Sun Java System Federation Manager 7.0 User's Guide

4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–2357–12
September, 2005

070627@17466

Contents

Figures

Tables

Examples

Preface

The *Sun Java System Federation Manager 7.0 User's Guide* provides information and procedures concerning this initial release of Sun™ Java System Federation Manager. The *User's Guide* includes installation instructions, administrative procedures, developer material, technical notes, and other relevant information.

Who Should Use This Book

The *Sun Java System Federation Manager 7.0 User's Guide* is intended for use by IT professionals, network administrators and software developers who implement a standards based service provider platform using Sun Java System server software. It is recommended that administrators understand the following technologies:

- Lightweight Directory Access Protocol (LDAP)
- Java™ 2 Platform, Enterprise Edition (J2EE™ platform) web containers and corresponding deployment tools
- JavaServer Pages™
- HyperText Transfer Protocol (HTTP)
- HyperText Markup Language (HTML)
- eXtensible Markup Language (XML)
- Web Services Description Language (WSDL)
- Security Assertion Markup Language (SAML)
- SOAP
- Liberty Alliance Project Specifications for Federation and Web Services

In addition, administrators should have a clear knowledge of the legal aspects of identity federation (including privacy laws) and corporate policies.

Before You Read This Book

Federation Manager can be installed to work with Sun Java System Access Manager. Access Manager is a component of the Sun Java Enterprise System suite of products.

- Because Federation Manager can be configured to work with Access Manager, you should be familiar with the Sun Java System Access Manager documentation. Specifically, the *Sun Java System Access Manager 7 2005Q4 Federation and SAML Administration Guide* might be helpful if you have not yet worked with a Liberty-based deployment.
- Because Sun Java System Directory Server can be used as a data store in a Federation Manager deployment, you should be familiar with the Sun Java System Directory Server 5 2005Q4 documentation set.
- Because Federation Manager contains features based on the Liberty Alliance Project specifications, you should be familiar with the Liberty Alliance Project specifications.
- Because Federation Manager can work in tandem with many of the products developed as part of the Sun Java Enterprise System suite, you should be familiar with the Sun Java Enterprise System documentation.

How This Book Is Organized

Sun Java System Federation Manager 7.0 User's Guide contains instructional and conceptual material regarding the use of Sun Java System Federation Manager. The book is organized into the chapters described in the following table.

TABLE P–1 Chapters in the Federation Manager User's Guide

Chapter	Description
1 Introducing Federation Manager	An overview of Federation Manager.
2 Installing and Deploying Federation Manager	Contains the installation procedure for Federation Manager and related deployment information.
3 Customizing Federation Manager	Contains information on how to customize your Federation Manager installation. It includes procedures on how to change the default data store, enable Secure Sockets Layer (SSL), and work with Sