OTPProviderModuleIds**. Each configuration is a mapping from parameter name to parameter values.

Each mapping specifies the configuration of the specified one-time password provider module.

Required: No

Example: See OTPProviderModuleConfigs.

OTPDelivery ModuleConfigs

The list of mappings from delivery type or one-time password delivery module IDs to configurations.

The delivery type or the one-time password delivery module ID must correspond to a delivery type or a one-time password delivery module ID that is specified in the property **OTPTypesTo OTPDeliveryModuleIds**. Each configuration is a mapping from parameter name to parameter values.

Each mapping specifies the configuration of the specified one-time password delivery module.

Required: No

Example: See OTPDeliveryModuleConfigs.

Sensitive Parameters

Some of the configurations of the one-time password provider module and the one-time password delivery module might be sensitive.

An example is the `SMTPPassword` configuration of the `EmailOTPDelivery` one-time password delivery module. You can declare these configurations as sensitive by annotating the parameter name with `@Sensitive`.

Tivoli Federated Identity Manager obfuscates all configurations that are declared as sensitive before it stores them in the Tivoli Federated Identity Manager configuration files.

When you view a one-time password federation, the sensitive configurations are displayed in obfuscated form. When you create a one-time response file that is based on an existing one-time password federation, the sensitive configurations are populated in obfuscated form.

For an example of a sensitive configuration, see OTPDeliveryModuleConfigs.

Examples

•

OTPTypesToOTPProviderModuleIds parameter

The following example shows the value of the parameter **OTPTypesToOTPProviderModuleIds**:

```
<object class="java.util.HashMap">
      <void method="put">
        <string>mac_otp</string>
        <string>MobileAuthCodeOTPModule</string>
      </void>
 </object>
```

This example contains only one mapping. It shows a mapping from one-time password type `mac_otp` to one-time password provider module ID `MobileAuthCodeOTPModule`.

- **DeliveryTypesToOTPDeliveryModuleIds parameter**

  The following example shows the value of the parameter **DeliveryTypesToOTPDeliveryModuleIds**:

```
<object class="java.util.HashMap">
      <void method="put">
         <string>sms_delivery</string>
         <string>SMSOTPDelivery</string>
      </void>
      <void method="put">
        <string>mail_delivery</string>
        <string>EmailOTPDelivery</string>
      </void>
 </object>
```

  This example contains two mappings.

  – The first mapping is a mapping from delivery type `sms_delivery` to one-time password delivery module ID `SMSOTPDelivery`.
  – The second mapping is a mapping from delivery type `mail_delivery` to one-time password delivery module ID `EmailOTPDelivery`.

- **OTPProviderModuleConfigs parameter**

  The following example shows the value of the parameter **OTPProviderModuleConfigs**:

```
<object class="java.util.HashMap">
      <void method="put">
        <string>mac_otp</string>
        <object class="java.util.HashMap">
          <void method="put">
            <string>StoreEntryFactoryHashAlgorithm</string>
            <object class="java.util.ArrayList">
              <void method="add">
                <string>SHA-256</string>
              </void>
            </object>
          </void>
          <void method="put">
            <string>GeneratorCharacterSet</string>
            <object class="java.util.ArrayList">
              <void method="add">
                <string>0123456789</string>
              </void>
            </object>
```

```
        </void>
      </object>
    </void>
  </object>
```

This example contains one mapping. This is a mapping from one-time password type `mac_otp` to its configuration. The configuration contains two parameters. The first parameter name is **StoreEntryFactoryHashAlgorithm**, which has only one parameter value *SHA-256*. The second parameter is **GeneratorCharacterSet**, which has only one parameter value *0123456789*.

- **OTPDeliveryModuleConfigs parameter**

  The following example shows the value of the parameter **OTPDeliveryModuleConfigs**:

```
<object class="java.util.HashMap">
      <void method="put">
        <string>mail_delivery</string>
        <object class="java.util.HashMap">
          <void method="put">
            <string>@Sensitive SMTPPassword</string>
            <object class="java.util.ArrayList">
              <void method="add">
                <string>password</string>
              </void>
            </object>
          </void>
        </object>
      </void>
      <void method="put">
        <string>sms_delivery</string>
        <object class="java.util.HashMap">
          <void method="put">
            <string>HTTPParameters</string>
            <object class="java.util.ArrayList">
              <void method="add">
                <string>From=+15127828860</string>
              </void>
              <void method="add">
                <string>To=$DEST_NO$</string>
              </void>
              <void method="add">
                <string>Message=$MSG$</string>
              </void>
            </object>
          </void>
        </object>
      </void>
    </object>
```

  This example contains 2 mappings.

  – The first mapping is a mapping from delivery type **mail_delivery** to its configuration. The configuration contains only one parameter. This parameter name is **SMTPPassword**, which has only one parameter value *password*. This parameter is a sensitive parameter, since it is annotated with `@Sensitive`.

  – The second mapping is a mapping from delivery type **sms_delivery** to its configuration. The configuration contains only one parameter. This parameter name is **HTTPParameters**, which has 3 parameter values *From=+15127828860*, *To=$DEST_NO$*, and *Message=MSG*.

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 (by 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. Optional parameters:

```
-uuid ID
-signInCallbackIds callback1,callback2
-signOutCallbackIds callback1,callback2
-locaIdCallbackIds callback1,callback2
-authenticationCallbackIds callback1,callback2
-authenticationPolicyCallbackIds 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 that is used with this parameter specifies the operation to perform on the domain. Valid values are listed in the following table.

Table 155. 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 155. 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.
delete	Delete a specific custom point of contact profile. **Note:** You cannot delete a default point of contact profile. You can delete only the point of contacts that you created. The default points 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 155. Values for the manageItfimPointOfContact **-operation** *parameter (continued)*

Value	Description and requirements
createResponseFile	Create a response file that you want to use to create a point of contact profile. You can create a response file for a new point of contact profile. You can also 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* **authenticationPolicyCallbackIds***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 create a 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 681.
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 that is 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 that are used by the point of contact for sign-in actions.

-signOutCallbackIds *callback*
A comma-separated list of callbacks that are used by the point of contact for sign-out actions.

-localIdCallbackIds *callback*
A comma-separated list of callbacks that are used by the point of contact for the local ID.

-authenticationCallbackIds *callback*
A comma-separated list of callbacks that are used by the point of contact for authentication.

-authenticationPolicyCallbackIds *callback*
A comma-separated list of callbacks that are used by the point of contact for determining authentication policy.

-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 that is 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 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
  -authenticationPolicyCallbackIds genericPocAuthnPolicyCallback
  -fileId c:\home\files\temp\empty.xml}
```

Note: The file that is 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 create 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 that is 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 that is 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 create 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:

Note: The file that is specified here is the response file and is used as input. Open the response file with a text editor before running this command. Ensure that the attributes that are defined in the file are correct for your environment.

```
$AdminTask manageItfimPointOfContact {-operation create
  -fimDomainName domain1
  -fileId c:\home\files\temp\empty.xml}
```

Point of contact response file

You must create a response file before you can create a point of contact profile with the **manageItfimPointOfContact** command. Then, edit the response file so that it contains the appropriate values for your environment.

You can create a response file when you create a 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
  -authenticationPolicyCallbackIds callback
  -fileId filename}
```

A response file is created after either of these commands are run. The content of the file differs depending on the properties that are 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.

Examples of the response file are 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 that are 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 683.

Table 156. 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 156. 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 authenticate 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.
authnpolicy.INDEX=	*callbackID,callbackID,*	One or more callback IDs that are used for the determination of the authentication policy. The INDEX represents the order in which this callback is invoked in the authentication policy determination 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.

Note: If the callback property name ends with `MappingRuleFileName`, the contents of the file is uploaded as a mapping rule. The property name of the `config` item is the text after the `MappingRuleFileName` suffix. For example, to create a property named `authentication.policy.map.rule`, the property on the response file must be named `CALLBACKID.authentication.policy.map.ruleMappingRuleFileName`.

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, know that a point of contact profile has a hierarchy in which there are the following callback types:

```
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 adds the properties to the callback and assign them to the correct type in the provided 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>authnpolicy.1</string>
   <object class="java.util.ArrayList">
    <void method="add">
     <string>genericPocAuthnPolicyCallback</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>
```

One-time password provider plug-in reference

The one-time password provider plug-in generates and validates one-time passwords. Configure the one-time password provider plug-in so that it can be used by Tivoli Federated Identity Manager.

Tivoli Federated Identity Manager provides three one-time password provider plug-ins:

- "MobileAuthCodeOTPModule"
- "TOTPModule" on page 687
- "HOTPModule " on page 689

The `MobileAuthCodeOTPModule` generates one-time password by randomly drawing one character at a time from the configured character set until the configured number of characters are drawn. The `MobileAuthCodeOTPModule` also stores the generated one-time password in the configured one-time password store plug-in. The one-time password is salted and hashed before it is stored in the configured one-time password store plug-in.

The `TOTPModule` generates one-time password by using a specified algorithm with a time-based one-time password application. Passwords are not communicated or stored, but are verified as a match between server and client as they are regenerated at regular intervals.

The `HOTPModule` generates one-time password by using a specified algorithm with a counter-based one-time password application. Passwords are not communicated or stored, but are verified as incremental matches between server and client.

MobileAuthCodeOTPModule

The following lists all the configurations of `MobileAuthCodeOTPModule`. All the configurations are optional.

`StoreModuleId`

The extension ID of the one-time password store plug-in that stores the one-time password.

The default `StoreModuleId`, which is `OTPProviderDynaCacheOTPStore`, is used when this configuration is not specified, or when it refers to a non-existing one-time password store plug-in.

Required: No

Multi-value: No

Example: `OTPProviderDynaCacheOTPStore`

StoreEntryLifetime
The lifetime of the one-time password that is stored in the one-time password store plug-in. The lifetime is in seconds.

The default `StoreEntryLifetime`, which is `300`, is used when this configuration is not specified, or when it is less than zero.

Required: No

Multi-value: No

Example: `300`

GeneratorCharacterSet
The character set from which the characters in the one-time password are generated.

The default `GeneratorCharacterSet`, which is `0123456789`, is used when this configuration is not specified, or when it is empty.

Required: No

Multi-value: No

Example: `0123456789`

GeneratorLength
The length of the characters in the one-time password.

The default `GeneratorLength`, which is `8`, is used when this configuration is not specified, or when it is less than one.

Required: No

Multi-value: No

Example: `8`

StoreEntryFactoryHashAlgorithm
The hash algorithm that is used for hashing the one-time password before it is stored in the one-time password store plug-in.

The default `StoreEntryFactoryHashAlgorithm`, which is `SHA-256`, is used when this configuration is not specified, or when it is not supported by the underlying Java Cryptography Architecture API.

For the list of supported hash algorithms, refer to *Appendix A* of the Java Cryptography Architecture(JCA) API Specification & Reference.

Required: No

Multi-value: No

Example: `SHA-256`

StoreEntryFactorySaltLength
The length of the randomly generated salt that is used for hashing the one-time password before it is stored in the one-time password store plug-in.

The default `StoreEntryFactorySaltLength`, which is `5`, is used when this configuration is not specified, or when it is less than one.

Required: No

Multi-value: No

Example: 5

TOTPModule

The following list describes all the configurations of TOTPModule.

OTPLength
The length of the generated one-time passwords, which can be between 6 to 9 characters or numbers.

The default OTPLength, which is 6, is used when this configuration is not specified.

Required: No

Multi-value: No

Example: 6

OTPTimeSkewIntervals
The skew intervals of the algorithm. The skew intervals consider any possible synchronization delay between the server and the client that generates the one-time password. For example, a skew interval of 2 means a one-time password in up to two intervals in the past, or two in the future are valid. For example, if it is interval 563, and intervals are 30 seconds, then one-time passwords for intervals 561-565 are computed and checked against within a range of 2.5 minutes.

The default OTPTimeSkewIntervals, which is 1, is used when this configuration is not specified.

Required: No

Multi-value: No

Example: 1

OTPGenerationAlgorithm
The algorithm that is used to generate the one-time password. Valid options include the following algorithms: HmacSHA1, HmacSHA256, or HmacSHA512

Note: Not all algorithms are supported by all Java levels. For example, Java 1.4 supports only HmacSHA1. For a list of supported hash algorithms for your Java version, refer to *Appendix A* of the Java Cryptography Architecture (JCA) API Specification & Reference for Java 1.4, or Java Cryptography Architecture (JCA) API Specification & Reference for Java 1.5.

The default OTPGenerationAlgorithm, which is HmacSHA1, is used when this configuration is not specified.

Required: No

Multi-value: No

Example: HmacSHA1

OTPGenerationIntervalSeconds
The number of seconds an interval lasts. This number determines how long a one-time password is active before the next one-time password generates.

The default `OTPGenerationIntervalSeconds`, which is `30`, is used when this configuration is not specified.

Required: No

Multi-value: No

Example: `30`

`OneTimeUseEnforcementEnabled`

Whether to cache one-time passwords if they are used to successfully log in. If set to `true`, then the reuse of a one-time password is prevented while it is in cache.

The default `OneTimeUseEnforcementEnabled`, which is `true`, is used when this configuration is not specified.

Required: No

Multi-value: No

Example: `true`

`OneTimeUseEnforcementStore`

The object cache to use for caching successful one-time passwords. This storage goes unused if `OneTimeEnforcementEnabled` is set to `false`.

The default `OneTimeUseEnforcementStore`, which is `OTPProviderDynaCacheOTPStore`, is used when this configuration is not specified.

Required: No

Multi-value: No

Example: `OTPProviderDynaCacheOTPStore`

`OTPSecretKeyAttributeName`

The attribute name that retrieves the user secret key for one-time password value generation. Any unique string value that identifies the attribute is valid.

The default `otp.hmac.secret.key` is used when this configuration is not specified.

Required: No

Multi-value: No

Example: `otp.hmac.secret.key`

`OTPSecretKeyAttributeNamespace`

The attribute namespace that retrieves the user secret key for one-time password value generation. Any string value that identifies the source of the attribute is valid. Null values are not valid.

The default `urn:ibm:security:otp:hmac` is used when this configuration is not specified.

Required: No

Multi-value: No

Example: `urn:ibm:security:otp:hmac`

HOTPModule

The following list describes all the configurations of `HOTPModule`.

`OTPLength`

The length of the generated one-time passwords, which can be between 6 to 9 characters or numbers.

The default `OTPLength`, which is `6`, is used when this configuration is not specified.

Required: No

Multi-value: No

Example: `6`

`OTPGenerationAlgorithm`

The algorithm that is used to generate the one-time password. Valid options include the following algorithms: `HmacSHA1`, `HmacSHA256`, or `HmacSHA512`

Note: Not all algorithms are supported by all Java levels. For example, Java 1.4 supports only `HmacSHA1`. For a list of supported hash algorithms for your Java version, refer to *Appendix A* of the Java Cryptography Architecture (JCA) API Specification & Reference for Java 1.4, or Java Cryptography Architecture (JCA) API Specification & Reference for Java 1.5.

The default `OTPGenerationAlgorithm`, which is `HmacSHA1`, is used when this configuration is not specified.

Required: No

Multi-value: No

Example: `HmacSHA1`

`MaxCounterLookahead`

The number of times to increment the counter to see whether the one-time password is valid before stopping. Any non-negative number is valid.

The default `MaxCounterLookahead`, which is `25`, is used when this configuration is not specified.

Required: No

Multi-value: No

Example: `25`

`OTPSecretKeyAttributeName`

The attribute name that retrieves the user secret key for one-time password value generation. Any unique string value that identifies the attribute is valid.

The default `otp.hmac.secret.key` is used when this configuration is not specified.

Required: No

Multi-value: No

Example: `otp.hmac.secret.key`

OTPSecretKeyAttributeNamespace

The attribute namespace that retrieves the user secret key for one-time password value generation. Any string value that identifies the source of the attribute is valid. Null values are not valid.

The default `urn:ibm:security:otp:hmac` is used when this configuration is not specified.

Required: No

Multi-value: No

Example: `urn:ibm:security:otp:hmac`

OTPCounterAttributeName

The attribute name that stores and retrieves the counter value for one-time password value generation. Any unique string value that identifies the attribute is valid.

The default `otp.hmac.counter` is used when this configuration is not specified.

Required: No

Multi-value: No

Example: `otp.hmac.counter`

OTPCounterAttributeNamespace

The attribute namespace that stores and retrieves the counter value for one-time password value generation. Any string value that identifies the source of the attribute is valid. Null values are not valid.

The default `urn:ibm:security:otp:hmac` is used when this configuration is not specified.

Required: No

Multi-value: No

Example: `urn:ibm:security:otp:hmac`

Related reference:

"One-time password response file" on page 669
Create a one-time password response file with the **manageItfimOneTimePassword** command to configure a new one-time password federation or modify an existing one-time password federation. Edit it with the appropriate values for your environment.

"One-time password delivery plug-in reference"
The one-time password delivery plug-in is a plug-in that delivers one-time passwords to users. Configure the one-time password delivery plug-in so that it can be used by Tivoli Federated Identity Manager.

One-time password delivery plug-in reference

The one-time password delivery plug-in is a plug-in that delivers one-time passwords to users. Configure the one-time password delivery plug-in so that it can be used by Tivoli Federated Identity Manager.

Tivoli Federated Identity Manager provides the following one-time password delivery plug-ins:

- `SMSOTPDelivery`
- `EmailOTPDelivery`

- NoOTPDelivery

The SMSOTPDelivery plug-in delivers the one-time password by using the Short Message Service or SMS. The SMSOTPDelivery first sends the phone number of the user and the one-time password in an **HTTP POST** request, whose content type is application/x-www-form-urlencoded, to the configured SMS Gateway. The SMS Gateway then sends the one-time password to the user through SMS. Tivoli Federated Identity Manager is not shipped with any SMS Gateway. You must configure your own SMS Gateway.

The EmailOTPDelivery plug-in delivers the one-time password by using email. The EmailOTPDelivery sends the email address of the user and the one-time password in a message, whose MIME type is text/plain, to the configured SMTP Server. The SMTP Server then sends the one-time password to the user by email. Tivoli Federated Identity Manager is not shipped with any SMTP Server. You must configure your own SMTP Server.

The NoOTPDelivery plug-in specifies that there is no delivery of a password. Use the NoOTPDelivery plug-in with time-based one-time password applications where password communication and storage is not necessary.

You must configure the SMSOTPDelivery and EmailOTPDelivery plug-ins before they can be used by Tivoli Federated Identity Manager. Configuration of SMSOTPDelivery and EmailOTPDelivery includes the following parameters:

SMSOTPDelivery parameters

ConnectionURL
The URL of the SMS Gateway where the phone number of the user and the one-time password is sent.

Required: True

Multi-value: No

Example: https://smsgateway.tfim.example.com/

HTTPParameters
The list of name and value pairs that is included in the body of the **HTTP POST** request to the SMS Gateway. In each pair, the name and the value must be separated by equal sign.

Tivoli Federated Identity Manager provides two macros, **$DEST_NO$** and **MSG,** that are replaced by the phone number of the user and the content of the SMS. These two macros can be used only as value in the name and value pair.

Required: True

Multi-value: Yes

Example:

- **From=+0123456789**
- **To=** *$DEST_NO$*
- **Body=** *MSG*

HTTPSTruststore
The Tivoli Federated Identity Manager keystore that validates the SMS Gateway SSL certificate.

This configuration must be specified only when SMSOTPDelivery communicates with the SMS Gateway by using HTTPS.

Required: False

Multi-value: No

Example: DefaultTrustedStore

BasicAuthUserName

The user name that is used in HTTP Basic authentication.

SMSOTPDelivery does not perform the HTTP basic authentication if this configuration is not specified.

Required: False

Multi-value: No

Example: username

BasicAuthPassword

The password that is used in HTTP basic authentication.

SMSOTPDelivery does not perform HTTP Basic authentication if this configuration is not specified.

Required: False

Multi-value: No

Example: password

ClientAuthKey

The Tivoli Federated Identity Manager certificate that is used as client certificate in SSL Client authentication. The certificate is a keystore and alias pair. The keystore and alias must be separated by underscore.

SMSOTPDelivery does not perform SSL Client authentication if this configuration is not specified.

Required: False

Multi-value: No

Example: DefaultKeystore_testkey

SuccessHTTPReturnCode

The response code from the SMS Gateway that is an acknowledge ment from the SMS Gateway that the request is successfully processed.

The default SuccessHTTP ReturnCode, which is 200, is used when this configuration is not specified.

Note: The SuccessHTTP ReturnCode match must be successful before the SuccessHTTP ResponseBody RegexPattern matching is done.

Required: False

Multi-value: No

Example: 200

SuccessHTTPResponseBodyRegexPattern

This parameter defines the Java regular-expression pattern that matches the

HTTP response body that is returned by the SMS Gateway. When the match is successful, Tivoli Federated Identity Manager determines that the SMS delivery is successful.

The default value is empty.

The default behavior is that the HTTP response body is not going to be matched against any Java regular-expression and the success or failure decision is going to be based on the `SuccessHTTP ReturnCode` value only.

Note: If the HTTP response from the SMS Gateway does not contain a body, the `SuccessHTTP ResponseBody RegexPattern` matching is not performed.

Required: False

Multi-value: No

Example:

- When the body of all responses by the SMS Gateway contains either `Success` or `Failure` followed by no newline character, the sample `SuccessHTTP Response BodyRegex Pattern` value is

  ```
  Success
  ```

- When the body of all responses by the SMS Gateway contains the following text:

  ```
  MGDID=TTTT
  TTTTTTTTT
  RESPONSE
  CODE=NNN
  SMS=TTTTTTT
  TTTTTTTT
  TTTTTTT
  DATE=NNNNNNNN
  ```

  where each line ends with the `\n` character without any preceding `\r` character, and the `RESPONSECODE` is defined such that a three-digit number from 0 to 199 indicates success, the sample `SuccessHTTP ResponseBody RegexPattern` value is

  ```
  (?s).*
  RESPONSE
  CODE=(\d{1,2}
  |[0-1]{1}
  \d{2})\n.*
  ```

EmailOTPDelivery plug-in configuration

SMTPHostname

The host name of the SMTP Server.

Required: True

Multi-value: No

Example: `smtpserver.tfim.example.com`

SMTPUsername
The user name that is used in SMTP authentication.

Required: False

Multi-value: No

Example: `username`

SMTPPassword
The password that is used in SMTP authentication.

Required: False

Multi-value: No

Example: `password`

SenderEmail
The email address that is used as the sender of the email that is sent to the user.

Required: True

Multi-value: No

Example: `otp_emailer@example.com`

One-time password user information provider plug-in reference

The one-time password user information provider plug-in retrieves values from a database for those one-time password algorithms that require user information.

You can configure Tivoli Federated Identity Manager to retrieve these values from a relational or file-based database. See the following topics for database setup information:

- "Setting up DB2 for one-time password user information storage" on page 696
- "Setting up solidDB for one-time password user information storage" on page 697

Tivoli Federated Identity Manager is shipped with the following plug-ins:

JDBCUserInfoModule
Provides the user information from a JDBC-based database.

FileUserInfoModule
Provides the user information from a file-based database. This plug-in is supported only on stand-alone, or nonclustered, environments.

JDBC user information provider

The following configuration settings are for `JDBCUserInfoModule`. All settings are optional.

DBDataSource
The JNDI name for the data source that corresponds to the user information database. Specify a defined data source in the WebSphere Application Server environment.

The default value is `jdbc/fim`.

Required: No

Multi-value: No

Example: `jdbc/fim`

DBLoggingEnabled
A Boolean value to enable finer-grained tracing on the database connection. Specify `true` or `false`.

The default value is `false`.

Required: No

Multi-value: No

Example: `false`

File user information provider

The following configuration settings are for `FileUserInfoModule`. All settings are optional.

FileDBFileName
The file name of the user information file database. Specify any unique and valid file name. The user under which the application server is running must have write access to this file. If the file does not exist, the product creates it.

The default value is `fileUserInfo.properties`.

Required: No

Multi-value: No

Example: `fileUserInfo.properties`

FileDBRootDirectory
The root directory where the user information file is stored. Specify any valid directory that the user, under which the application server is running, has write access.

The default value is the Tivoli Federated Identity Manager configuration repository `/etc` directory:*WAS_ROOT*`/profiles/`*WAS_PROFILE*`/config/itfim/`*FIM_DOMAIN*`/etc`

Required: No

Multi-value: No

Example: `/opt/IBM/WebSphere/AppServer/profiles/ip/config/itfim/fimipdomain/etc`

FileDBKeyTokensDelimeter
The character to separate the different values that comprise the database entry key. Specify any character that is not in the attribute name, attribute namespace, or attribute data type values.

The default value is `%`.

Required: No

Multi-value: No

Example: `%`

Related reference:

"One-time password response file" on page 669
Create a one-time password response file with the **manageItfimOneTimePassword** command to configure a new one-time password federation or modify an existing one-time password federation. Edit it with the appropriate values for your environment.

"One-time password delivery plug-in reference" on page 690
The one-time password delivery plug-in is a plug-in that delivers one-time passwords to users. Configure the one-time password delivery plug-in so that it can be used by Tivoli Federated Identity Manager.

Setting up DB2 for one-time password user information storage

You can set up DB2® as your user information database for one-time password calculation.

Before you begin

- Review the DB2 requirements for user information provider support. See Additional software.
- Install the DB2 database.

Procedure

1. In WebSphere Application Server, create and configure a JNDI context that is named `jdbc/fim`. See the WebSphere Application Server Information Center. Search for *configuring a data source*.
2. Set custom schema properties by completing the following steps:
   a. In the administrative console, click **Resources** > **JDBC** > **Data sources**.
   b. Click the name of the data source that was created in step 1 to open the **Configuration** page.
   c. Click **Custom properties**.
   d. Click **currentSchema**.
   e. In the **Value** field, type `FIM_DB`.
   f. Click **OK**.
   g. Click **Save directly to the master configuration**.
3. Run the `.sql` file to create the database schema for one-time password user information.

   Linux or UNIX operating systems

   The `.sql` file to create the database for DB2 is in the *FIM_HOME*`/dbscripts/db2/` directory. For example, the directory is `/opt/IBM/FIM/dbscripts/db2/`.

   a. Edit the `create_schema.sql` file, and replace **&DBUSER** and **$DBPASSWD** with the database user name and password.
   b. Run the `create_schema.sql` file by using the **db2** command. For example:

      ```
      db2 -tvf /opt/IBM/FIM/dbscripts/db2/create_schema.sql
      ```

   Windows operating systems

   The `.sql` file to create the database for DB2 is in the *FIM_HOME*`\dbscripts\ db2\` directory. For example, the directory is `C:\Program Files\IBM\FIM\dbscripts\db2\`.

   a. Edit the `create_schema.sql` file, and replace **&DBUSER** and **$DBPASSWD** with the database user name and password.

b. Run the `create_schema.sql` file by using the **db2** command. For example:

```
db2 -tvf C:\Progra~1\IBM\FIM\dbscripts\db2\create_schema.sql
```

4. Using the data source that was created in step 1 on page 696 and the administrative console, test the connection to the database.

Setting up solidDB for one-time password user information storage

You can set up solidDB® as your user information database for one-time password calculation.

Before you begin

- Review the solidDB requirements for user information provider support. See Additional software.
- Install the solidDB database.

Procedure

1. In WebSphere Application Server, create and configure a JNDI context that is named `jdbc/fim`.See the WebSphere Application Server Information Center. Search for *configuring a data source*.
2. Set custom schema properties by completing the following steps:
   a. In the administrative console, click **Resources** > **JDBC** > **Data sources**.
   b. Click the name of the data source that was created in step 1 to open the **Configuration** page.
   c. Click **Custom properties**.
   d. Click **currentSchema**.
   e. In the **Value** field, type `FIM_DB`.
   f. Click **OK**.
   g. Click **Save directly to the master configuration**.
3. Create a solidDB database with the solidDB tools. Set the database catalog and user name to `FIM_DB`. Enter the following command one line:

   ```
   SOLID_BIN_DIRECTORY/solid -UFIM_DB -PDBPASSWORD -CFIM_DB -xexit
     -xdisableallmessageboxes -xhide -CWORKING_DIRECTORY
   ```

   where:

   SOLID_BIN_DIRECTORY
   Specifies the solidDB database installation `bin` directory.

   DBPASSWORD
   Specifies the database password.

   WORKING_DIRECTORY
   Specifies the working directory where the solidDB `.ini` file and license file are located.

   The following is an example command for Linux that specifies the solidDB evaluation license:

   ```
   /opt/solidDB/soliddb-7.0/bin/solid -UFIM_DB -Ppassword -CFIM_DB -xexit
     -xdisableallmessageboxes -xhide -C/opt/solidDB/soliddb-7.0/eval_kit/standalone
   ```

   See the solidDB documentation for more details.
4. Start the solidDB database with the following command:

   ```
   SOLID_BIN_DIRECTORY/solid -UFIM_DB -PDBPASSWORD -CFIM_DB
     -xdisableallmessageboxes -xhide -CWORKING_DIRECTORY
   ```

The following is an example command for Linux that specifies the solidDB evaluation license:

```
/opt/solidDB/soliddb-7.0/bin/solid -UFIM_DB -Ppassword -CFIM_DB
  -xdisableallmessageboxes -xhide -C/opt/solidDB/soliddb-7.0/eval_kit/standalone
```

See the solidDB documentation for more details.

5. Run the `.sql` file to create the database schema for one-time password user information.

   Linux or UNIX operating systems

   The `.sql` file to create the database schema for solidDB is in the *FIM_HOME*`/dbscripts/soliddb/` directory. For example, on Linux, it is `/opt/IBM/FIM/dbscripts/soliddb/`.

   Run the `create_schema.sql` file by using the **solsql** command.

   ```
   solsql "NETWORK_NAME" FIM_DB DBPASSWORD
     /opt/IBM/FIM/dbscripts/soliddb/create_schema.sql
   ```

   where:

   NETWORK_NAME
   Specifies the network name of a solidDB server to which you are connecting.

   DBPASSWORD
   Specifies the database password.

   For example:

   ```
   solsql "tcpip 1964" FIM_DB password
     /opt/IBM/FIM/dbscripts/soliddb/create_schema.sql
   ```

   Windows operating systems

   The `.sql` file to create the database schema is in the `C:\Program Files\IBM\FIM\dbscripts\soliddb\` directory.

   Run the `create_schema.sql` file by using the **solsql** command.

   ```
   solsql "NETWORK_NAME" FIM_DB DBPASSWORD
     C:\Progra~1\IBM\FIM\dbscripts\soliddb\create_schema.sql
   ```

   where:

   NETWORK_NAME
   Specifies the network name of a solidDB server to which you are connecting.

   DBPASSWORD
   Specifies the database password.

   For example:

   ```
   solsql.exe "tcpip 1964" FIM_DB password
     C:\Progra~1\IBM\FIM\dbscripts\soliddb\create_schema.sql
   ```

6. Using the data source that was created in step 1 on page 697 and the administrative console, test the connection to the database.

Chapter 47. Tuning the one-time password

Improve the performance of the one-time password system by tuning the `OTPProviderDynaCacheOTPStore` component.

The `OTPProviderDynaCacheOTPStore` is the one-time password store plug-in. It uses WebSphere Application Server object cache as its underlying storage.

The following one-time password modules use this store plug-in:

- `MobileAuthCodeOTPModule`: Stores one-time passwords in the store.
- `TOTPModule`: Uses the store for one time use enforcement.

You can tune `OTPProviderDynaCacheOTPStore` by using two approaches:

Tune the WebSphere Application Server object cache

The WebSphere Application Server object cache that is used by `OTPProviderDynaCacheOTPStore` is `itfim-otp`. You can tune this object cache by changing the size of the cache or enabling disk offload.

See the WebSphere Application Server documentation for more details.

Tune how `OTPProviderDynaCacheOTPStore` uses the WebSphere Application Server object cache

The `OTPProviderDynaCacheOTPStore` retrieves the one-time password from the WebSphere Application Server object cache by polling. If the one-time password is not available, the `OTPProviderDynaCacheOTPStore` waits for a certain amount time before it tries to retrieve the one-time password again. This cycle continues until the one-time password is available. If the one-time password is still not available after a certain amount of time, `OTPProviderDynaCacheOTPStore` times out.

You can configure the amount of time that the `OTPProviderDynaCacheOTPStore` waits before trying to retrieve the one-time password again by setting the runtime custom property `DistributedMap.GetRetryDelay`.

You can configure the number of additional tries before the `OTPProviderDynaCacheOTPStore` timeouts by setting the runtime custom property `DistributedMap.GetRetryLimit`.

See General properties for more details.

Part 8. Customization

The topics in the Customization section explain how to customize components and functions of Tivoli Federated Identity Manager to better suit your environment.

Chapter 48. Customizing runtime properties

Custom properties can be used to tailor the runtime service of the Tivoli Federated Identity Manager to meet specific needs.

The use of custom properties is an advanced task. Familiarize yourself with Tivoli Federated Identity Manager architecture and services to understand how to use the custom properties. See the Tivoli Federated Identity Manager information center for more information.

Creating a custom property

You can customize the domain configuration by defining a custom property.

About this task

The syntax for custom properties is:

```
property_name = property_value
```

Procedure

1. Log 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 **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 **OK** to apply the changes that you have made and exit from the panel.

Deleting a custom property

You can remove a custom property based on its scope.

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.

Custom properties reference

You can set values for a number of custom properties. This reference section describes each of the custom properties.

To add the custom properties to your domain configuration, see "Creating a custom property" on page 703.

General properties

DistributedMap.GetRetryLimit
When specified, and when the value is greater than 0, the wrapper will query the distributed map the configured number of times before returning that the data is not in the map.
- Value type: Integer
- Example value: 2

DistributedMap.GetRetryDelay
When the retry limit is higher than 1, this value sets the time to wait in milliseconds between retries. The default is 2000, or 2 seconds.
- Value type: Integer
- Example value: 2000

componentName.**statisticsEnabled**
When specified as True, statistics tracking function for a specific component is enabled and the data collected and can be retrieved using the mechanisms presented by the component. When set to false, statistics are not tracked. Typically, this property is set to true for components that need to be timed or counted.
- Value type: Boolean
- Example value: False

Custom properties for single sign-on protocol service

Use the single sign-on custom properties to suit your deployment requirements.

requireSoapActionForSoap
This parameter controls the single sign-on protocol service behavior when it receives a request through the browser POST method and it needs to determine if it is a SOAPRequest or a BrowserRequest. Use of this parameter enables the service to handle non-compliant SOAP clients that do not send the required SOAPAction header on SOAP requests.

Default value: true

- Value type: boolean
- Example value: true

requireContentTypeForSoap
This parameter controls whether or not a SOAPRequest must contain a content-type of either `text/xml` or `application/soap+xml`. This parameter enables the single sign-on protocol service to handle non-compliant SOAP clients.

Note: When this parameter, and requestSoapActionForSoap are both false, all posts will be interpreted as SOAPRequests.

Default value: true

- Value type: boolean
- Example value: True

POC.allowsCredRefresh
When set to true, this parameter causes the LocalLogoutAction to be skipped on the service provider during single sign-on and federation. Instead, the credentials are refreshed. Set this parameter to true for the Web Plug-ins. Otherwise, set it to false.

Default value: true

- Value type: boolean
- Example value: True

SPS.PageFactory.HtmlEscapedTokens
A comma-separated list of tokens that must be HTML-escaped when being rendered in pages sent to the browser. Typically, this property includes any macros in the SPS.PageFactory.Exception2Macro runtime custom property (if used). This property is an important security consideration for preventing cross-site scripting vulnerabilities.

- Value type: string
- Example value: @TOKEN_A@, @TARGET@

SPS.PageFactory.Exception2Macro
This runtime custom property is a comma-separated list of classname:macro pairs. Classname is the full name of an exception class. Macro is the replacement macro to which the class maps. The macro must start and end with "@" as shown in the example values.

- Value type: string
- Example values: com.demo.MyException: @MYEXCEPTION@, com.tivoli.am.fim.trustserver.sts.STSException: @STSEXCEPTION@

SPS.POC.Default.Header.Names.Enabled
When specified, this property enables the use of default header names for the

point of contact header values. If false, the only headers that will be read or written will have to be part of the sps.xml configuration file.

- Value type: boolean
- Example value: false

POC.WebSeal.SignOutInfoDelegate.UserSessionIdHeaderName
This value overrides the default tagvalue_user_session_id.

- Value type: String
- Example value: tagvalue_user_session_id

SOAP.AuthType
The authentication type to be used when accessing the SOAP endpoint. The value can either be `ba` indicating basic authentication, or `cert` indicating client certificate-based authentication.

- Value type: String
- Example value: ba

TFIM.SOAP.Port
This parameter is a comma-separated list of port numbers.

- Value type: String
- Example value: 9443, 9445

SPS.WebSealPoc.ContextPoolSize
Specifies the number of PDContext objects available in the pool. This value reflects the number of clients that need to be authorized when using single sign-on.

You might need to increase the value based on the logout load of the system. When a large number of logouts occur at the same time, the Tivoli Federated Identity Manager runtime might run out of PDContext objects and logouts might start to fail. Because each PDContext object uses system resources, such as memory and file descriptors, care should be taken to select a value. The value must be greater than 0.
Default value: 5

- Value type: integer
- Example value: 5

SPS.WebSealPoc.DisablePDSignout
When set to true, this parameter disables the sign-out functionality of the single sign-on protocol service WebSEAL Point of Contact client. When the sign-out operation is invoked, it logs that no sign-out occurs and returns successfully. When this parameter is enabled, the single sign-on protocol service does not require the Tivoli Access Manager Java runtime (PDJRTE) to be configured.

Default value: false

- Value type: boolean
- Example value: true

SPS.WebSealPoc.Force.PdAdmin.Task
When set to true, this value forces the WebSeal Point of Contact callback to always use **pdadmin server** tasks to logout the user.

- Value type: boolean
- Example value: false

SPS.WebSealPoc.ContextPoolInitAttempts
This value represents the amount times that the PDContext objects initialization will be tried. The default is 1 and the value needs to be greater then 0.

- Value type: integer
- Example value: 1

SPS.WebSealPoc.ContextPoolInitTimeout
This value represents the maximum amount of time to be used during PDContext objects initialization. After the time has expired, the initialization will stop. The default is 10000 and the value needs to be greater then 0. The amount is on milliseconds.

- Value type: integer
- Example value: 10000

Custom properties for the trust service

Use the trust service custom properties to suit your deployment requirements

username.disable.password.validation
When set to true, this parameter causes the UsernameTokenSTSModule to skip password validation.

The default is false.

- Value type: Boolean
- Example value: true

username.jaas.provider.hostname
Specifies a name for the local host if WebSphere was not configured with the value of localhost for the host name.

The default is localhost.

- Value type: String
- Example value: localhost

username.jaas.provider.port
Specifies the port configured for the local WebSphere NameServer service.

The default is 2809.

- Value type: Integer
- Example value: 2809

pdjrte.context.min.pool.size
Specifies the minimum size of the Authorization context pool. This parameter is used by the UsernameTokenSTSModule. Set this parameter only if a performance evaluation requires it to be set.

- Value type: Integer
- Example value: 5

pdjrte.context.max.pool.size
Specifies the maximum size of the Authorization context pool. This parameter is used by the UsernameTokenSTSModule. Set this parameter only if a performance evaluation requires it to be set.

- Value type: Integer
- Example value: 50

ivcred.allow.groupUpdate
If set to true, attempts to modify the credential by adding groups.

Note: Do not use this parameter under any circumstances.

- Value type: Boolean
- Example value: false

ivcred.insert.CRLF76

When set to true, the base64 encoded `IVCred` generated by the Security Token Service module `STSTokenIVCred` is split into multiple lines. If this custom property is set to false, the base64 encoded `IVCred` generated by the Security Token Service module `STSTokenIVCred` is not split into multiple lines.

Default value: True

- Value type: Boolean
- Example value: False

saml.use.rst.lifetime

Directs the SAML modules to use the lifetime of the RequestSecurityToken element to derive the lifetime of the issued SAML assertion. When set to false, does not use the RequestSecurityToken lifetime.

Default value: false

- Value type: Boolean
- Example value: false

passticket.disable.uppercase.principal

Uses the local RACF® handler to direct the PassTicket Module not to transform all the principal name to uppercase before attempting to generate a PassTicket. When set to false, always raises the principal to uppercase for the local RACF handler.

Default value: false

- Value type: Boolean
- Example value: false

sts.use.issuer.saml20.sso

Directs the SAML 2.0 module to use the Issuer value, instead of the NameID NameQualifier value to look up an alias during a single sign-on operation.

Default value: false

- Value type: Boolean
- Example value: false

username.wss.namespace.override

If not specified, the default is the WSS 1.1 token profile namespace. The key for this property can be used as a prefix to set the scope of the property to a specific STS Chain; for example, `username.wss.namespace.override.uuid1234`.

- Value type: String
- Example value: *<a_URI_namespace>*

STS.validateMappingRules

Specifies whether the mapping rule is validated when it is imported through the console or the command-line interface. If the **STS.validateMappingRules** parameter is specified, and the value is equal to the string `false`, ignoring the case, then the mapping rule is not validated. Otherwise, the mapping rule is validated.

- Value type: Boolean
- Example value: false

authorizationsts.initial.num.context
Specifies the initial amount of context objects to be created at startup. This parameter controls the number of connections created and maintained by the pool.

- Value type: Integer
- Example value: 5

authorizationsts.max.num.context
Specifies the maximum amount of context objects to be created throughout. This parameter controls the number of connections created and maintained by the pool.

- Value type: Integer
- Example value: 10

Custom properties for OAuth 2.0

Use the OAuth 2.0 custom properties to suit your deployment requirements.

OAuth20.DoNotSendXFrameOptionsHeader
This parameter directs the OAuth 2.0 endpoints to not include the X-Frames-Options: SAMEORIGIN header in any responses to the OAuth client or the resource owner, in particular the consent for authorization page. This setting is turned off by default. Use this setting only when the authorization server and the OAuth client have a strong trust relationship.

By default, the **OAuth20.DoNotSendXFrameOptionsHeader** option does not exist. To use this setting, create the parameter, and set the value to **true**.

- Value type: Boolean
- Example value: True

Custom properties for SAML 1.0

Use the SAML 1.0 custom properties to suit your deployment requirements.

saml.use.legacy.clockskew.default
Tivoli Federated Identity Manager, by default, uses the local clock of the run time when validating SAML assertion timestamps. Set this parameter to **true** if you want to add a 60 second clock skew between the server and the SAML assertion timestamp.

Default value: False

- Value type: Boolean
- Example value: False

Custom properties for SAML 1.1

Use the SAML 1.1 custom properties to suit your deployment requirements.

SAML.AllowDebugMessages
When specified as true, and a SAML artifact resolution failure occurs, the SystemOut.log and SystemErr.log contains an informational message. In addition, the message contains extra debug information about the request that contained the failed artifact and provides a reason for the event.

Note: This message is only available in English.

Default value: False

- Value type: Boolean

- Example value: SAML.AllowDebugMessage=true

saml.use.legacy.clockskew.default
Tivoli Federated Identity Manager by default adds a clock skew of 60 seconds when validating the SAML assertion timestamps. To disable the 60 second default, add the custom property: `saml.use.legacy.clockskew.default = false`

Default value: True

- Value type: Boolean
- Example value: True

Custom properties for the key service

Use custom properties for the key service to suit your requirements.

kessjksservice.include.keyinfo.x509.certificate.data
Includes a base64 encoded certificate in the KeyInfo element of the signature. When this element is true, either by default or by explicit use of this property, then the other KESS runtime properties are ignored. When not specified, the default is true.

- Value type: boolean
- Example value: true

kessjksservice.include.keyinfo.x509.subject.key.identifier
Includes the subject key identifier in the KeyInfo element of the signature when the given certificate supports it. This can be used in addition to issuer.details and subject.name. When not specified, the default is false.

- Value type: boolean
- Example value: true

kessjksservice.include.keyinfo.x509.issuer.details
Adds X509 issuer details to the KeyInfo element of the signature. This can be used in addition to subject.key.identifier and subject.name. When not specified, the default is false.

- Value type: boolean
- Example value: true

kessjksservice.include.keyinfo.x509.subject.name
Adds the X509 subject distinguished name (DN) to the KeyInfo element of the signature. This can be used in addition to subject.key.identifier and issuer.details. When not specified, the default is false.

- Value type: boolean
- Example value: true

kessjksservice.exclude.inclusive.namespace.prefixes
A comma-separated list of prefix names. When set, the prefixes in the list are not added to the InclusiveNamespaces list that is in the Signature Element.

- Value type: string
- Example value: ds

kessjksservice.supportedalgorithms.signature
This custom runtime property is a list of signature algorithm URI separated by commas. The default value is the full set of signature algorithms supported by IBM Tivoli Federated Identity Manager.

You can modify this custom runtime property to include untested algorithms that your system supports, but are not supported by IBM. Be careful when modifying this property to avoid a signature failure.

Default value: http://www.w3.org/2000/09/xmldsig#rsa-sha1,http://www.w3.org/2000/09/xmldsig#dsa-sha1,http://www.w3.org/2001/04/xmldsig-more#rsa-sha256

- Value type: string
- Example value: http://www.w3.org/2000/09/xmldsig#rsa-sha1,http://www.w3.org/2000/09/xmldsig#dsa-sha1,http://www.w3.org/2001/04/xmldsig-more#rsa-sha256

kessjksservice.supportedalgorithms.messagedigest

This custom runtime property is a list of digest algorithm URIs separated by commas. The default value is the full set of digest algorithms supported by IBM Tivoli Federated Identity Manager.

You can modify this custom runtime property to include untested algorithms that your system supports, but are not supported by IBM. Be careful when modifying this property to avoid a signature failure.

Default value: http://www.w3.org/2000/09/xmldsig#sha1,http://www.w3.org/2001/04/xmlenc#sha256,http://www.w3.org/2001/04/xmlenc#sha512

- Value type: string
- Example value: http://www.w3.org/2000/09/xmldsig#sha1,http://www.w3.org/2001/04/xmlenc#sha256,http://www.w3.org/2001/04/xmlenc#sha512

key.selection.criteria

This custom runtime property allows you to configure the order of certificates or keys. Use these values for the custom property:

only.alias
Alias only: The selected key only, without Auto rollover. If the key is invalid, the software indicates failure. Configure the property to use the value

shortest.lifetime
Shortest lifetime: For signing, a valid key with the shortest available lifetime. For validation, key lifetime availability runs from shortest to longest.

longest.lifetime
Longest Lifetime: For signing, a valid key with the longest available lifetime. For validation, key lifetime availability runs from longest to shortest.

- Value type: string
- Example value: only.alias

Custom properties for a SOAP client

Use the SOAP custom properties to suit your deployment requirements.

com.tivoli.am.fim.soap.client.jsse.provider

The Java Secure Socket Extension (JSSE) provider name that should be used instead of IBMJSSE for SOAP client socket connections.

- Value type: String
- Example value: IBMJSSE

com.tivoli.am.fim.soap.client.jce.provider
The Java Cryptography Extension (JCE) provider name that should be used instead of IBMJCE for SOAP client keystores.

- Value type: String
- Example value: IBMJCE

com.tivoli.am.fim.soap.client.trust.provider
The Java Trust Manager provider algorithm name that should be used instead of IbmX509 for SOAP client Trust Managers.

- Value type: String
- Example value: IbmX509

Custom properties for SAML 2.0

Use the SAML 2.0 custom properties to suit your deployment requirements.

SAML.Assertion.IncludeNSPrefixList.DS
When specified as true, `ds` is included into the `PrefixList` attribute of the `InclusiveNamespaces` in the SAML assertion.

Default value: False

- Value type: Boolean
- Example value: True

SAML20.LogoutRequest.NotOnOrAfter.Enabled
When specified as true, the NotOnOrAfter attribute will be included on LogoutRequest messages from the identity provider to the service provider.

Default value: True

- Value type: Boolean
- Example value: True

SAML20.LogoutRequest.NotOnOrAfter.Lifetime
Specifies the time in seconds used to set the NotOnOrAfter attribute on a logout request.

Default value: 120

- Value type: Integer
- Example value: 300

saml.use.legacy.clockskew.default
Tivoli Federated Identity Manager by default adds a clock skew of 60 seconds when validating the SAML assertion timestamps. To disable the 60 second default, add the custom property: `saml.use.legacy.clockskew.default = false`

Default value: True

- Value type: Boolean
- Example value: True

SAML20.IDP.UnsolicitedSSO.RelayState.URLEncoding
When specified as true, the `RelayState` in an unsolicited authentication response is URL encoded by the identity provider before it is sent to the service provider. This configuration applies to a response that is sent using `HTTP POST` binding and `HTTP ARTIFACT` binding with the `HTTP POST` artifact delivery method.

The URL encoding can be controlled in three levels:

Global level
Controls the URL encoding for all federations and partners.

Configuration example: `SAML20.IDP.UnsolicitedSSO.RelayState.URLEncoding = true`

Federation level
Controls the URL encoding for a specific federation and all its partners.

Configuration example: `SAML20.IDP.UnsolicitedSSO.RelayState.URLEncoding_<FEDERATIONID> = true`

Example for SAML20.IDP.UnsolicitedSSO.RelayState.URLEncoding_<FEDERATIONID>:

```
SAML20.IDP.UnsolicitedSSO.RelayState.URLEncoding_
https://idp/sps/fed/saml20 = true
```

Partner level
Controls the URL encoding for a specific federation and a specific partner.

Configuration example: `SAML20.IDP.UnsolicitedSSO.RelayState.URLEncoding_<FEDERATIONID>_<PARTNERID>= true`

Example for SAML20.IDP.UnsolicitedSSO.RelayState.URLEncoding_<FEDERATIONID>_<PARTNERID>:

```
SAML20.IDP.UnsolicitedSSO.RelayState.URLEncoding_https://idp/
sps/fed/saml20_https://sp/sps/fed/saml20 = true
```

Default value: True

- Value type: Boolean
- Example value: False

`<FEDERATION>` represents the Provider ID of the federation and `<PARTNER>` represents the Provider ID of the partner. You can obtain the Provider ID of the federation from the Federation Properties page in the Console while the Provider ID of the partner can be obtained from the Partner Properties page in the Console.
You can use the three levels of control concurrently. Tivoli Federated Identity Manager implements concurrent use by checking the `RelayState` settings to decide what action to take in the following order:

1. Partner level setting
2. Federation level setting
3. Global level setting

SAML20.SP.UnsolicitedSSO.RelayState.URLEncoding
When specified as true, the `RelayState` in an unsolicited authentication response is URL decoded by the service provider after it is received from the identity provider.

The URL encoding can be controlled in three levels:

Global level
Controls the URL encoding for all federations and partners.

Configuration example:

```
SAML20.SP.UnsolicitedSSO.RelayState.URLEncoding = true
```

Federation level

Controls the URL encoding for a specific federation and all its partners.

Configuration example: `SAML20.SP.UnsolicitedSSO.RelayState.`

`URLEncoding_<FEDERATIONID> = true`

Example for SAML20.SP.UnsolicitedSSO.RelayState.

URLEncoding_<FEDERATIONID>:

```
SAML20.SP.UnsolicitedSSO.RelayState.URLEncoding_https://sp/sps/
fed/saml20 = true
```

Partner level

Controls the URL encoding for a specific federation and a specific partner.

Configuration example: `SAML20.SP.UnsolicitedSSO.RelayState.`

`URLEncoding_<FEDERATIONID>_<PARTNERID>= true`

Example for SAML20.SP.UnsolicitedSSO.RelayState.URLEncoding_

<FEDERATIONID>_<PARTNERID>:

```
SAML20.SP.UnsolicitedSSO.RelayState.URLEncoding_https://sp/sps/
fed/saml20_https://idp/sps/fed/saml20 = true
```

Default value: True

- Value type: Boolean
- Example value: False

`<FEDERATION>` represents the Provider ID of the federation and `<PARTNER>` represents the Provider ID of the partner. You can obtain the Provider ID of the federation from the Federation Properties page in the console while the Provider ID of the partner can be obtained from the Partner Properties page in the console.
You can use the three levels of control concurrently. Tivoli Federated Identity Manager implements concurrent use by checking the `RelayState` settings to decide what action to take in the following order:

1. Partner level setting
2. Federation level setting
3. Global level setting

Custom properties for the console

`STS.showSSOChains`

This parameter controls if the console allows an administrator to manage or modify chains that were generated automatically for single sign-on transactions. Setting this value to `false` does not disable the custom property. You must remove the key and value pair from the custom properties table.

- Value type: boolean
- Example value: true

`STS.showUSCChains`

This parameter controls if the console allows an administrator to manage or modify chains that were generated automatically for User Self Care federations. Setting this value to `false` does not disable the custom property. You must remove the key and value pair from the custom properties table.

- Value type: boolean

- Example value: true

STS.showAQChains
This parameter controls if the console allows an administrator to manage or modify chains that were generated automatically for SAML 2 federations that enable the Attribute Query service. Setting this value to `false` does not disable the custom property. You must remove the key and value pair from the custom properties table.
- Value type: boolean
- Example value: true

Custom property for OpenID

Use the OpenID custom properties to suit your deployment requirements.

OpenID.TrustedSitesManagerModuleID
A plugin module ID for a module that implements the com.tivoli.am.fim.protocols.openid_trusted_sites_manager extension point. There are two examples which implement this extension:
- TrustedSitesManagerCookieImpl
- TrustedSitesManagerMemoryImpl

When the parameter is not specified, the default is TrustedSitesManagerCookieImpl.
- Type: String
- Example value: TrustedSitesManagerCookieImpl

OPENID.DiscoveredInformationExpirationSeconds
Specifies the number of seconds for which discovered information for any OpenID user-supplied identifier is cached. If this value is less than or equal to zero, the data is not cached at all (default). This parameter controls a cache for discovered information. Use this parameter only when the same OP identifier logon is frequently used by a majority of the users of the system. For example, in an intranet deployment.

OPENID.SkipClaimedIdDiscovery
Controls whether verification is done on claimed identifiers when an OP-identifier logon is performed. This parameter is only set to true in an environment which uses trusted OP with the relying party. Otherwise, a security exposure exists. This parameter is typically used in an intranet environment.
- Type: Boolean
- Example value: False (default)

Custom property for transport security protocol

Specifying the transport security protocol for HTTPS connections

The IBM Tivoli Federated Identity Manager creates `SSL_TLS` as the default secure protocol for HTTPS connections. To change or override the default protocol, specify the following runtime custom property in the`fim.appservers.properties` file:

```
com.tivoli.am.fim.soap.client.ssl.protocol= PROTOCOL
```

PROTOCOL corresponds to one of the protocols supported by the Java Secure Socket Extension used by the underlying WebSphere Application Server.

Examples:

- `SSL_TLS`
- `SSL`
- `SSLv2`
- `SSLv3`
- `TLS`
- `TLSv1`

Note: The protocol examples might not necessarily be supported.

Custom properties for LTPA tokens

Specifying custom Tivoli Federated Identity Manager runtime properties that force compatible QName generation

WebSphere Application Server versions 6.0.2 and 6.1 do not distinguish between LTPA v1 and LTPA v2 tokens in Web Services. Only one BinarySecurityToken ValueType is supported for LTPA tokens, and the QName of the value type is:

```
http://www.ibm.com/websphere/appserver/tokentype/5.0.2#LTPA
```

When the Tivoli Federated Identity Manager STS issues an LTPA v2 token, the token is created with the following QName. This QName is correct, but it is not supported by WebSphere Application Server versions 6.0.2 and 6.1:

```
http://www.ibm.com/websphere/appserver/tokentype#LTPAv2
```

This APAR provides custom Tivoli Federated Identity Manager runtime properties that force compatible QName generation if needed. To enable compatibility mode, set either or both of the following custom runtime properties:

`ltpa.enable.compat.mode.[`*`chainid_uuid`*`]=true ltpa.enable.compat.mode=true`

where *chainid_uuid* is the value of the Chain UUID. For example:

ltpa.enable.compat.mode.[uuideb42e428-011b-1ebc-a0cb-9e6c4b35c1c7]=true

To determine the value of Chain UUID, select **Trust Service Chains** > **Select Action** > **Show Chain ID in column in table** from the administration console. This action selection causes a new column to appear in the table that displays the unique Chain ID.

Chapter 49. Customizing an authentication login form for single sign-on

Customize an authentication login form by adding parameters to a WebSphere or WebSEAL point of contact server profile.

When a user requests access to a single sign-on federation, the identity provider initiates single sign-on by authenticating the user. To authenticate the user, the identity provider uses a point of contact server to display a forms-based login page.

When an identity provider participates in multiple federations or hosts multiple partners in one federation, the administrator can customize the default login form.

As administrator, you can customize:

- The login page based on the contents of the requests sent by the service providers.
- The look and feel of the login form.
- The type of authentication required.
- The login pages for WebSEAL and WebSphere point of contact servers.

To customize the login page, use the Tivoli Federated Identity Manager administration console to configure a new point of contact server profile. In the new profile, add a parameter to the authentication callback, and specify one or more values for the parameter.

Tivoli Federated Identity Manager provides some parameters which are always available and consistent across all federation types and some which are specific to the type of federation.

The protocols which support protocol-specific parameters are:

- SAML 1.x
- SAML 2
- OpenID

The set of defined values are described in "Supported macros for customizing an authentication login form."

Task overview:

1. Review the supported values for your protocol type, and identify the ones you want to use. See "Supported macros for customizing an authentication login form."
2. Create a new point of contact server profile. See "Configuring a point of contact server to support customization of login pages" on page 720.

Supported macros for customizing an authentication login form

This topic describes the set of macros for customizing an authentication login form.

Tivoli Federated Identity Manager supplies contextual authentication parameters in customizing login forms. When using WebSEAL as the point of contact server, these are query-string parameters to the login page. For WebSphere, they are in the WASReqURL cookie when the login page is loaded. The parameters are macros in the configuration of the authentication callback for the point of contact server profile.

Note: When you use the WebSphere point of contact, the value of the query string parameter needs to be URL decoded twice.

Supported macros are:

- Protocol independent macros
- SAML protocol macros
- OpenID protocol macros
- OAuth protocol macros

Note: If the value of `authentication.macros` is longer than the permitted length of query string parameter, the WASReqURL cookie will not be present in the identity provider.

Protocol independent macros for customizing an authentication login form

The following macros are protocol independent and can be used regardless of the federation type used.

Table 157. Supported Protocol independent macros

Macro	Query-String Parameter name	Description
%FEDID%	FedId	Specifies a unique identifier (UUID) used internally by Tivoli Federated Identity Manager to identify the federation.
%FEDNAME%	FedName	Specifies the user-assigned name of the federation.

SAML protocol supported macros for customizing an authentication login form

The following macros are supported for SAML protocol. Macros are supported for both SAML 1.x and SAML 2.0, except as indicated.

Table 158. Supported SAML protocol macros

Macro	Query-String Parameter name	Description and value
%PARTNERID%	PartnerId	Represents the SSO partner that the user uses to sign in. SAML value: The value is the ProviderID of the partner.
%TARGET%	Target	Represents the target URL at the partner, if known. SAML value: The value is the value of the target parameter.

Table 158. Supported SAML protocol macros (continued)

Macro	Query-String Parameter name	Description and value
%SPRELAYSTATE%	SPRelayState	Supported for SAML 2.0 only. Represents RelayState data in accompanying the SSO request, if applicable. SAML value: The RelayState data that accompanies the SAML AuthnRequest.
%ACSURL%	AssertionConsumerURL	Represents the assertion consumer service URL of the partner, if applicable. SAML value: The value is the Partner ACS URL.
%AUTHNCONTEXT%	AuthnContext	**Supported for SAML 2.0 only** Represents the AuthnContext in request (if applicable). SAML value: The value is a base-64 encoded string representing the XML from the RequestedAuthnContext in the SAML AuthnRequest (if present).
%SSOREQUEST%	SSORequest	**Supported for SAML 2.0 only** Represents the entire SSO request (if applicable). SAML value: The value is a base-64 encoded string representing the XML from the entire SAML AuthnRequest.
%FORCEAUTHN%	ForceAuthn	**Supported for SAML 2.0 only** The value `true` or `false`. SAML value: If the ForceAuthn flag is set in the SAML 2 SSO request causing the user to re-authenticate, the value is `true`. Otherwise the value is `false`.

OpenID supported macros for customizing an authentication login form

The following macros are supported for the OpenID protocol.

Table 159. Supported OpenID protocol macros

Macro	Query-String Parameter name	Description and value
%PARTNERID%	PartnerId	Represents the SSO partner that the user uses to sign in. OpenID value: The value of the openid.trustroot parameter.
%TARGET%	Target	Represents the target URL at the partner, if known. OpenID value: The value of the openid.return_to parameter.
%SSOREQUEST%	SSORequest	Represents the entire SSO request (if applicable). OpenID value: The checkid_setup request as a base64-encoded version of the url-encoded SSO request.

Table 159. Supported OpenID protocol macros (continued)

Macro	Query-String Parameter name	Description and value
%UNSATISFIEDPAPEPOLICIES%	UnsatisfiedPapePolicies	Represents a list of strings which represent PAPE policies. These strings are returned as "not yet satisfied" by the identity provider mapping rule in an OpenID identity provider federation. OpenID value: PAPE policies returned in the ContextAttributes Attribute openid.pape.to_be_satisfied_auth_policies
%FORCEAUTHN%	ForceAuthn	Specifies if authentication on the identity provider is forced. The values are `true` or `false`. OpenID value: The value is `true` if one of these criteria is satisfied: • the PAPE max_auth_age was zero (meaning forced to authenticate again) • the IDP mapping rule on the OpenID identity provider is forcing authentication due to unsatisfied PAPE policies • the authentication time returned by the IDP mapping rule does not satisfy the (non-zero) max_auth_age requested by the RP Otherwise, the value is `false`.

OAuth protocol supported macros for customizing an authentication login form

The following table indicates how an OAuth federation populates the authentication macros.

Table 160. Supported OAuth protocol macros

Macro	Query-String Parameter name	Description and value
%PARTNERID%	PartnerId	The OAuth unique client identifier.
%TARGET%	Target	OAuth client redirection URI.
%SSOREQUEST%	SSORequest	A base-64 encoded string representing the query and body parameters from the OAuth request.

Configuring a point of contact server to support customization of login pages

This topic describes how to a configure custom point of contact server to support customization of a login page.

Before you begin

Ensure that you:

- Understand how customized login pages are supported. See Chapter 49, “Customizing an authentication login form for single sign-on,” on page 717.
- Know which macros to specify for the authentication callback parameter. See “Supported macros for customizing an authentication login form” on page 717.

Note: You do not need to create and publish a custom Point of Contact callback plug-in before specifying authentication macros. Support for authentication macros is provided by default. When you run the configuration wizard, you can ignore the message that states that you must publish a plug-in before using the wizard.

About this task

The following procedure describes how to add a custom point of contact server that is like a point of contact server already defined in your environment in order

to modify the information shown in a login page.

Procedure

1. Log on to the administration console.
2. Click **Tivoli Federated Identity Manager** > **Domain Management** > **Point of Contact**.
3. Select the existing point of contact server that you want to base your new point of contact server on. You must select a profile for either WebSEAL or **WebSphere**.
4. Click **Create Like** to open the Welcome Panel of the Point of Contact Profile wizard.
5. Click **Next** to open the Profile Name panel. It shows information from the profile on which you are basing your new point of contact server.
6. Enter a name for the profile.
7. (Optional) Enter a description.
8. Click **Next**. The Sign in panel opens.
9. Accept the default entries for the sign-in callbacks, the parameters for each callback, and the order in which they are used.
10. Click **Next**.
11. Accept the default entries for the Sign out panel.
12. Click **Next**.
13. Accept the default entries for the Local ID panel.
14. Click **Next**.
15. Click **Add Parameters** in the Callback Parameters section on the Authentication panel.
16. Enter `authentication.macros` at Name.
17. Enter the macros you want to use at Values. To specify multiple values and separate the macros, place a backslash (\) and a comma between values. For example: `%FEDID%\,%FEDNAME%\,%PARTNERID%`
18. Click **Next** to display the Summary panel. It lists all the callbacks and parameters you specified in the preceding steps.
19. Click **Finish** to complete the setup or click **Back** to return to the previous panels and revise your selections.
20. Click **Current Domain portlet**.
21. Click **Load configuration changes to the Tivoli Federated Identity Manager runtime**.

What to do next

"Activating a point of contact server" on page 746

Pass SAML request element to the point of contact server

Use a callback parameter to pass specific SAML request elements to the point of contact server. Pass specific SAML request elements in addition to other macros.

In a typical single sign-on event, attributes are passed as query string parameters in the redirect URL. This feature uses a callback parameter in addition to existing supported macros. For more information about customizing an authentication login form, see Customizing an authentication login form for single sign-on.

Passing specific attributes from a SAML request requires the **extended.authentication.macros** parameter. Its value is a set of key-value pairs that are separated by a slash (\). For each pair, the key and the value are separated by the = character.

Important: Do not put a white space before and after the \ and = characters.

Each pair represents an attribute that is passed to the point of contact. Each attribute is passed to the point of contact as a query string parameter.

The key of a pair consists of two parts:

- The first part is the protocol name where the pair is applicable.
- The second part is the name of the query string parameter.

A period (.) separates the two parts. The value of a pair is the protocol-specific method for selecting the attribute.

The following example is the Backus-Naur Form (BNF) format of the callback parameter `extended.authentication.macros`:

```
<callback-parameter-value>    ::= <key-value-pairs>
<key-value-pairs>             ::= <key-value-pair> | <key-value-pair> "\," <key-value-pairs>
<key-value-pair>              ::= <key> "=" <value>
<key>                         ::= <protocol-name> "." <query-string-parameter-name>

<protocol-name>               ::= name of the protocol where the pair is applicable
<query-string-parameter-name> ::= name of the query string parameter
<value>                       ::= method for selecting the attribute/element
```

Note: Currently, only SAML 2.0 protocol is supported. The protocol name is `SAML20`. The method for selecting the attribute is an XPath path expression. Currently, only the canonical form XPath 1.0 is supported.

The following example for the callback parameter shows a sample value of the parameter:

extended.authentication.macros

Value:

```
SAML20.AssertionConsumerServiceURL=/samlp:AuthnRequest/@AssertionConsumerServiceURL\,
SAML20.AuthnRequestAttributes=/samlp:AuthnRequest/@*\,
SAML20.Issuer=/samlp:AuthnRequest/saml:Issuer\,
SAML20.AuthnRequestElements=/samlp:AuthnRequest/*
```

Each attribute is passed to the point of contact as a query string parameter. The value of the second part of the key of a pair is the query string parameter name. The value of a pair selects the attribute that is passed to the point of contact.

If a single attribute is selected, the canonical form of this attribute is BASE-64 encoded. The result is used as the query string parameter value. If multiple attributes are selected, the string representation of each attribute is BASE-64 encoded. The encoded strings are concatenated with a comma (,) as the separator. The result is the query string parameter value. Only the canonical form XML-C14 that is specified in the World Wide Web Consortium is supported.

Both the query string parameter name and the query string parameter value are URL encoded before they are appended into the redirect URL.

Chapter 50. Customizing single sign-on event pages

Tivoli Federated Identity Manager generates files that are displayed in response to events that occur during single sign-on requests. The response displayed might be a form (such as when login information is required) or an error or information statement about a condition that occurred while the request was processed.

You have the option of customizing the event pages, as follows:

- Modifying their appearance or content.
- Specifying which geographic or language locale to use when the pages are displayed.

Before continuing with the customization, you should have a thorough understanding of how event pages are generated and displayed. See "Generation of event pages."

Generation of event pages

Event pages are displayed in response to events that occur during single sign-on requests. They usually contain a form (such as a prompt for user name and password information) or text (such as an informational or error message).

Event pages are dynamic pages that are generated by Tivoli Federated Identity Manager using the following information:

Template files
: XML or HTML files that are provided with Tivoli Federated Identity Manager and contain elements, such as fields, text, or graphics, and sometimes macros that are replaced with information that is specific to the request or to provide a response to the request.

Page identifiers
: Event information that corresponds to one or more template files. Each page identifier corresponds to a specific event condition, such as a specific error or a condition in which a message or a form must be displayed. To create an event page, page identifiers are mapped to one or more template files. The mapping function allows multiple page identifiers to point to the same template file.

Message catalogs
: Text that is used to replace macros in the template files.

When a request is received, the appropriate response page is generated as follows:

1. Processing of the request occurs and a response to an event is required.
2. Template files and page identifiers are read from the file system.
3. Macros in the template files are replaced with values that are appropriate for the response that is needed.
4. An appropriate event page is generated.
5. The generated event page is displayed.

For information about the relationship between page identifiers and template files, see "Page identifiers and template files" on page 724.

Page identifiers and template files

A page identifier specifies an event and each event corresponds to one or more *template files*. Some page identifiers are specific to the specification (such as SAML 1.x) and some are general.

To modify the text, graphics, or other elements of the page that is shown for an event, follow the succeeding tasks:

1. Modify the template file or copy a template file.
2. Use the copy as the basis for a new file.
3. Map the event to that new file.

General page identifiers and their template files

Table 161. General page identifiers and their template files

Page identifier (Event)	Description	Template file
/proper/errors/noprotdet	Shows when protocol is unknown	/proper/errors/noprotdet.html
/proper/errors/missing_component	Shows when protocol is unknown	/proper/errors/ missingcomponent.html
/proper/errors/protocol_error	Shows when a protocol module throws an exception	/proper/errors/protocol_error.html
/proper/errors/need_authentication	Shows when the initial URL information is not found on the user session.	/proper/errors/ need_authentication.html
/proper/errors/access_denied	Shows when access is denied.	/proper/errors/access_denied.html
/proper/errors/missing-initial-url.html	Shows when the initial URL information is not found on the user session.	/proper/errors/allerror.html
/proper/errors/unauth-access-to-waspoc-delegate.html	Shows when the WebSphere point of contact delegate protocol has been accessed without appropriate authentication.	/proper/errors/allerror.html
/proper/login/formlogin.html	Shows when using form-based authentication. **Attention:** Do not change the action value and parameter names for the form POST. They must remain unchanged for the form to function properly.	/proper/login/formlogin.html
/proper/login/formloginerror.html	Shows when an error occurs using the formlogin.html file. See “Customizing the login form” on page 98.	/proper/login/formloginerror.html
/proper/genericpoc/ login_success.html	Shows when the generic point of contact implementation performs a successful login without a target URL.	/proper/login/login_success.html
/proper/waspoc/login_success.html	Shows when the WebSphere point of contact implementation performs a successful login without a target URL.	/proper/login/login_success.html

Table 161. General page identifiers and their template files (continued)

Page identifier (Event)	Description	Template file
/proper/waspoc/login_failure.html	Shows when an error occurs during login using the WebSphere point of contact implementation.	/proper/login/login_failure.html

SAML 1.x page identifiers and their template files

Table 162. SAML 1.x page identifiers and their template files

Page identifier (Event)	Description	Template file
/saml/invalid_request.html	Shows when a request is not valid.	/saml/allerror.html
/saml/unknown_sp.html	Shows when an unknown service provider is encountered.	/saml/allerror.html
/saml/unknown_ip.html	Shows when an unknown identity provider is encountered.	/saml/allerror.html
/saml/invalid_ip_request.html	Shows when an identity provider provides an invalid request.	/saml/allerror.html
/saml/unauth_user.html	Shows when the running user has not authenticated.	/saml/allerror.html
/saml/cannot_exchange_for_sp.html	Shows when there is an error encountered during the token exchange.	/saml/allerror.html
/saml/no_ip_post_page.html	Shows when the identity provider does not have a POST page.	/saml/allerror.html
/saml/no_return_token.html	Shows when there is no return token.	/saml/allerror.html
/saml/ip_post_to_sp.html	Shows the POST HTML form when the identity provider posts the SAML response to the service provider.	/saml/allerror.html
/saml/invalid_response.html	Shows when an invalid response message is encountered.	/saml/allerror.html
/saml/ip_response_invalid.html	Shows when identity provider response is invalid.	/saml/allerror.html
/saml/ cannot_exchange_for_resource.html	Shows when there is an error encountered during the token exchange.	/saml/allerror.html
/saml/missing_context_attribute.html	Shows when the required context attribute is not represented.	/saml/allerror.html
/saml/ missing_config_parameter.html	Shows when a required SPS configuration item is missing.	/saml/allerror.html
/saml/ could_not_retrieve_assertion.html	Shows when the service provider cannot get the assertion from the Response or from the SOAP back channel.	/saml/allerror.html
/saml/ could_not_perform_local_auth.html	Shows when an error is encountered when the EAI header is returned.	/saml/allerror.html
/saml/ could_not_create_signed_request.html	Shows when a signed SAML assertion request cannot be generated.	/saml/allerror.html

Table 162. SAML 1.x page identifiers and their template files (continued)

Page identifier (Event)	Description	Template file
/saml/sp_missing_target.html	Shows at the service provider if the initial request to the WAYF endpoint does not contain a TARGET parameter.	/saml/allerror.html
/saml/error_parsing_soap_response.html	Shows when there is an error encountered when the service provider attempts to retrieve the Assertion from the SOAP endpoint of the identity provider.	/liberty/error_parsing_soap_response.html
/saml/unknown_ip_wayf.html	Shows when the "where you are from" cookie contains an identity provider ID that is not configured on the federation.	/saml/allerror.html

SAML 2.0 page identifiers and their template files

Table 163. SAML 2.0 page identifiers and their template files

Page identifier	Description	Template files
/saml20/error_building_msg.html	Shows for errors building SAML 2 messages.	/saml20/error_building_msg.html
/saml20/error_missing_config_param.html	Shows when an invalid configuration parameter is detected at runtime.	/saml20/error_missing_config_param.html
/saml20/error_sending_msg.html	Shows for errors sending SAML 2 messages.	/saml20/error_sending_msg.html
/saml20/error_validating_msg.html	Shows for errors validating SAML 2 messages.	/saml20/error_validating_msg.html
/saml20/error_validating_art.html	Shows for errors validating SAML 2 artifacts.	/saml20/error_validating_art.html
/saml20/invalid_msg.html	Shows for errors validating SAML 2 messages.	/saml20/invalid_msg.html
/saml20/invalid_art.html	Shows for errors validating SAML 2 artifacts.	/saml20/invalid_art.html
/saml20/authn_failed.html	Shows when a SAML 2 authentication fails.	/saml20/authn_failed.html
/saml20/logout_failed.html	Shows for logout failures.	/saml20/logout_failed.html
/saml20/art_exchange_failed.html	Shows when exchange of a SAML artifact for a response fails.	/saml20/art_exchange_failed.html
/saml20/nimgmt_update_failed.html	Shows for name identifier management update failure.	/saml20/nimgmt_update_failed.html
/saml20/nimgmt_terminate_failed.html	Shows for name identifier management terminate failure.	/saml20/nimgmt_terminate_failed.html
/saml20/error_validating_msg_signature.html	Shows for errors validating SAML 2 message signatures.	/saml20/error_validating_msg_signature.html
/saml20/error_decrypting_msg.html	Shows for errors decrypting SAML 2 messages.	/saml20/error_decrypting_msg.html
/saml20/error_parsing_msg.html	Shows for errors parsing SAML 2 messages.	/saml20/error_parsing_msg.html

Table 163. SAML 2.0 page identifiers and their template files (continued)

Page identifier	Description	Template files
/saml20/error_parsing_art.html	Shows for errors parsing SAML 2 artifacts.	/saml20/error_parsing_art.html
/saml20/invalid_init_msg.html	Shows for errors validating SAML 2 messages.	/saml20/invalid_init_msg.html
/saml20/ error_validating_init_msg.html	Shows for errors validating SAML 2 messages.	/saml20/ error_validating_init_msg.html
/saml20/logout_success.html	Shows for successful logouts.	/saml20/logout_success.html
/saml20/logout_partial_success.html	Shows for partial logout completion.	/saml20/logout_partial_success.html
/saml20/ nimgmt_terminate_success.html	Shows for name identifier management terminate success.	/saml20/ nimgmt_terminate_success.html
/saml20/ nimgmt_update_success.html	Shows for name identifier management update success.	/saml20/ nimgmt_update_success.html
/saml20/consent_to_federate.html	Shows and prompts a user for consent to federate.	/saml20/consent_to_federate.html
/saml20/saml_post_artifact.html	Shows for sending SAML 2.0 artifacts for POST profiles.	/saml20/saml_post_artifact.html
/saml20/saml_post_request.html	Shows for sending SAML 2.0 requests for POST profiles.	/saml20/saml_post_request.html
/saml20/saml_post_response.html	Shows for sending SAML 2.0 responses for POST profiles.	/saml20/saml_post_response.html
/saml/ could_not_create_signed_request.html	Shows when a signed SAML assertion request cannot be generated.	/saml/allerror.html
/saml/sp_missing_target.html	Used at the service provider if the initial request to the WAYF endpoint does not contain a TARGET parameter.	/saml/allerror.html

Liberty page identifiers

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

Table 164. Liberty page identifiers

Page identifier	Description
/liberty/error_parsing_soap_response.html	Reports that SOAP response cannot be parsed.
/liberty/fed-terminate-success.html	Displayed when termination is successful.
/liberty/lib-cant-modify-alias.html	Displayed when alias modification fails.
/liberty/lib-fed-consent.html	Sent to ask user for consent to federate.
/liberty/lib-fed-post-request.html	Form used for POSTing an authentication request.
/liberty/lib-fed-post.html	Form used for POSTing a response.
/liberty/lib-internal-error-page.html	Sent for an error if nothing else can be sent.
/liberty/lib-ipi-consent.html	Asks user to consent to perform IP introduction to the service providers.

Table 164. Liberty page identifiers (continued)

Page identifier	Description
/liberty/lib-ipi-post.html	Reports IP introduction success.
/liberty/lib-login-failed-page.html	Not currently used.
/liberty/lib-logout-failed-page.html	Sent to user by the IP when logout failed for any reason.
/liberty/lib-logout-page.html	Sent to user to report all session terminations after a logout.
/liberty/lib-logout-success-page.html	Sent to user by the IP to report a successful logout.
/liberty/logoutFailure.gif	Image to indicate logout failure if the HTTP GET single logout technique is being used.
/liberty/logoutSuccess.gif	Image to indicate logout success if the HTTP GET single logout technique is being used.
/liberty/lib-message-timestamp-failure.html	Sent if the issue time is outside tolerance.
/liberty/lib-no-fed-exists.html	Sent when no federation exists.
/liberty/lib-no-liberty-assertion.html	Reports no assertion found in response.
/liberty/lib-no-local-login.html	Reports failure of local login.
/liberty/lib-no-service-available.html	Reports no assertion or alias service exists.
/liberty/lib-register-name-identifier-success.html	Reports successful registration of a name identifier.
/liberty/lib-request-id-not-matching-resp.html	Reports that a response does not correlate to any known request.
/liberty/lib-sig-validation-failure.html	Not currently used.
/liberty/lib-version-mismatch.html	Not currently used.
/pages/itfim/wayf/wayf-html.html	HTML WAYF response.

WS-Federation page identifiers

Table 165. WS-Federation page identifiers

Page identifier	Description
/wsfederation/cannot_exchange_for_resource.html	Reports that IP WS-Trust request failed on the service provider.
/wsfederation/cannot_exchange_for_sp.html	Reports that IP cannot exchange a token for the service provider.
/wsfederation/cannot_local_auth.html	Used when the service provider cannot validate a token.
/wsfederation/invalid_ip_response.html	Reports that the service provider cannot understand an IP response.
/wsfederation/invalid_request.html	Not a WS-Federation request.
/wsfederation/invalid_sp_request.html	Shows when a request is not valid.
/wsfederation/ip_post_to_sp.html	Used by WS-Federation for sending information from the IP to the service provider.
/wsfederation/no_ip_post_page.html	Shows when the identity provider does not have a post page.

Table 165. WS-Federation page identifiers (continued)

Page identifier	Description
/wsfederation/no_return_token.html	Reports that IP cannot find a token to return to the service provider.
/wsfederation/signout_cleanup_failed.html	Not currently used.
/wsfederation/ signout_cleanup_failed_no_auth.html	Used when WS-Federation signout failed because the user was not authenticated.
/wsfederation/signout_cleanup_to_sp.html	Used by WS-Federation signout to trigger signouts on service providers.
/wsfederation/signout_successful.html	Used when WS-Federation signout is successful.
/wsfederation/sp_ip_returned_fault.html	Reports that fault returned by IP to the service provider.
/wsfederation/unauth_user.html	Reports that user not authenticated on this IP.
/wsfederation/unknown_ip_wayf.html	Reports that the service provider cannot determine the IP.
/wsfederation/unknown_sp.html	Reports that the service provider is unknown to the IP.

Low-level independent page identifiers

Table 166. Independent page identifiers

Page identifier	Description
/proper/errors/cannot_process	Used for unspecified internal errors.
/proper/errors/missing_component	Shows when protocol is unknown.
/proper/errors/noprotdet	Shows when protocol is unknown.
/proper/errors/not_started	Used when the SPS is not running, which usually indicates a configuration error of some type.
/proper/errors/protocol_error	Shows when a protocol module throws an exception.
/pages/itfim/wayf/error-no-ips.html	Reports that no identity providers exist, so WAYF processing cannot be done.
/pages/itfim/wayf/error-missing-template.html	Used when no WAYF template page can be found.
/pages/itfim/wayf/error-invalid-template.html	Used when the WAYF page is invalid.
/pages/itfim/wayf/wayf-html.html	Shows when a federation has more than one identity provider and the ITFIM_WAYF_IDP query-string parameter or the WAYF cookie is not present.

Location of template files

The template files are stored in the succeeding directory by default:

AIX

`/usr/IBM/FIM/pages/`*`locale`*`/`

Linux or Solaris

`/opt/IBM/FIM/pages/locale/`

Windows

`C:\Program Files\IBM\FIM\pages\locale\`

The locale subdirectory is specific to the geographic or language locales of the template files. The default locale directory is named `C` and all files are in English. If a language pack was installed, additional locales are available.

The template files are published from their default subdirectories into WebSphere Application Server directories. See "Publishing updates" on page 735.

Attention: If you must modify the template files, modify them on the Tivoli Federated Identity Manager server. *Do not* modify them in the WebSphere Application Server directories.

Content of template files

HTML template files can contain macros that are replaced with context-specific information that is retrieved when the response page is built and returned. If your template file contains, for example, the macro `@EXCEPTION_MSG@`, an exception message is included in the response page.

The presence of a macro in a template file does not guarantee that the macro contains an actual value when the response page is built. A value for the macro must be defined when the page is built in order for the macro to return a value.

When customizing an HTML template file, use only those macros defined in the template file. If you add new macros to the template file, values for the added macros will not be returned when the final response page is generated.

Macros use the succeeding format:

```
@MACRO@
```

Where *MACRO* represents the name of the macro; for example, `@EXCEPTION_MSG@`

The succeeding macros are used in the template files.

Table 167. Macros used in the template files

Substitution macro	Brief Description
@ACTION@	The action is the URL where the form that contains the POST response is sent. Used in an HTML POST response sent by an identity provider to a browser for a single sign-on protocol service request.
@CAUSE@	Information regarding the cause of the error.
@DETAIL@	Additional information about an error or exception that has occurred as part of request processing. Because additional text is not always available, even if the @DETAIL@ macros is used in an HTML template file, there is no guarantee that the macros can provide additional text about the exception.
@EXCEPTION_MSG@	Text message that describes an exception that has occurred in request processing.

Table 167. Macros used in the template files (continued)

Substitution macro	Brief Description
@EXCEPTION_STACK@	Full exception stack of an exception that has occurred in request processing.
@FEDERATION_DISPLAY@	The name of the current federation, that is, the one currently in use.
@FEDERATION_ID@	The unique identifier of the current federation.
@PARTNER_ID@	Federation single sign-on protocol of a federation partner.
@REQ_ADDR@	Internet Protocol (IP) address of the endpoint that requested a federation action.
@RESPONSE@	Used in an HTML POST response from an identity provider, substituted for with the SAML response.
@SAMLSTATUS@	Collection of SAML status values received during the single sign-on action processing.
@SOAP_ENDPOINT@	The SOAP endpoint URL that is used to retrieve the assertion using a SAML artifact.
@TARGET@	Used to provide the service provider target in an HTML POST response sent by an identity provider) to a browser for a single sign-on protocol service request.
@TIMESTAMP@	A value for the current time.
@TOKEN:form_action@	The URL where the form that contains the POST message is sent during a POST binding.
@TOKEN:IPDisplayName@	The identity provider unique name.
@TOKEN:IPProviderID@	The identity provider unique identifier.
@TOKEN:PartnerID@	The partner unique identifier.
@TOKEN:RelayState@	The SAML protocol RelayState value.
@TOKEN:SamlMessage@	The base64 encoded SAML message that is sent on a form.
@TOKEN:SPDisplayName@	The service provider unique name.
@TOKEN:SPProviderID@	The service provider unique identifier.
@TOKEN:UserName@	The authenticated user name that submitted the single sign-on action.
@WAYF_FEDERATION_DISPLAY_NAME@	Name that shows for the current federation, as presented in the console. Used on a page presenting a WAYF (Where Are You From) challenge and requesting that a user selects an identity provider.
@WAYF_FEDERATION_ID@	Identifier of the current federation in the configuration file. Used on a page presenting a WAYF challenge and requesting that a user selects an identity provider.
@WAYF_FORM@	Identifier information for the WAYF HTML form that is presented to a user to acquire identity provider information in an SPS action where the identity provider for the requestor has not yet been determined (it is not yet in the cookie presented).
@WAYF_FORM_ACTION@	Endpoint of the single sign-on protocol service; this is the originally requested address (URL). Used on a page presenting a WAYF challenge and requesting that a user selects an identity provider.

Table 167. Macros used in the template files (continued)

Substitution macro	Brief Description
@WAYF_FORM_METHOD@	HTTP method used on a request that has resulted in a WAYF on a page that is prompting for identity provider information. The method can be either GET, POST or HEAD.
@WAYF_FORM_PARAM_ID@	Identifier of the form parameter for the current identity provider, is typically the configured cookie name. Used on a page that is presenting a WAYF challenge and requesting that a user selects an identity provider.
@WAYF_HIDDEN_NAME@	Name of one of the initial parameters to a request that results in a WAYF and is used on a page that is prompting for identity provider information.
@WAYF_HIDDEN_VALUE@	Value of one of the initial parameters to a request that results in a WAYF, and is used on a page that is prompting for identity provider information.
@WAYF_IP_DISPLAY_NAME@	The configured display name of the current identity provider on a page presenting a WAYF challenge.
@WAYF_IP_ID@	Configuration ID of the current identity provider on a page presenting a WAYF challenge.

Template page for the WAYF page

The Where Are You From (WAYF) page is used at the service provider. The WAYF page enables users to select their identity provider if there is more than one configured in the federation.

When a user arrives at a service provider, a WAYF identifier can be delivered through a cookie or query-string parameter with the request. The entity ID of the identity provider is stored as the value of the cookie or query-string parameter. If the WAYF identifier cookie or query-string parameter is not present, the WAYF page opens.

An example URL that includes the query string parameter for WAYF:

```
https://sp.host.com/FIM/sps/samlfed/saml20/
logininitial?RequestBinding=HTTPRedirect&ResponseBinding
=HTTPPost&ITFIM_WAYF_IDP=https://idp.host.com/FIM/sps/samlfed/saml20
```

This example is for a SAML 2.0 single sign-on URL. The query string parameter name is `ITFIM_WAYF_IDP`. The value of the identity provider ID is `https://idp.host.com/FIM/sps/samlfed/saml20`.

The WAYF page requires the user to indicate where they came from. If the user is not logged on to their identity provider, they are asked to log on. Depending on the attributes passed, the service provider can grant or deny access to the service.

The template pages are stored in the following directory by default:

<FIM_Install_Dir>`/pages/`*<locale>*`/pages/itfim/wayf`

See, “Low-level independent page identifiers” on page 729 for more information about WAYF template pages.

Administrators can use this page without modifications, but in some cases might want to modify the HTML style to match their specific deployment environment.

This template file provides several replacement macros:

@WAYF_FORM_ACTION@
This macro is replaced with the endpoint of the original request. This macro does not belong within a repeatable section.

@WAYF_FORM_METHOD@
This macro is replaced with the HTTP method of the original request. This macro does not belong within a repeatable section.

@WAYF_FORM_PARAM_ID@
This macro is replaced with ID used by the action for the identity provider. This macro is repeated once for each identity provider.

@WAYF_IP_ID@
This macro is replaced with the unique ID of the identity provider. This macro is repeated once for each identity provider.

@WAYF_IP_DISPLAY_NAME@
This macro is replaced with the configured display name of the identity provider. This macro is repeated once for each identity provider.

@WAYF_HIDDEN_NAME@
This macro is replaced with the name of the hidden parameter. This macro is repeated once for each original request parameter and is hidden.

@WAYF_HIDDEN_VALUE@
This macro is replaced with the value of the hidden parameter. This macro is repeated once for each original request parameter and is hidden.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!--
html wayf template that presents the choice as radio buttons.
-->
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
        <title>Where are you from</title>
    </head>
    <body style="background-color:#ffffff">
        <div>
            <!--
            Insert the federation ids here just so we can show some tokens
            [RPT federations]
                @WAYF_FEDERATION_ID@
                @WAYF_FEDERATION_DISPLAY_NAME@
            [ERPT federations]
            -->
            <form id="wayfForm" name="wayfForm"
                    action="@WAYF_FORM_ACTION@" method="@WAYF_FORM_METHOD@">
                <div>
                    <table>

                    [RPT ips]
                        <tr>
                            <td>
                                <input type="radio" id="@WAYF_FORM_PARAM_ID@"
                                    name="@WAYF_FORM_PARAM_ID@"
                                    value="@WAYF_IP_ID@"/>@WAYF_IP_DISPLAY_NAME@
                            </td>
                        </tr>
                    [ERPT ips]

                    </table>
                    <!-- the hidden inputs must be present -->

                    [RPT hidden]
                    <input type="hidden" name="@WAYF_HIDDEN_NAME@"
                        id="@WAYF_HIDDEN_NAME@"
                        value="@WAYF_HIDDEN_VALUE@"/ >
                    [ERPT hidden]

                </div>
                <input type="submit" name="submit" value="Submit" />
            </form>
        </div>
    </body>
</html>
```

Figure 78. Template page wayf-html.html

Modifying or creating the template files

To customize the appearance of the event pages, you can modify the template files or create new template files.

Before you begin

Before continuing with this procedure, be sure that you are familiar with how event pages are generated. See "Generation of event pages" on page 723.

About this task

Attention: Modify the template files in the directory on the Tivoli Federated Identity Manager server (as described below). When all of your changes are complete, publish them to the WebSphere Application Server configuration repository directory. *Do not* edit these files in the configuration repository.

Procedure

1. Decide which event pages you want to modify. See the list of events and their corresponding template files at "Page identifiers and template files" on page 724.
2. Stop the WebSphere Application Server where the runtime component is installed. Use the **stopServer** command. See the WebSphere Information Center if you need help.
3. Locate the template file that corresponds to the event page you want to modify or make a copy of an existing template file and use it create a new file.

   The template files are located in a geographic-specific or language-specific locale subdirectory for the file. The default locale subdirectory is named `C` and all files are in English. If a language pack was installed, additional locales are available. You can also create your own locales, as described in "Creating a page locale" on page 736.

   The default directory for the template files is as follows:

   AIX

   `/usr/IBM/FIM/pages/`*`locale`*`/`

   Linux or Solaris

   `/opt/IBM/FIM/pages/`*`locale`*`/`

   Windows

   `C:\Program Files\IBM\FIM\pages\`*`locale`*`\`
4. Use a text editor to modify or create new files.
5. Save the files to the appropriate location, such as the same directory where you edited them or from which you copied them.

What to do next

When you have completed this step, continue with publishing the files to the configuration repository as described in "Publishing updates."

Publishing updates

When all updates and additions have been made to the template files, you must publish the files to the configuration repository so they will be displayed.

Procedure

1. Log on to the management console.
2. Select **Tivoli Federated Identity Manager** > **Domain Management** > **Event Pages**.
3. Locate the event or events that you want to map to the new or updated pages.

4. In the **HTML Page Displayed** field for each event you are modifying, type the path and file name for the file you want to use for that event.
5. Click **Apply**. A warning message is displayed that explains you must publish the updated files to the configuration repository.
6. Click **Publish Pages** to publish the changes right away.

   Otherwise, click **Close** and later, when you are ready to publish click **Domain Management** > **Runtime Node Management** and on the Runtime Node Management panel, click the **Publish pages** button.

Creating a page locale

The template files that are used to generate event pages are located in a geographic-specific or language specific locale subdirectory for the file. The default locale subdirectory is named `C` and all files are in English. Additional locales and corresponding languages are also available. In addition, you can create your own locale.

Before you begin

Before continuing with this procedure, be sure that you are familiar with how event pages are generated. See "Generation of event pages" on page 723.

Procedure

1. Log on to the management console.
2. Select **Tivoli Federated Identity Manager** > **Domain Management** > **Event Pages**. The Event Pages panel opens.
3. Click the **Page Locale** tab to open the Page Locale panel.
4. Click **Create**. A placeholder list item is added to the list of page locales with the Page Locale name of **locale** and a Page Root Directory of **page_root**.
5. Enter a locale abbreviation to replace **locale**.
6. Enter a name for the directory of the locale in place of **page_root**.
7. Click **Apply** or **OK**. A warning message is displayed that explains you must publish the updated files to the configuration repository.
8. Click **Publish** to publish the changes right away.

   Otherwise, click **Close** and later, when you are ready to publish click **Domain Management** > **Runtime Node Management** and on the Runtime Node Management panel, click the **Publish pages** button.

Deleting a page locale

You can delete any page locales other than the default C page locale, which was installed when Tivoli Federated Identity Manager was installed.

About this task

Deleting a page locale removes it from the environment and prevents the pages in that locale from being displayed.

Procedure

1. Log on to the management console.
2. Select **Tivoli Federated Identity Manager** > **Domain Management** > **Event Pages**. The event page opens.

3. Click the **Page Locale** tab to open the Page Locale panel.
4. In the **Select** field, select the button next to the page locale you want to remove. For descriptions of the locales, refer to the online help.
5. Click **Delete** and then click **Apply** to apply your changes and remain in the Page Locale portlet, or click **OK** to apply your changes and exit from the portlet.

Customizing multiple-use physical page templates

In certain circumstances, you must customize physical page templates that are referred to by multiple page identifiers.

About this task

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

There are some physical page templates that are referred to by multiple page identifiers in `sps.xml`.

For example, `<sps:PageIdentifierMapping name="/liberty/error_parsing_soap_response.html" location="/liberty/error_parsing_soap_response.html" />` and `<sps:PageIdentifierMapping name="/saml/error_parsing_soap_response.html" location="/liberty/error_parsing_soap_response.html" />`

If the SAML response must differ from the Liberty response, edit the pages as specified in this procedure:

Procedure

1. For each locale affected, copy the Liberty page into the SAML directory.
2. Edit the two pages as desired.
3. Edit the second PageIdentifierMapping in the preceding example to read `<sps:PageIdentifierMapping name="/saml/error_parsing_soap_response.html" location="/saml/error_parsing_soap_response.html" />`
4. Publish these changes as described in "Publishing updates" on page 735.

Customizing the Consent to Federate Page for SAML 2.0

A *consent to federate page* is an HTML form which prompts a user to give consent in joining a federation. You can customize the *consent to federate page* to specify what information it requests from a user.

Before you begin

Determine what values you want to use for the consent to federate page.

About this task

When a user accesses a federation, they agree to join the federation. The HTML form `consent_to_federate.html` prompts for this consent. You can customize what the form requests by adding consent values. These values indicate how a user agrees to join a federation and if service providers are notified of the consent. Identity providers receive the consent values in the SAML 2.0 response.

The following values determine how a user joins a federation:

1 A user agrees to join a federation without notifying the service provider.

0 A user refuses to join a federation.

A URI value
A URI can indicate whether the user agrees to join a federation and if you want to notify the service provider about the user consent. The following table lists and describes the supported URI values.

Table 168. Supported consent values for SAML 2.0 response

Consent value	URI	Description
Unspecified	`urn:oasis:names:tc: SAML:2.0:consent: unspecified`	The consent of the user is not specified.
Obtained	`urn:oasis:names:tc: SAML:2.0:consent: obtained`	Specifies that user consent is acquired by the issuer of the message.
Prior	`urn:oasis:names:tc: SAML:2.0:consent: prior`	Specifies that user consent is acquired by the issuer of the message before the action which initiated the message.
Implicit	`urn:oasis:names:tc: SAML:2.0:consent: current-implicit`	Specifies that user consent is implicitly acquired by the issuer of the message when the message was initiated.
Explicit	`urn:oasis:names:tc: SAML:2.0:consent: current-explicit`	Specifies that the user consent is explicitly acquired by the issuer of the message at the instance that the message was sent.
Unavailable	`urn:oasis:names:tc: SAML:2.0:consent: unavailable`	Specifies that the issuer of the message was not able to get consent from the user.
Inapplicable	`urn:oasis:names:tc: SAML:2.0:consent: inapplicable`	Specifies that the issuer of the message does not need to get or report the user consent.

Follow the steps in this procedure to customize the consent to federate page.

Important: Modify the template files on the Tivoli Federated Identity Manager server. When all of your changes are complete, you can publish them on the WebSphere Application Server configuration repository directory. **Do not** edit these files in the configuration repository.

Procedure

1. Use the `stopServer` command to stop the WebSphere Application Server where Tivoli Federated Identity Manager is installed. For more information, see the WebSphere Information Center.
2. Use a text editor to access `consent_to_federate.html`.

   The template files are in a geographic-specific or language-specific subdirectory. All files are in English. If you installed a language pack, additional locales are available. The default directory depends on the operating system.

   AIX `/usr/IBM/FIM/pages/locale/saml20/`

   Linux or Solaris
   `/opt/IBM/FIM/pages/locale/saml20/`

Windows

```
C:\Program Files\IBM\FIM\pages\locale\saml20\
```

3. Add the appropriate consent values for your federation. See About this task for a complete list of values.
4. Save the files to the appropriate location. This location might be same directory where you edited them.
5. Restart the WebSphere Application Server.

Example

The following example shows an added URI with a consent value `Obtained`:

```
<input type="radio" checked name="Consent"
value="urn:urn:oasis:names:tc:SAML:2.0:consent:obtained"/>
Consent Obtained.br/>
```

In this example, the user consent is acquired by the issuer of the message.

What to do next

Publish the files to the configuration repository. See “Publishing updates” on page 735.

Chapter 51. Developing a custom point of contact server

The point of contact server in your Tivoli Federated Identity Manager environment is the first entity to process a request for access to a resource. You can choose one of the provided options for a point of contact server or you can create a custom point of contact server.

About this task

A custom point of contact server is made up of several customized callback modules that define sign in, sign out, local ID, and authentication.

A custom point of contact server might be appropriate in your environment if you want to integrate an existing authentication or Web access management application with Tivoli Federated Identity Manager.

For example, a custom point of contact server would be useful in the following scenarios:

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

  You could implement a point of contact server that uses a local identity callback (to retrieve the user for the transaction) or implement a custom point of contact server that uses a SignIn callback to assert the user identity to the environment, or implement a point of contact server that uses both types of callbacks.

Developing a custom point of contact server requires programming experience with developing callback modules and knowledge of Tivoli Federated Identity Manager programming concepts. See the developerWorks links 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.

When you have completed the development work, integrate the solution with your Tivoli Federated Identity Manager environment as specified in this procedure:

Procedure

1. Publish the callback plug-ins to the Tivoli Federated Identity Manager runtime module. See “Publishing callback plug-ins” on page 742.
2. Gather the parameter information that you will need for configuring each of the callback modules.
3. Create a new point of contact server profile. You have the option of creating a new profile or using an existing profile as the basis for your new point of contact server profile. See either of the following topics:
   - “Creating a new point of contact server” on page 742
   - “Creating a point of contact server like an existing server” on page 744
4. Activate the point of contact server. See “Activating a point of contact server” on page 746.

Publishing callback plug-ins

If you have developed the modules for a custom point of contact server, you must publish the plug-ins for those modules so that you can use them in your Tivoli Federated Identity Manager environment.

Before you begin

Before continuing with this task, ensure that you have developed the appropriate callback modules for your custom point of contact server. For more information, see Chapter 51, "Developing a custom point of contact server," on page 741.

Procedure

1. Copy the callback plug-ins to the `/plugins` directory where you installed Tivoli Federated Identity Manager. For example, on Windows, the directory is `/opt/IBM/FIM/plugins`.
2. Log 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**.

What to do next

After publishing the plug-ins, you can continue with creating the point of contact profile.

Creating a new point of contact server

Tivoli Federated Identity Manager provides several options for a point of contact server depending on your role in the federation. In addition, you have the option of developing your own point of contact server. If you have developed your own, you must add it to your environment using the console.

Before you begin

Before you can add the custom point of contact server to your environment, you must:

- Publish any custom point of contact callback plug-ins to the runtime node. See "Publishing callback plug-ins."
- Know what type of parameters will be used, if any, and the corresponding values that need to be passed to these callbacks.

About this task

The following procedure describes how to add a custom point of contact server that is unlike any point of contact server already defined in your environment. If you are adding a custom point of contact server that is similar to another point of contact server, use the procedure in "Creating a point of contact server like an existing server" on page 744.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Manage Configuration** > **Point of Contact**.

3. Click **Create**. The Welcome panel of the Point of Contact Profile wizard opens.
4. Ensure that you have completed the prerequisite steps.
5. Click **Next**. The Profile Name panel opens.
6. Type a name for the profile of your custom point of contact server and optionally a description.
7. Click **Next**. The Sign In panel opens.
8. In the Sign In panel, specify the sign-in callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback.
   a. Select a callback in the **Available Callbacks** list. Click **Add** to add it to the **Callbacks in Use** list. Repeat this step to add all the callbacks that you need for the point of contact server.
   b. Click the **Add Parameters** button. A callback parameters section is shown for each callback that is in the Callbacks in Use list. Parameter fields with the default values of `new key` and `new value` are shown.
   c. Add parameters for each callback by changing the default name and value settings to the name and value of the parameters you want to add. To add more parameters, click the **Create** button. When you click **Create**, another parameter field with the default values is added to the parameter list.
   d. Repeat the preceding steps until all parameters have been added to all the callbacks.
9. Click **Next**. The Sign Out panel opens.
10. In the Sign Out panel, specify the sign-out callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback.
    a. Select a callback in the **Available Callbacks** list. Click **Add** to add it to the **Callbacks in Use** list. Repeat this step to add all the callbacks that you need for the point of contact server.
    b. Click the **Add Parameters** button. A callback parameters section is shown for each callback that is in the Callbacks in Use list. Parameter fields with the default values of `new key` and `new value` are shown.
    c. Add parameters for each callback by changing the default name and value settings to the name and value of the parameters you want to add. To add more parameters, click the **Create** button. When you click **Create**, another parameter field with the default values is added to the parameter list.
    d. Repeat the preceding steps until all parameters have been added to all the callbacks.
11. Click **Next**. The Local ID panel opens.
12. In the Local ID panel, specify the local ID callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback.
    a. Select a callback in the **Available Callbacks** list. Click **Add** to add it to the **Callbacks in Use** list. Repeat this step to add all the callbacks that you need for the point of contact server.
    b. Click the **Add Parameters** button. A callback parameters section is shown for each callback that is in the Callbacks in Use list. Parameter fields with the default values of `new key` and `new value` are shown.
    c. Add parameters for each callback by changing the default name and value settings to the name and value of the parameters you want to add. To add more parameters, click the **Create** button. When you click **Create**, another parameter field with the default values is added to the parameter list.
    d. Repeat the preceding steps until all parameters have been added to all the callbacks.

13. Click **Next**. The Authentication panel opens.
14. In the Authentication panel, specify the sign-out callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback.
    a. Select a callback in the **Available Callbacks** list. Click **Add** to add it to the **Callbacks in Use** list. Repeat this step to add all the callbacks that you need for the point of contact server.
    b. Click the **Add Parameters** button. A callback parameters section is shown for each callback that is in the Callbacks in Use list. Parameter fields with the default values of `new key` and `new value` are shown.
    c. Add parameters for each callback by changing the default name and value settings to the name and value of the parameters you want to add. To add more parameters, click the **Create** button. When you click **Create**, another parameter field with the default values is added to the parameter list.
    d. Repeat the preceding steps until all parameters have been added to all the callbacks.
15. Click **Next**. The Summary panel opens. It lists all the callbacks and parameters you specified in the preceding steps.
16. Click **Finish** to complete the setup or click **Back** to return to previous panels and revise your selections.

What to do next

To make this point of contact server active, continue with the instructions in "Activating a point of contact server" on page 746.

Creating a point of contact server like an existing server

Tivoli Federated Identity Manager provides several options for a point of contact server depending on your role in the federation. In addition, you have the option of developing your own point of contact server and basing it on an existing server. If you have developed your own, you must add it to your environment using the console.

Before you begin

Before you can add the custom point of contact server to your environment, you must:

- Publish any custom point of contact callback plug-ins to the runtime node. See "Publishing callback plug-ins" on page 742.
- Know what type of parameters will be used, if any, and the corresponding values to be passed to these callbacks.

About this task

The following procedure describes how to add a custom point of contact server that is like a point of contact server already defined in your environment. If you are adding a custom point of contact server that is not similar to an existing point of contact server, use the procedure in "Creating a new point of contact server" on page 742.

Procedure

1. Log on to the console.

2. Click **Tivoli Federated Identity Manager** > **Manage Configuration** > **Point of Contact**.
3. Select the existing point of contact server which is the basis of your new point of contact server.
4. Click **Create Like**. The Welcome panel of the Point of Contact Profile wizard opens.
5. Ensure that you have completed the prerequisite steps.
6. Click **Next**. The Profile Name panel and the information from the profile that you selected are shown.
7. Type a name for the profile of your custom point of contact server and optionally a description.
8. Click **Next**. The Sign In panel opens.
9. In the Sign In panel, specify the sign-in callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback.

   Because you selected a profile for this point of contact server, the callbacks and parameters for that profile will be shown as the ones in use.

   To add or remove callbacks, use the **Add** and **Remove** buttons. The callbacks in the Callbacks in Use list are the ones that will be used with your new point of contact server.
10. Click **Next**. The Sign Out panel opens.
11. In the Sign Out panel, specify the sign-in callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback.

    Because you selected a profile for this point of contact server, the callbacks and parameters for that profile will be shown as the ones in use.

    To add or remove callbacks, use the **Add** and **Remove** buttons. The callbacks in the Callbacks in Use list are the ones that will be used with your new point of contact server.
12. Click **Next**. The Local ID panel opens.
13. In the Local ID panel, specify the sign-in callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback.

    Because you selected a profile for this point of contact server, the callbacks and parameters for that profile will be shown as the ones in use.

    To add or remove callbacks, use the **Add** and **Remove** buttons. The callbacks in the Callbacks in Use list are the ones that will be used with your new point of contact server.
14. Click **Next**. The Authentication panel opens.
15. In the Authentication panel, specify the sign-in callbacks to use, the order in which the callbacks are used, and the parameters to use with each callback.

    Because you selected a profile for this point of contact server, the callbacks and parameters for that profile will be shown as the ones in use.

    To add or remove callbacks, use the **Add** and **Remove** buttons. The callbacks in the Callbacks in Use list are the ones that will be used with your new point of contact server.
16. Click **Next**. The Summary panel opens. It lists all the callbacks and parameters you specified in the preceding steps.
17. Click **Finish** to complete the setup or click **Back** to return to previous panels and revise your selections.

What to do next

To make this point of contact server active, continue with the instructions in "Activating a point of contact server."

Activating a point of contact server

To use a point of contact server as the active server in your environment, you must activate it.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Manage Configuration** > **Point of Contact**.
3. Select the point of contact server that you want to activate.
4. Click **Make Active**. The point of contact server you selected is activated and used as the point of contact server in your environment.

Chapter 52. Customizing signature X.509 certificate settings

When you sign messages or assertions, the X.509 certificate (public key) is included with your signature as a base64-encoded X.509 certificate. However, you can specify whether this data should be excluded and whether additional data should be included with your signatures.

Before you begin

Before using this procedure, you must have configured your federation. In addition, if you are an identity provider in a SAML 1.x federation, your assertion signature settings are configured when you add your service provider partners. To modify the settings for your assertion signature, you must have already configured a service provider partner.

Procedure

1. Log on to the console.
2. Click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Federations**.

   Or, if you are an identity provider to modify your SAML 1.x assertion signature settings, click **Tivoli Federated Identity Manager** > **Configure Federated Single Sign-on** > **Partners**. The Federations panel shows a list of configured federations.
3. Select a federation. The Partners panel shows a list of configured partners.
4. Select a partner.
5. Click **Properties**.
6. Select the properties to modify. The properties are described in the online help.
7. When you have finished modifying properties, click **OK** to close the Properties panel.

Chapter 53. Running WebSphere Application Server with Java 2

If you are running Java 2 security on the WebSphere Application Server where Tivoli Federated Identity Manager is installed, you must modify the java.policy to grant permission to the Tivoli Federated Identity Manager directories that are in the temp subdirectory of your WebSphere profile.

Procedure

1. Locate the java.policy directory and open it in a text editor. The default locations of the file are:

   AIX

   ```
   /usr/IBM/WebSphere/AppServer/java/jre/lib/security/java.policy
   ```

   Linux or Solaris

   ```
   /opt/IBM/WebSphere/AppServer/java/jre/lib/security/java.policy
   ```

   Windows

   ```
   C:\Program Files\IBM\WebSphere\AppServer
   ```

2. Add the following lines to java.policy:

   ```
   grant codeBase "file:${server.root}/temp/node_name/server_name/
     ITFIMManagementService/-" {permission java.security.AllPermission;
   };
   grant codeBase "file:${server.root}/temp/node_name/server_name/
     ITFIMRuntime/-" {permission java.security.AllPermission;
   };
   ```

   node_name is the name of the node such as IBM-FCFB36CC28ENode05

   server_name is the name of the server such as server1

3. Save and close the `java.policy` file.
4. Restart WebSphere Application Server.

Part 9. Appendixes

Appendix A. tfimcfg reference

Use the **tfimcfg** command to configure WebSEAL or the Web Gateway Appliance as a point of contact server or configure LDAP settings for the alias service.

tfimcfg usage

```
TFIM Autoconfiguration Tool Version 6.2.2.3 [XXXXXXa]

Usage: java -jar tfimcfg.jar [-action <mode>] [options]
The tfimcfg tool has several modes of operation.  Each mode uses different
command line options.

Configuring and unconfiguring WebSEAL servers:
   -action tamconfig: configures a WebSEAL server.  This mode is the default.
   Options:
      -cfgfile <file>: WebSEAL configuration file.
         This option is required.
      -rspfile <file>: response file for non-interactive configuration.
         Default: interactive configuration
      -record: generate response file without making changes to WebSEAL configuration.
      -sslfactory: specify the secure socket connection factory to use,
either TLS or SSL.
         When the TFIM environment is enabled for
FIPS the only supported
factory type is TLS.
         If the parameter is not specified the factory default is SSL.

   -action tamunconfig: unconfigures a WebSEAL server.
   Options:
      -cfgfile <file>: WebSEAL configuration file.
         This option is required.
      -rspfile <file>: response file for non-interactive unconfiguration.
         Default: interactive configuration

Configuring and unconfiguring LDAP servers:
   -action ldapconfig: configures an LDAP server.
   Options:
      -rspfile <file>: response file to control the configuration.  The
         response file should be based on the sample ldapconfig.properties
         file.  This option is required.

   -action ldapunconfig: unconfigures an LDAP server.
   Options:
      -rspfile <file>: response file to control the configuration.  The
         response file should be based on the sample ldapconfig.properties
         file.  This option is required.

Configuring and unconfiguring Web Gateway Appliance servers:
   -action wgaconfig: configures a Web Gateway Appliance server.
   Options:
      -cfgurl <url>: Web Gateway Appliance configuration URL.
         This option is required.
      -rspfile <file>: response file for non-interactive configuration.
         Default: interactive configuration
      -record: generate response file without
making changes to Web Gateway Appliance configuration.
      -sslfactory: specify the secure socket connection factory to use,
either TLS or SSL.
         When the TFIM environment is enabled for FIPS
the only supported factory type is TLS.
         If the parameter is not specified the factory default is SSL.

   -action wgaunconfig: unconfigures a Web Gateway Appliance server.
   Options:
```

```
-cfgfile <file>: Web Gateway Appliance configuration URL.
   This option is required.
-rspfile <file>: response file for non-interactive unconfiguration.
   Default: interactive configuration
```

The log files for the `tfimcfg.jar` tool are written to the system temporary directory. The system temporary file directory is specified by the system property `java.io.tmpdir`.

Configuring WebSEAL or Web Gateway Appliance as point of contact with the tfimcfg tool

Use the tfimcfg tool to configure WebSEAL or Web Gateway Appliance as point of contact.

Before you begin

Make sure that your WebSEAL server is listening for connections on the appropriate IP addresses and port numbers. You can control the IP address and port number by using the WebSEAL configuration file or the Web Gateway Appliance administration console. The IP address is controlled by the [server] network-interface configuration option, and the port numbers are controlled by the [server] `https-port` and [server] `http-port` options.

To use the `tfimcfg` tool, you must meet the following conditions:

- Obtain an IBM® JRE that is supported by the version of PDJrte installed.
- For WebSEAL only: Use PDJrte to configure the IBM JRE in full mode.

For IBM Security Access Manager WebSEAL, version 7.0 or later, you must also meet the following conditions:

- Obtain an IBM JRE, version 6.0, Update 10 or later.
- Configure the *com.ibm.security.cmskeystore.CMSProvider* in the **java.security** file, which is in `$JAVA_HOME/lib/security`, of the IBM JRE.
- Ensure that the `ikeycmd` tool in the `$JAVA_HOME/bin` is on the path.

When you configure WebSEAL as a Tivoli Federated Identity Manager point of contact, you must run the `tfimcfg.jar` tool on the same computer where the WebSEAL instance is configured.

When you configure a Web Gateway Appliance reverse proxy instance as a Tivoli Federated Identity Manager point of contact, you can run the `tfimcfg.jar` tool on the same computer where Tivoli Federated Identity Manager is installed.

About this task

The `tfimcfg` tool uses the host name and port number of your WebSEAL server to determine which federation URLs must be configured on your server. The `tfimcfg` tool tries to determine the host name that is used by your WebSEAL server that is based on several factors:

- The [server] web-host-name WebSEAL configuration setting
- The [server] network-interface WebSEAL configuration setting
- The local hostname of your system
- Host name and IP address resolution

Note: The `tfimcfg` tool might not be able to accurately determine the host name that is used by clients to contact your WebSEAL server, particularly in complex network environments. In any case, the `tfimcfg` tool prompts you to confirm the host name that is used by the WebSEAL server. Enter the correct WebSEAL server name. See "Running the tfimcfg tool" on page 756 for more details.

Depending on the configuration options you select, the tool completes some or all of the following steps:

- Update the WebSEAL key database with the SSL certificate used by the Tivoli Federated Identity Manager server.
- Save a backup of your WebSEAL configuration file.
- Save information in the WebSEAL configuration file necessary to unconfigure this federation from the WebSEAL server later.
- Attach ACLs to grant and restrict access to federation URLs.
- Create a WebSEAL junction to the Tivoli Federated Identity Manager server.
- Configure WebSEAL EAS for Risk-Based Access or OAuth Policy Enforcement Point.
- Configure WebSEAL to send HTTP-Tag-Value pairs to the junction.
- Enable EAI authentication for any endpoints that are used for login, logout, or SOAP.
- Enable BA authentication if you requested BA authentication for your SOAP endpoints.
- Enable certificate authentication if you requested certificate authentication for your SOAP endpoints.
- Disable BA authentication and enable forms authentication if you fulfill the following conditions:
  - It is the first time your WebSEAL server is configured for Tivoli Federated Identity Manager.
  - You did not select BA authentication for SOAP endpoints.
- Delete any ACLs that are and are no longer used.
- Restart your WebSEAL server.
- Save a response file so you can repeat the configuration later.
- Record a log file of all the configuration changes made to your WebSEAL server.

Tivoli Federated Identity Manager provides a `tfimcfg.jar` file that is used to modify the WebSEAL configuration when you use WebSEAL as a Tivoli Federated Identity Manager point of contact.

Procedure

1. Set up a `JAVA_HOME` environment variable for the JRE: For example:

   ```
   export JAVA_HOME=/opt/ibm/java-x86_64-60/jre, or
   export JAVA_HOME=/opt/IBM/WebSphere/AppServer/java/jre
   ```

2. Add `$JAVA_HOME/bin` to the path export `PATH=$JAVA_HOME/bin:$PATH`.
3. From the command line, type:
   - For WebSEAL server:

     ```
     java -jar tfimcfg.jar -action tamconfig -cfgfile WebSEAL_filename
     /opt/pdweb/etc/webseald-default.conf
     ```

   - For Web Gateway Appliance server:

     ```
     java -jar tfimcfg.jar -action wgaconfig -cfgurl Web_Gateway_Appliance_URL
     ```

Results

After the tfimcfg tool completes the basic Tivoli Federated Identity Manager configuration steps on your WebSEAL server, you can customize the WebSEAL configuration.

If you are using multiple WebSEAL replicas for high availability or performance, you can repeat the configuration by using the response file that is generated by the tfimcfg tool. To repeat the configuration, copy the response file to the other WebSEAL server machines, invoke the tfimcfg tool against a different Web Gateway Appliance or select a different Web Gateway Appliance Reverse Proxy instance, and run the configuration:

```
# java -jar tfimcfg.jar -rspfile <path-to-response-file> \
-cfgfile <path-to-webseal-config-file>
```

or

```
# java -jar tfimcfg.jar -action wgaconfig -rspfile <path-to-response-file> \
-cfgurl Web_Gateway_Appliance_URL
```

You must run the tfimcfg tool once for each federation you are configuring on the WebSEAL server. Changes to federation URLs or profiles might require that you rerun the tfimcfg tool. Do not rerun tfimcfg tool when you add more partners to a federation. When you configure multiple federations on a single WebSEAL server, the configuration options are necessary for one federation but might not be suitable for other federations. For example, Identity Provider and Service Provider federations have different requirements.

Running the tfimcfg tool

Use the tfimcfg tool to configure a WebSEAL instance.

Before you begin

The JRE must be configured with the Tivoli Access Manager 6.0 run time for Java. Ensure that the JRE in the current path is the correct one. For example:

```
local:/ # which java
/opt/IBMJava2-142/jre/bin/java
```

About this task

The following assumptions are made about the environment where the tool is being run:

- The example uses an IBM Tivoli Federated Identity Manager 6.1.0 environment, but the process must be nearly identical on subsequent releases of the IBM Tivoli Federated Identity Manager product.
- IBM HTTP Server is configured with the WebSphere web server plug-in to forward requests to a WebSphere server with an IBM Tivoli Federated Identity Manager Identity Provider configured. The IHS server is configured with a host name of **ihs.myidp.com** and is listening on port 80.
- The contact point for the Identity Provider is a WebSEAL server with an instance name of the Identity Provider.
- The contact point for the Identity Provider is not the same as the SOAP endpoint, so a different WebSEAL instance handles the SOAP endpoint URL traffic.

- The IBM Tivoli Federated Identity Manager product is installed in the directory `/opt/IBM/FIMidp`.
- The JVM (IBM Java 2 1.4.2) is installed.
- Tivoli Access Manager is installed and configured, with administrator **sec_master** and password of **passw0rd**.

Procedure

1. Start the utility with the WebSEAL Identity Provider instance configuration file and with the command **–action tamconfig**.

   You must specify the configuration file for the WebSEAL instance to be configured.

   ```
   local:/opt/IBM/FIMidp/tools/tamcfg # java -jar
   ./tfimcfg.jar –action tamconfig –cfgfile
   /opt/pdweb/etc/webseald-idp.conf
   ```

2. Enter the Tivoli Access Manager administrator user ID and password. Since the Tivoli Access Manager administrator user is the default `sec_master` in this example, press **Enter** without specifying the value.

   ```
   TAM administrator user-id [sec_master]: <enter>
   TAM administrator password: passw0rd
   Creating TAM administration context...
   TAM administration context created successfully.
   Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
   ```

3. Enter 1 to continue the processing. The tool prompts for the list of URLs that matches the service endpoint URLs defined for the federation in Tivoli Federated Identity Manager. These URLs are the contact points provided by the WebSEAL server instance. Since the tool correctly identified the WebSEAL host name from the WebSEAL configuration data, press **Enter** without specifying the value.

   ```
   WebSEAL hostname [www.myidp.com]: <enter>
   WebSEAL URLs:
   http://www.myidp.com/
   https://www.myidp.com/
   ```

4. Enter the host name and port of the WebSphere Application Server where the IBM Tivoli Federated Identity Manager runtime application is installed and running. WebSphere Application Server must be active when the `tfimcfg` tool is run. In this example, SSL is not used to communicate with the WebSphere Application Server.

   ```
   Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
   ITFIM hostname []: ihs.myidp.com
   ITFIM HTTP port: 80
   Use SSL connection to ITFIM server (y/n): n
   Testing connection to
   http://ihs.myidp.com:80/Info/InfoServiceXML.
   Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
   ```

5. Select the federation to be configured. In this example, there is only one SAML 1.1 federation, `saml11Fed`, which is created.

   ```
   Federation to configure:
   1. saml11Fed
   2. Cancel
   Enter your choice [1]: 1
   ```

6. Enter 1 to continue with the configuration.

   ```
   <p>The following endpoints will be configured on this WebSEAL</p><p>server:
   </p><p>https://www.myidp.com/FIM/sps/saml11Fed/saml11/login</
   p><p>Press 1 for Next, 2 for Previous,
   3 to Repeat, C to Cancel: <b>1</b></p>
   ```

7. Enter 1 to accept the value. The tool displays the URL that is chosen for all authenticated users access because it is recognized as the Intersite Transfer Service endpoint. This endpoint is the URL that generates a token and transfer the user identity to another site.

```
<p>URLs allowing all authenticated users access:</p><p>
https://www.myidp.com/FIM/sps/saml11Fed/saml11/login</
p><p>Press 1 for Next, 2 for Previous,
3 to Repeat, C to Cancel: <b>1</b></p>
```

8. Enter 1 to continue to this summary page:

```
Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
---------------------------------------------
Planned configuration steps:
A backup of the WebSEAL configuration will be saved as
/opt/pdweb/etc/webseald-idp.conf.2006-03-02-17-08-49.
A junction to the FIM server will be created at /FIM.
ACLs denying access to all users will be attached to:
/WebSEAL/www.myidp.com-idp/FIM
ACLs allowing access to all authenticated users will be
attached to:
/WebSEAL/www.myidp.com-idp/FIM/sps/saml11Fed/saml11/login
/WebSEAL/www.myidp.com-idp/FIM/fimivt
HTTP-Tag-Value header insertion will be configured for the
attributes:
ssn=ssn
name=name
email=email
user_session_id=user_session_id
```

9. Enter 1 to continue the configuration changes.

```
Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
Beginning configuration...
Configuration backup:/opt/pdweb/etc/webseald-idp.conf.2006-03-
02-17-15-54
Attaching ACLs.
Creating ACL itfim_saml11Fed_nobody.
Creating ACL itfim_saml11Fed_anyauth.
Creating junction /FIM.
Created junction at /FIM
Editing configuration file...
Restarting the WebSEAL server...
Configuration complete.
```

Results

The tool creates a response file for the repeated application for this configuration step to other WebSEAL servers. Finally, it suggests some possible next steps.

Configuring the SOAP traffic with the tfimcfg tool

Use the tfimcfg tool to configure the WebSEAL server that handles the SOAP traffic.

About this task

In this example, the Identity Provider is using transport security to restrict access to its SOAP endpoint, which is a dedicated WebSEAL instance that is configured as `idpsoap`. This method allows client-side certificate authentication to be mandatory for this WebSEAL instance without affecting the rest of the Tivoli Federated Identity Manager environment.

The Identity Provider SOAP endpoint or Artifact Resolution Service requires that the client must authenticate by using a certificate that specifies that the user is a

member of the Tivoli Access Manager group soapusers.

Procedure

1. Start the Tivoli Federated Identity Manager Configuration Utility for Tivoli Access Manager.

   ```
   /opt/IBM/FIMidp/tools/tamcfg # java -jar ./tfimcfg.jar -action
   tamconfig -cfgfile /opt/pdweb/etc/webseald-idpsoap.conf
   ```

2. Enter the Tivoli Access Manager administrator data and WebSEAL host name.

   ```
   TAM administrator user-id [sec_master]: <enter>
   TAM administrator password: passw0rd
   Creating TAM administration context...
   TAM administration context created successfully.
   Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
   WebSEAL hostname [soap.myidp.com]: <enter>
   WebSEAL URLs:
   https://soap.myidp.com/
   ```

3. Enter the IBM Tivoli Federated Identity Manager host name.

   The host name is the WebSphere Application Server where Tivoli Federated Identity Manager is running. Specify the federation that is being configured.

   ```
   Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
   ITFIM hostname []: ihs.myidp.com
   ITFIM HTTP port: 80
   Use SSL connection to ITFIM server (y/n): n
   Testing connection to
   http://ihs.myidp.com:80/Info/InfoServiceXML.
   Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
   Federation to configure:
   1. saml11Fed
   2. Cancel
   Enter your choice [1]: 1
   The following endpoints will be configured on this WebSEAL
   server:
   https://soap.myidp.com/FIM/sps/saml11Fed/saml11/soap
   ```

   For example, **saml11Fed**. The tool then indicates the list of URLs that are identified as related to the federation. This list is the list of service endpoint URLs that are recognized as related to this WebSEAL instance.

4. Enter 1 to select **Certificate authentication**.

5. Enter **soapusers**. Because the matched endpoint is Artifact Resolution Service, the tool requests the type of authentication that must be configured for authenticated SOAP clients. This Identity Provider is configured with a policy that only clients with valid certificates whose users are members of Tivoli Access Manager group soapusers are allowed to access Tivoli Federated Identity Manager SOAP endpoint.

   ```
   Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
   Access type for endpoint URL
   https://soap.myidp.com/FIM/sps/saml11Fed/saml11/soap:
   1. Certificate authentication
   2. User-id/password authentication
   3. Unauthenticated access
   Enter your choice [1]: 1
   Group for SOAP access: soapusers
   URLs used for authenticated SOAP clients:
   https://soap.myidp.com/FIM/sps/saml11Fed/saml11/soap
   Authentication type: certificate
   Access group: soapusers
   ```

6. Enter n to the request for IVT configuration.

Because IVT configuration is already done for the Identity Provider during configuration of `www.myidp.com`, or the WebSEAL instance Identity Provider, it is not required again.

The summary shows the following details.

- A backup copy of the WebSEAL configuration file is created.
- A junction that is called **/FIM** is created between WebSEAL and the host or port that contacts the Tivoli Federated Identity Manager run time. The name **/FIM** is taken from the URL of the relevant service endpoint.
- A **deny all** ACL is attached to **/FIM**. This security policy ensures that any access that is not explicitly allowed is denied.
- A **allow group soapusers only** ACL is attached to the Artifact Resolution Service endpoint.

```
Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
-----------------------------------------------
Planned configuration steps:
A backup of the WebSEAL configuration will be saved as
/opt/pdweb/etc/webseald-idpsoap.conf.2006-03-02-19-52-54.
A junction to the FIM server will be created at /FIM.
ACLs denying access to all users will be attached to:
/WebSEAL/soap.myidp.com-idpsoap/FIM
ACLs allowing access to the group soapusers will be attached to:
/WebSEAL/soap.myidp.com-idpsoap/FIM/sps/saml11Fed/saml11/soap
Certificate authentication will be enabled.
HTTP-Tag-Value header insertion will be configured for the
attributes:
user_session_id=user_session_id
```

7. Enter **1** to continue the configuration changes and the following information is displayed:

```
Configuration backup:/opt/pdweb/etc/websealdidpsoap.
conf.2006-03-02-19-52-54
Attaching ACLs.
Creating ACL itfim_saml11Fed_soapusers.
Creating junction /FIM.
Created junction at /FIM
Editing configuration file...
Restarting the WebSEAL server...
Configuration complete.
```

Setting up a soapusers group and certificate

Set up a client-side certificate that identifies the user as a member of a Security Access Manager group. Set up the client so that this group is allowed access according to Tivoli Access Manager policy.

About this task

This task provides instructions for setting up both the Tivoli Access Manager user and group and the certificate for the client side authentication. With the SOAP over HTTP bindings, the Identity Provider authenticates the soap client with any following options:

- Basic authentication
- Client-side certificate

Procedure

1. Create an entry for the suffix for the user and group in LDAP.
   - The suffix `o=ibm, c=us` is used, so you must create an appropriate LDIF file.
   - The LDIF file and the **`idsldapupdate`** command to create the suffix.

- The LDAP administrator is `cn=root` with password `passw0rd`.
- The LDAP server is Tivoli Directory Server.

```
# cat ibmorg.ldif
dn: o=ibm,c=us
changetype: add
objectclass: organization
o: ibm
# idsldapmodify -D cn=root -w passw0rd –f /studentfiles/files/ibmOrg.ldif
```

2. Create a Tivoli Access Manager user `spsoapuser` and group `soapusers`.

```
# pdadmin -a sec_master -p passw0rd <<SOAPUSER
user create spsoapuser cn=spsoapuser,o=ibm,c=us spsoapuser mssoap passw0rd
user modify spsoapuser account-valid yes
group create soapusers cn=soapusers,o=ibm,c=us soapusers
group modify soapusers add spsoapuser
SOAPUSER
```

3. Create or obtain a certificate for use by a Service Provider for client authentication to the SOAP endpoint. The public key of the service provider must be imported to the appropriate keystore by using the IBM Tivoli Federated Identity Manager key service.

   The iKeyman tool that is shipped with Tivoli Access Manager and WebSphere can be used to create a self-signed certificate.

   The certificate must specify that the subject DN is the Tivoli Access Manager user that is created or `cn=spsoapuser, o=ibm, c=us` in the example.
4. Configure the new WebSEAL instance with a `.kdb` file that validates a client certificate that is used by the Service Provider.

tfimcfg limitation with Sun Java 1.4.2.4

Certain versions of Sun Java are incompatible with tfimcfg.

The incompatibility causes the following error:

```
HPDAZ0602E Corrupted file: Insufficient information to contact Policy Server
```

The problem occurs because the Sun JRE is unable to read the keystores generated by the Tivoli Access Manager PDJrteCfg. When this error occurs, you must either use an IBM JVM, or apply the latest JRE patches from Sun. If the problem persists after applying the patches from Sun, use an IBM JVM for the configuration.

tfimcfg LDAP properties reference

The tfimcfg utility reads a properties file to obtain the values to use when configuring an LDAP user registry. The properties file contains values that you can modify.

`ldap.hostname`
The LDAP server host name. Default: `localhost`

`ldap.port`
The LDAP port number. Default: `389`

The default value is for non-SSL communication. When you have configured the LDAP server to communicate using SSL, the default port is 636.

`ldap.suffix.add`
Boolean value that specifies whether tfimcfg adds suffixes to the LDAP server as needed. Supports IBM Tivoli Directory Server Versions 6.1, 6.0 and 5.2 only.

Default:

```
ldap.suffix.add=true
```

ldap.suffix.user.configuration
ldap.organization.configuration

Boolean values that specify whether tfimcfg creates LDAP containers to store Tivoli Federated Identity Manager users and groups. The Tivoli Federated Identity Manager users and groups are:

- Tivoli Federated Identity Manager server users and groups
- Tivoli Federated Identity Manager Installation Verification Tool (IVT) users and groups

When you do not need those users and groups, or you already have LDAP containers that to use for those users and groups, set these values to `false`.

When ldap.organization.configuration is true, tfimcfg creates the `dc=example,dc=com` LDAP objects.

Default:

```
ldap.suffix.user.configuration=true
ldap.organization.configuration=true
```

ldap.suffix.alias.configuration

Boolean value that specifies whether tfimcfg creates an LDAP suffix to store single sign-on aliases. The default alias is `cn=itfim`.

```
ldap.suffix.alias.configuration=true
```

ldap.suffix.tam.configuration

Boolean value that specifies whether tfimcfg creates the `secAuthority=Default` suffix for Tivoli Access Manager.

- When you have already configured Tivoli Access Manager set this value to `false`.
- When Tivoli Access Manager is not using this LDAP server, set this value to `false`.

```
ldap.suffix.tam.configuration=true
```

Note: If the secAuthority=Default suffix exists, the tfimcfg program ignores the value of the ldap.suffix.tam.configuration property.

ldap.fim.configuration

Boolean value that specifies whether tfimcfg configures LDAP for the Tivoli Federated Identity Manager alias service.

Default value: `true`.

ldap.ivt.sp.configuration

Boolean value that specifies whether tfimcfg creates users and groups for the service provider in the Installation Verification Tool (IVT) application.

Default value: `true`.

ldap.ivt.ip.configuration

Boolean value that specifies whether tfimcfg creates users and groups for the identity provider in the Installation Verification Tool (IVT) application.

Default value: `true`.

ldap.modify.acls

Boolean value that specifies whether tfimcfg attaches appropriate ACLs (access control lists) to the LDAP server. These ACLs grant read and write access to the Tivoli Federated Identity Manager administrative users created by tfimcfg.

Note: tfimcfg attaches ACLs for IBM LDAP and Sun ONE servers. For other LDAP servers, you must attach the ACLs manually.

When the value is set to `false`, you must attach the ACLs manually.

Default value: `true`.

ldap.admin.dn
The DN used by the LDAP administrator to issue bind requests.

Default: `cn=root`

ldap.admin.password
The password for the LDAP administrator.

Default: `passw0rd`

ldap.security.enabled
Boolean value that specifies whether communication with the LDAP server must use SSL.

Default: `false`.

ldap.security.trusted.jks.filename
The name of the Java keystore that contains the signer of the LDAP-presented SSL certificate that LDAP presents during trusted communications.

ldap.suffix.user.dn
ldap.suffix.user.name
ldap.suffix.user.attributes
ldap.suffix.user.objectclasses
When you want tfimcfg.jar to create LDAP containers for your users, you can set these values to control the Distinguished Names (DNs) that are used.

Defaults:

```
ldap.suffix.user.dn=dc=com
ldap.suffix.user.name=com
ldap.suffix.user.attributes=dc
ldap.suffix.user.objectclasses=domain
```

ldap.suffix.alias.dn
Distinguished Name (DN) to use for storing single sign-on alias. This value of this property must begin with `cn=`. Modify this value when you do not want to use the default DN.

Default:

```
ldap.suffix.alias.dn=cn=itfim
```

ldap.organization.dn
ldap.organization.name
ldap.organization.attributes
ldap.organization.objectclasses
When you want tfimcfg.jar to create LDAP containers for your groups, you can set these values to control the Distinguished Names (DNs) that are used.

Defaults:

```
ldap.organization.dn=dc=example,dc=com
ldap.organization.name=example
ldap.organization.attributes=dc
ldap.organization.objectclasses=domain
```

ldap.user.container.dn
ldap.group.container.dn

The distinguished names to use for the containers for users and groups.

Defaults:

```
ldap.user.container.dn=cn=users,dc=example,dc=com
ldap.group.container.dn=cn=groups,dc=example,dc=com
```

ldap.fim.server.bind.dn
ldap.fim.server.bind.shortname
ldap.fim.server.bind.password

The distinguished name, short name, and password that the Tivoli Federated Identity Manager server (application) uses to bind to the LDAP server.

Default:

```
ldap.fim.server.bind.dn=uid=fimserver,cn=users,dc=example,dc=com
ldap.fim.server.bind.shortname=fimserver
ldap.fim.server.bind.password=passw0rd
```

ldap.fim.admin.group.dn
ldap.fim.admin.group.shortname

The distinguished name and short name for the Integrated Solutions Console administration group.

Default:

```
ldap.fim.admin.group.dn=cn=fimadmins,cn=groups,dc=example,dc=com
ldap.fim.admin.group.shortname=fimadmins
```

ldap.user.objectclasses
ldap.group.objectclasses
ldap.user.shortname.attributes

The values for LDAP containers for user objectclass, group objectclass, and user shortname attributes.

Default:

```
ldap.user.objectclasses=person,organizationalPerson,inetOrgPerson
ldap.group.objectclasses=groupOfUniqueNames
ldap.user.shortname.attributes=cn,sn,uid
```

Default ldapconfig.properties file

The ldapconfig.properties file is distributed as part of the runtime and management service component. Some of the properties have default values.

```
ldap.hostname=localhost
ldap.port=389

# If true, new suffixes will be added to the LDAP server as needed.
# Only supported for IDS 5.2 and 6.0
ldap.suffix.add=true

# If true, data for the LDAP user suffix (dc=com, by default) will be
# created.
ldap.suffix.user.configuration=true

# If true, data for the SSO alias suffix (cn=itfim, by default) will be
# created.
ldap.suffix.alias.configuration=true

# If true, create the secAuthority=Default suffix for TAM
ldap.suffix.tam.configuration=true
ldap.fim.configuration=true
ldap.ivt.sp.configuration=true
ldap.ivt.ip.configuration=true
ldap.organization.configuration=true
ldap.modify.acls=true

ldap.admin.dn=cn=root
ldap.admin.password=passw0rd

ldap.security.enabled=false
ldap.security.trusted.jks.filename=

ldap.suffix.user.dn=dc=com
ldap.suffix.user.name=com
ldap.suffix.user.attributes=dc
ldap.suffix.user.objectclasses=domain

# DN to use for storing SSO aliases.  This must begin with cn=
ldap.suffix.alias.dn=cn=itfim

ldap.organization.dn=dc=example,dc=com
ldap.organization.name=example
ldap.organization.attributes=dc
ldap.organization.objectclasses=domain

ldap.user.container.dn=cn=users,dc=example,dc=com
ldap.group.container.dn=cn=groups,dc=example,dc=com

ldap.fim.server.bind.dn=uid=fimserver,cn=users,dc=example,dc=com
ldap.fim.server.bind.shortname=fimserver
ldap.fim.server.bind.password=passw0rd

ldap.fim.admin.group.dn=cn=fimadmins,cn=groups,dc=example,dc=com
ldap.fim.admin.group.shortname=fimadmins

ldap.user.objectclasses=person,organizationalPerson,inetOrgPerson
ldap.group.objectclasses=groupOfUniqueNames
ldap.user.shortname.attributes=cn,sn,uid
```

Figure 79. Default values for ldapconfig.properties

Sample output from tfimcfg configuration of LDAP

The following section shows a sample output from the running of tfimcfg.

The command for running tfimcfg to configure LDAP entries for the alias service is:

```
java -jar tfimcfg.jar -action ldapconfig -rspfile /tmp/ldapconfig.properties
```

The following figure is an output from running the command on an identity provider. The example uses an `ldapconfig.properties` file that has the default values.

```
Configuring LDAP server.
LDAP server vendor: International Business Machines (IBM),
   version 6.0.
Adding LDAP suffix secAuthority=Default.
Reloading IBM Directory Server configuration.
Adding LDAP suffix dc=com.
Reloading IBM Directory Server configuration.
Creating LDAP object dc=com.
Adding LDAP suffix cn=itfim-cmd.
Reloading IBM Directory Server configuration.
Creating LDAP object cn=itfim-cmd.
Creating LDAP object dc=example,dc=com.
Creating LDAP object cn=users,dc=example,dc=com.
Creating LDAP object cn=groups,dc=example,dc=com.
Creating LDAP object uid=fimserver,cn=users,dc=example,dc=com.
Creating LDAP object cn=fimadmins,cn=groups,dc=example,dc=com.
Adding user uid=fimserver,cn=users,dc=example,dc=com to group
   cn=fimadmins,cn=groups,dc=example,dc=com.
Creating LDAP object o=identityprovider,dc=com.
Creating LDAP object cn=MEemployee,o=identityprovider,dc=com.
Creating LDAP object cn=MEmanager,o=identityprovider,dc=com.
Creating LDAP object cn=MEexecutive,o=identityprovider,dc=com.
Creating LDAP object cn=elain,o=identityprovider,dc=com.
Creating LDAP object cn=mary,o=identityprovider,dc=com.
Creating LDAP object cn=chris,o=identityprovider,dc=com.
Updating IBM LDAP ACLs for suffix CN=ITFIM-CMD.
Updating IBM LDAP ACLs for suffix SECAUTHORITY=DEFAULT.
Updating IBM LDAP ACLs for suffix DC=COM.
Done updating LDAP server configuration.
```

Figure 80. Sample output from tfimcfg.jar

Appendix B. URLs for initiating SAML single sign-on actions

The SAML specifications provide limited or no guidance about the endpoints or methods that users must use to initiate single sign-on actions. However, in a Tivoli Federated Identity Manager environment, URLs are defined that user can use to initiate single sign-on actions.

The following reference is useful for architects or application developers who are implementing the user interaction components of their federation.

Note: These URLs are not used for partner-to-partner communication. For more information, see Chapter 16, "SAML endpoints and URLs," on page 173.

SAML 1.x initial URL

The intersite transfer service URL is where the sign-on request process begins in a SAML 1.x federation. The URL for initiating a single sign-on request has the following syntax:

Syntax

```
https://identity_provider_hostname:port_number/sps/
  federation_name/samlxx/login?TARGET=
  service_provider_id/target_application_location
  [optional query strings]
```

Elements

https *or* **http**
The URI scheme. `https` for resources that are protected by secure sockets layer (SSL). `http` for resources that are not protected by SSL.

identity_provider_hostname
The host name of the point of contact server of the identity provider.

port_number
The port number of the intersite transfer service endpoint. The default value is 9443.

sps The designation for the Tivoli Federated Identity Manager Server. This element cannot be changed.

federation_name
The name you assign to the federation when you create it.

saml*xx*
The designation of the SAML protocol you choose to use in your federation. The values can be one of the following:

- saml (for SAML 1.0)
- saml11 (for SAML 1.1)

login This element indicates what type of endpoint is using the port. **login** is used for the intersite transfer service.

Use the **TARGET** query string. You have the option of using either, both, or neither of the optional query strings (**SP_PROVIDER**) and (**PROTOCOL**), see the following examples:

TARGET
The URL of the target application that a user can log on to using single sign-on.

SP_PROVIDER_ID
The value of query string specifies the provider ID of the service provider that is the target of the single sign-on request. This query string is optional but might be necessary. The use of this query string removes any ambiguity about which service provider is the target of the single sign-on request.

Without this query string, the service provider is determined by matching the *URI://hostname[:port]* of the URL in the TARGET query string to the *URI://hostname[:port]* of the provider ID for the service provider partner that is configured for the federation. This parameter is used with requests that are initiated at the identity provider.

PROTOCOL
The value of this parameter specifies the type of single sign-on profile (browser artifact or browser POST) that can be used for the single sign-on request. The syntax of the extension is `PROTOCOL=[BA|POST]`, with BA indicating Browser Artifact and POST indicating Browser POST. The query string overrides local identity provider configuration.

The use of the extension is optional. When the extension is not present, the profile choice is determined by the configuration file settings. To use this extension, you must enable the IBM PROTOCOL extension setting during the configuration steps for creating a SAML 1.x federation on an identity provider.

These query strings can be used individually or in combination. For example, the URL used to initiate single sign-on, when the SP_PROVIDER_ID is used but the PROTOCOL extension is not, has the following syntax:

```
https://intersite_transfer_service_URL?SP_PROVIDER_ID=
  provider_ID_of_service_provider&TARGET=target_application_URL
```

With the SP_PROVIDER_ID and the PROTOCOL extension, the URL has the following syntax:

```
https://intersite_transfer_service_URL?SP_PROVIDER_ID=
  provider_ID_of_service_provider&TARGET=target_application_URL
  &PROTOCOL=[BA|POST]
```

Examples

Single sign-on URL, without the optional parameters:
The following example shows the single sign-on URL for an identity provider using a federation named `ipfed`, the SAML 1.1 protocol, a service provider with a provider ID of `https://sp.example.com:9443`, and an application called `snoop`:

```
https://idp.example.com:9443/sps/ipfed/saml11/login?TARGET=
  https://sp.example.com:9443/snoop/
```

Single sign-on URL, when SP_PROVIDER_ID and PROTOCOL extension *are* used:

The following example shows a URL that is used to initiate single sign-on when the IBM PROTOCOL extension *is* used. In this example, even if the identity provider is configured to use a POST profile for the service provider named `sp`, the following use of the PROTOCOL extension would force the identity provider to use the browser artifact profile:

```
https://idp.example.com:9443/sps/ipfed/saml11/login?SP_PROVIDER_ID=
  https://sp.example.com:9443/sps/spfed/saml11&TARGET=
  https://sp.example.com:9443/
  snoop&PROTOCOL=BA
```

Single sign-on URL, when SP_PROVIDER_ID is used but the PROTOCOL extension is *not* used:

The following example shows a URL that is used to initiate single sign-on when the SP_PROVIDER_ID is used but the IBM PROTOCOL extension is *not* used:

```
https://idp.example.com:9443/sps/ipfed/saml11/login?SP_PROVIDER_ID=
  https://sp.example.com:9443/sps/spfed/saml11&TARGET=
  https://sp.example.com:9443/snoop
```

SAML 2.0 profile initial URLs

In the Tivoli Federated Identity Manager environment, specially formed URLs can be used for user-initiated single sign-on actions. These URLs incorporate the single sign-on action to take, the binding to be used for the action, and the location where the action takes place. These URLs are called *profile initial URLs*.

The SAML 2.0 specification defines the endpoints that are to be used for partner-to-partner communications. But, the specification does not define the way in which users can initiate a single sign-on action with those endpoints.

Architects and application developers, who design and implement the interaction of their users with the single sign-on process, must understand profile initial URLs and incorporate them into their web applications.

The following sections describe the format of the SAML 2.0 profile initial URLs that are supported in a Tivoli Federated Identity Manager environment.

Assertion consumer service initial URL (service provider)

In a SAML 2.0 federation, the assertion consumer service URL can be initiated at the identity provider server site or the service provider site. This topic describes the syntax for initiating single sign-on at the service provider.

Syntax for initiating single sign-on at the service provider

```
https://provider_hostname:port_number/sps/
federation_name/saml20/logininitial?
RequestBinding=RequestBindingType&
ResponseBinding=ResponseBindingType&
NameIdFormat=NameIDFormatType&
IsPassive=[true|false]&
ForceAuthn=[true|false]&
AllowCreate=[true|false]&
AuthnContextClassRef = ClassReference&
AuthnContextDeclRef = DeclarationReference&
AuthnContextComparison = [exact| minimum | maximum |better]&
Target=target_application_location
```

Elements

https *or* **http**

The URI scheme. `https` for resources that are protected by secure sockets layer (SSL). `http` for resources that are not protected by SSL.

provider_hostname

The host name of the provider point of contact server.

port_number
The port number of the intersite transfer service endpoint. The default value is 9443.

sps The designation for the Tivoli Federated Identity Manager Server. This element cannot be changed.

federation_name
The name you assign to the federation when you create it.

saml20
The designation of SAML 2.0.

logininitial
This element indicates what type of endpoint is using the port. **logininital** is used to initiate the single sign-on service.

The following query strings must also be used in the URL:

RequestBinding
The binding that is used to send the request. The valid values when initiating single sign-on at the service provider are:

- HTTPPost
- HTTPArtifact
- HTTPRedirect

ResponseBinding
The binding that is used by the responder to return the response. The valid values when initiating single sign-on at the service provider are:

- HTTPPost
- HTTPArtifact

Target The URL of the application that a user can log on to using single sign-on.

NameIdFormat
The name ID format that is to be used for name identifiers. Valid values are:

- Transient (anonymous)
- Persistent
- Encrypted (for encrypted name IDs)
- Email

Persistent is the default setting. If the NameIdFormat attribute is not included, a persistent name ID is used.

AllowCreate
Indicates if new persistent account linkage is performed on the request. The default value is `true`.

Note: To use this parameter, the **NameIdFormat** must be set to Persistent.

ForceAuthn
Specifies whether the identity provider authenticates the user. A value of `true` means that the user must be authenticated. The default value is `false`.

Note:

- Depending on the federation configuration, the more restrictive setting is implemented. For example, if you set the federation configuration to force a user to authenticate, setting the ForceAuthn element to `false` is not implemented.
- If you plan to use WebSEAL cookie management with SAML 2.0 ForceAuthn, ensure that the list of managed cookies does not include the WebSphere session cookie. See "Configuring WebSEAL to manage cookies" on page 532.

IsPassive

Indicates if the identity provider must take control of the user agent if set to `true`. The identity provider is not permitted to request the user to provide login credentials.

The default value is `false`.

Note: Depending on the federation configuration, the more restrictive setting is implemented. For example, if you set the federation configuration to prevent the identity provider from taking control of the user agent, setting the IsPassive element to `false` is not implemented.

AuthnContextClassRef

Specifies one or more string values which identify authentication context class URI references.

Note: Use either `AuthnContextClassRef` or `AuthnContextDeclRef`. If both are supplied, `AuthnContextClassRef` is used.

AuthnContextDeclRef

Specifies one or more string values which identify authentication context declaration URI references.

Note: Use either `AuthnContextClassRef` or `AuthnContextDeclRef`. If both are supplied, `AuthnContextClassRef` is used.

AuthnContextComparison

Specifies the type of comparison used to determine the requested context classes or declarations. The comparison type must be one of the following variables:

- exact
- minimum
- maximum
- better

The default value is exact.

AttributeConsumerSvcIndex

Specifies the index of the set of attributes to return. This attribute does not correspond to any configuration. Administrators can use `AttributeConsumerSvcIndex` to select which user identity attributes to include in the user token during the identity mapping phase.

This attribute is supported on both the Identity Provider and Service Provider.

AssertionConsumerSvcIndex

Specifies the index of the Assertion Consumer Service URL where the Identity Provider sends the response. The value must correspond to the endpoint in the Service Provider metadata.

This attribute is supported on both the Identity Provider and Service Provider.

Note: In case `ResponseBinding` and `AssertionConsumerSvcIndex` are specified, the latter takes precedence.

Example

Single sign-on URL when initiated at service provider:
The following example shows the single sign-on URL when initiated at a service provider. The name of the federation is `spfed`, and uses the SAML 2.0 protocol, HTTPPost as the request binding and response binding, and a target application at `https://sp.example.com:9443/banking`:

```
https://sp.example.com:9443/sps/
spfed/saml20/logininitial?
RequestBinding=HTTPPost&
ResponseBinding=HTTPPost&
NameIdFormat=persistent&
IsPassive=true&
ForceAuthn=true&
AllowCreate=true&
RequestedAuthnContextComparison=minimum&
AuthnContextClassRef=classref1&
AttributeConsumerSvcIndex=1&
Target=https://sp.example.com:9443/banking
```

Single sign-on service initial URL (identity provider)

In a SAML 2.0 federation, the single sign-on service URL can be initiated at the identity provider server site or the service provider site. This topic describes the syntax for initiating the service at the identity provider.

Syntax for initiating single sign-on at the identity provider

```
https://provider_hostname:port_number/sps/
federation_name/saml20/logininitial?RequestBinding=RequestBindingType&
PartnerId=target_partner_provider_ID
&NameIdFormat=NameIDFormatType&AllowCreate=[true|false]
&Target=target_application_location
```

Elements

https *or* **http**
The URI scheme. Use `https` for resources that are protected by secure sockets layer (SSL). Use `http` for resources that are not protected by SSL.

provider_hostname
The point of contact server host name of the provider.

port_number
The port number of the inter-site transfer service endpoint. The default value is 9443.

sps
The designation for the Tivoli Federated Identity Manager Server. This element cannot be changed.

federation_name
The name you assign to the federation when you create it.

saml20
The designation of SAML 2.0.

logininitial
This element indicates what type of endpoint is using the port. **logininital** is used to initiate the single sign-on service.

Target This element is URL-encoded and set as the value of the **RelayState** parameter in the unsolicited response delivered by the identity provider to the service provider. A Tivoli Federated Identity Manager Service Provider interprets this value as the URL of the application that a user can log on to using single sign-on.

The URL must also contain the following query strings:

RequestBinding
The binding that is used to send the response to the service provider. The valid values when initiating single sign-on at the identity provider are:

- HTTPPost
- HTTPArtifact

PartnerId
The provider ID of the target partner.

NameIdFormat
The name ID format that is to be used for name identifiers. Valid values are:

- Transient (anonymous)
- Persistent
- Encrypted (for encrypted name IDs)
- Email

Persistent is the default setting. If the NameIdFormat attribute is not included, a persistent name ID is used.

AllowCreate
Indicates whether to do a new persistent account linkage upon request. The default value is False.

Note: You must set **NameIdFormat** to Persistent to use this parameter.

AttributeConsumerSvcIndex
Specifies the index of the set of attributes to return. This attribute does not correspond to any configuration. Administrators can use `AttributeConsumerSvcIndex` to select which user identity attributes to include in the user token during the identity mapping phase.

This attribute is supported on both the Identity Provider and Service Provider.

AssertionConsumerSvcIndex
Specifies the index of the Assertion Consumer Service URL where the Identity Provider sends the response. The value must correspond to the endpoint in the Service Provider metadata.

This attribute is supported on both the Identity Provider and Service Provider.

Note: In case `ResponseBinding` and `AssertionConsumerSvcIndex` are specified, the latter takes precedence.

Example

Single sign-on URL when initiated at identity provider:

The following example shows the single sign-on URL when initiated at an identity provider, using the SAML 2.0 protocol. `AssertionConsumerSvcIndex` refers to the index of the ACS URL to send the response. `AttributeConsumerServiceIndex` refers to the index or set of attributes to return.

```
https://ip/FIM/sps/
saml20/saml20/logininitial?
RequestBinding=HTTPArtifact&
NameIdFormat=persistent&
AllowCreate=true&
AssertionConsumerSvcIndex=0&
AttributeConsumerSvcIndex=1&
PartnerId=https://sp/FIM/sps/saml20/saml20&
Target=https://sp.example.com:9443/banking
```

Single logout service initial URL

In a SAML 2.0 federation , the single logout service URL is used by a partner to contact the Single logout profile. The URL to initiate the service has the following syntax:

Syntax

```
https://provider_hostname:port_number/sps/
  federation_name/saml20/sloinitial
..?RequestBinding=RequestBindingType
```

Elements

https *or* **http**

The URI scheme. `https` for resources that are protected by secure sockets layer (SSL). `http` for resources that are not protected by SSL.

provider_hostname

The host name of the point of contact server for the service or identity provider.

port_number

The port number of the artifact resolution service endpoint. The default value is 9444.

sps

The designation for the Tivoli Federated Identity Manager server. This element cannot be changed.

federation_name

The name you assign to the federation when you create it.

saml20

The designation that SAML 2.0 is used in your federation.

sloinitial

This element indicates what type of endpoint is using the port. **sloinitial**is used to initiate the single logout service

The following query must also be included:

RequestBinding

The binding that is used to send the request. The valid values are:

- HTTPPost
- HTTPRedirect

- HTTPArtifact
- HTTPSOAP

Examples

Single logout URL when initiated at service provider:

The following example shows the single logout URL when initiated at a service provider in a federation named `spfed`, using the SAML 2.0 protocol, HTTPRedirect as the request binding:

```
https://sp.example.com:9443/sps/spfed/saml20/sloinitial?
  RequestBinding=HTTPRedirect
```

Single logout URL when initiated at identity provider:

The following example shows the single logout URL when initiated at an identity provider in a federation named `ipfed`, using the SAML 2.0 protocol, HTTPArtifact as the request binding:

```
https://idp.example.com:9444/sps/ipfed/saml20/sloinitial?
  RequestBinding=HTTPArtifact
```

Name identifier management service initial URL

In a SAML 2.0 federation, the name identifier management service URL is used by a partner to contact the Name Identifier Management service.

Syntax

The URL to initiate the service has the following syntax:

```
https://provider_hostname:port_number/sps/
  federation_name/mnidsinitial?RequestBinding=RequestBindingType
  &PartnerId=target_partner_provider_ID&NameIdTerminate=[True|False]
```

Elements

https *or* **http**

The URI scheme. `https` for resources that are protected by secure sockets layer (SSL). `http` for resources that are not protected by SSL.

provider_hostname

The host name of the point of contact server for the service or identity provider.

port_number

The port number of the artifact resolution service endpoint. The default value is 9444.

sps

The designation for the Tivoli Federated Identity Manager server. This element cannot be changed.

federation_name

The name you assign to the federation when you create it.

saml20

The designation that SAML 2.0 is used in the federation.

mnidsinitial

This element indicates what type of endpoint is using the port. **mnidsinitial** is used to initiate the name identifier.

The following query strings must also be included:

RequestBinding
The binding that is used to send the request to the partner. The valid values when initiating single sign-on at the identity provider are:

- HTTPPost
- HTTPArtifact
- HTTPRedirect
- HTTPSOAP

PartnerId
The provider ID of the target partner.

NameIdTerminate
A value that indicates if the name ID management flow must terminate the name ID mapping. Valid values are:

True Ends the account linkage.

False Indicates that the name ID flow updates the name identifiers (aliases). False is the default, if no value is explicitly specified.

Examples

Name identifier initiated at the identity provider:
The following example shows the name identifier URL initiated at an identity provider in a federation named `ipfed`, using the SAML 2.0 protocol and HTTP SOAP as the request binding:

```
https://idp.example.com:9444/sps/ipfed/saml20/mnidsinitial?
  RequestBinding=HTTPSOAP&PartnerId=https://saml20sp:444/sps/
  saml20/saml20&NameIdTerminate=true
```

Name identifier initiated at the service provider:
The following example shows the name identifier URL initiated at a service provider in a federation named `spfed`, using the SAML 2.0 protocol and HTTP Artifact as the request binding:

```
https://sp.example.com:9444/sps/spfed/saml20/mnidsinitial?
  RequestBinding=HTTPArtifact&PartnerId=https://saml20ip/FIM/sps/
  saml20/saml20&NameIdTerminate=true
```

Appendix C. Using the command-line interface to configure Tivoli Federated Identity Manager SHA256 support

Learn how to configure the required parameters from the response file to support SHA256 in Tivoli Federated Identity Manager.

Procedure

1. Click **Integrated Solutions Console** > **Tivoli Federated Identity Manager** to do the following tasks:
   a. Create SAML 2.0 identity provider and service provider federations. See "Creating your role in the federation" on page 214.
   b. Export metadata file.
   c. Add partners. Ensure that you select a signing key and a signature algorithm. See Table 169 for more information on the SHA256 attributes.

Table 169. SAML 2.0 SHA256 Parameter Configuration Matrix

SAML 2.0 SHA256 Attributes	Values	Applicable to?	Remarks
`SigningKeyIdentifier`	Yes	IP and SP	Signs outgoing SAML messages and SAML assertion. If the `AssertionSigningKeyIdentifier` is specified, the `AssertionSigningKeyIdentifier` signs the SAML assertion instead. The attribute is configurable in the management console and federation response file.
`AssertionSigningKeyIdentifier`	For example: `DefaultKeyStore_ dsatestkey`	IP only	Signs outgoing SAML assertion. The attribute is configurable only in federation response file.

`SignatureAlgorithm`	`http://www.w3.org/2000/09/xmldsig#dsa-sha1`
	
	`http://www.w3.org/2000/09/xmldsig#rsa-sha1`
	
	`http://www.w3.org/2001/04/xmldsig-more#rsa-sha256`
		IP and SP	Signs outgoing SAML messages and SAML assertion.
			
			If the `AssertionSignatureAlgorithm` is specified, the `AssertionSignatureAlgorithm` signs the SAML assertion.
			
			The `SignatureAlgorithm` value must match the key type specified for the `SigningKeyIdentifier`.
			
			If the `AssertionSigningKeyIdentifier` is specified and the `AssertionSignatureAlgorithm` is not specified, the `SignatureAlgorithm` value must match the key type specified for the `AssertionSigningKeyIdentifier`.
			
			The attribute is configurable in the management console and partner response file.

Table 169. SAML 2.0 SHA256 Parameter Configuration Matrix (continued)

SAML 2.0 SHA256 Attributes	Values	Applicable to?	Remarks
DigestAlgorithm	http://www.w3.org/2000/09/xmldsig#sha1 http://www.w3.org/2001/04/xmlenc#sha256 http://www.w3.org/2001/04/xmlenc#sha512	IP and SP	Generates SAML messages and SAML assertion digest values. If the AssertionDigestAlgorithm is specified, the AssertionDigestAlgorithm hashes the SAML assertion digest. If not specified, the DigestAlgorithm becomes: • SHA1, when the SignatureAlgorithm is DSA-SHA1 or RSA-SHA1 • SHA256, when the SignatureAlgorithm is RSA-SHA256. The attribute is configurable only in partner response file.
AssertionSignatureAlgorithm	http://www.w3.org/2000/09/xmldsig#dsa-sha1 http://www.w3.org/2000/09/xmldsig#rsa-sha1 http://www.w3.org/2001/04/xmldsig-more#rsa-sha256	IP only	Signs outgoing SAML assertion. The value must match the AssertionSigningKeyIdentifier key type. The SignatureAlgorithm signs the SAML assertion if the AssertionSignatureAlgorithm is not specified. The attribute is configurable only in partner response file.
AssertionDigestAlgorithm	http://www.w3.org/2000/09/xmldsig#dsa-sha1 http://www.w3.org/2000/09/xmldsig#rsa-sha1 http://www.w3.org/2001/04/xmldsig-more#rsa-sha256	IP only	Signs outgoing SAML assertion. The value must match the AssertionSigningKeyIdentifier key type. The SignatureAlgorithm signs the SAML assertion if the AssertionSignatureAlgorithm is not specified. The attribute is configurable only in partner response file.
AssertionValidateKeyIdentifier	For example: DefaultTrustedKeyStore_IP-validationkey	SP only	The key service provider uses to validate SAML Assertion from the identity provider. The AssertionValidateKeyIdentifier must be identical to the public key of the identity provider when signing a SAML assertion. The attribute is configurable only in partner response file.

2. From the command-line interface, generate the response files of the identity provider and the service provider federation and the partners that you have added. Use the following commands:

   Identity provider federation response file:

   ```
   wsadmin>$AdminTask manageItfimFederation
   {-operation createResponseFile -fimDomainName <fimdomain>
   -federationName <IP_fedname> -fileId output_file}
   ```

   Service provider federation response file:

   ```
   wsadmin>$AdminTask manageItfimFederation
   {-operation createResponseFile -fimDomainName <fimdomain>
   -federationName <SP_fedname> -fileId output_file}
   ```

   Identity provider partner response file:

   ```
   wsadmin>$AdminTask manageItfimPartner
   {-operation createResponseFile -fimDomainName <fimdomain>
   -federationName <IP_fedname> -partnerName <Partner_name>
   -fileId output_file}
   ```

Service provider partner response file:

```
wsadmin>$AdminTask manageItfimPartner
{-operation createResponseFile -fimDomainName <fimdomain>
-federationName <SP_fedname> -partnerName <Partner_name>
-fileId output_file}
```

3. Edit the SAML 2.0 SHA256 attributes in the identity provider, service provider federation, and partner response files according to the available data in Table 170.

Table 170. Identity Provider and Service Provider SHA256 Federation and Partner Response File Parameters

SAML 2.0 SHA256 Parameter	Identity Provider	Service Provider
Federation		
SigningKeyIdentifier	<void method="put"> <string>SigningKeyIdentifier </string> <object class="java.util.ArrayList"> <void method="add"> <string>DefaultKeyStore_<*dsakey*> </string> </void> </object> </void>	<void method="put"> <string>SigningKeyIdentifier </string> <object class="java.util.ArrayList"> <void method="add"> <string>DefaultKeyStore_<rsakey> </string> </void> </object> </void>
AssertionSigningKeyIdentifier	<void method="put"> <string>AssertionSigningKeyIdentifier </string> <object class="java.util.ArrayList"> <void method="add"> <string>DefaultKeyStore_<*rsakey*> </string> </void> </object> </void>	N/A
Partner		
AssertionValidateKeyIdentifier	N/A	<void method="put"> <string>AssertionValidateKeyIdentifier </string> <object class="java.util.ArrayList"> <void method="add"> <string>DefaultTrustedKeyStore_ <*IP_publickey*> </string> </void> </object> </void>
SignatureAlgorithm	<void method="put"> <string>SignatureAlgorithm </string> <object class="java.util.ArrayList"> <void method="add"> <string>http://www.w3. org/2000/09/ xmldsig#dsa-sha1 </string> </void> </object> </void>	<void method="put"> <string>SignatureAlgorithm </string> <object class="java.util.ArrayList"> <void method="add"> <string>http://www.w3.org/2001/04/ xmldsig-more#rsa-sha256 </string> </void> </object> </void>
DigestAlgorithm	<void method="put"> <string>DigestAlgorithm </string> <object class="java.util.ArrayList"> <void method="add"> <string>http://www.w3. org/2000/09/ xmldsig#sha1 </string> </void> </object> </void>	<void method="put"> <string>DigestAlgorithm </string> <object class="java.util.ArrayList"> <void method="add"> <string>http://www.w3.org/2001/04/ xmlenc#sha512 </string> </void> </object> </void>

Table 170. Identity Provider and Service Provider SHA256 Federation and Partner Response File Parameters (continued)

SAML 2.0 SHA256 Parameter	Identity Provider	Service Provider
AssertionSignatureAlgorithm	<void method="put"> <string>AssertionSignatureAlgorithm </string> <object class="java.util.ArrayList"> <void method="add"> <string>http://www.w3. org/2001/04/ xmldsig-more#rsa-sha256 </string> </void> </object> </void>	<void method="put"> <string>AssertionSignatureAlgorithm </string> <object class="java.util.ArrayList"/> </void>
AssertionDigestAlgorithm	<void method="put"> <string>AssertionDigestAlgorithm </string> <object class="java.util.ArrayList"> <void method="add"> <string>http://www.w3.org/2001/04/ xmlenc#sha512 </string> </void> </object> </void>	<void method="put"> <string>AssertionDigestAlgorithm </string> <object class="java.util.ArraList"/> </void>

4. Update the properties of the identity provider and the service provider federations using the modified federation response file.

   For the identity provider:

   ```
   wsadmin>$AdminTask manageItfimFederation
   {-operation modify -fimDomainName <fimdomain>
   -federationName <IP_fedname>
   -fileId <Path_to_IP_federation_response_file>}
   ```

   For the service provider:

   ```
   wsadmin>$AdminTask manageItfimFederation
   {-operation modify -fimDomainName <fimdomain>
   -federationName <SP_fedname>
   -fileId <Path_to_SP_federation_response_file>}
   ```

5. Update the properties of the identity provider and the service provider partner using the modified partner response file.

   For the identity provider:

   ```
   wsadmin>$AdminTask manageItfimPartner
   {-operation modify -fimDomainName <fimdomain>
   -federationName <IP_fedname>
   -partnerName <IP_partner_name>
   -fileId <Path_to_IP_partner_response_file>}
   ```

   For the service provider:

   ```
   wsadmin>$AdminTask manageItfimPartner
   {-operation modify -fimDomainName <fimdomain>
   -federationName <SP_fedname>
   -partnerName <SP_partner_name>
   -fileId <Path_to_SP_partner_response_file>}
   ```

6. Enable the identity provider and the service provider partners.

   For the identity provider:

   ```
   wsadmin>$AdminTask manageItfimPartner
   {-operation enable -fimDomainName <fimdomain>
   -federationName <IP_fedname>
   -partnerName <IP_partner_name>}
   ```

For the service provider:

```
wsadmin>$AdminTask manageItfimPartner
{-operation enable -fimDomainName <fimdomain>
-federationName <SP_fedname>
-partnerName <SP_partner_name>}
```

7. Perform single sign-on, or single log-off.

Appendix D. Disabling logging to enhance performance

When using Tivoli Federated Identity Manager with Tivoli Access Manager, you can improve the performance on a service provider by disabling logging for theTivoli Access Manager policy server.

To reduce usage of the central processor unit (CPU), complete the following steps:

1. Back up the policy director directory. For example, on Linux or UNIX:
   `/opt/IBM/WebSphere/AppServer/java/jre/PolicyDirector`
2. Open the following file in a text editor:
   `/opt/IBM/WebSphere/AppServer/java/jre/PolicyDirector/PDJLog.properties`
3. Disable message logging by setting the following parameter to false:
   `baseGroup.PDJMessageLogger.isLogging=false`

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not 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

Special characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Printed in USA

GC27-2719-02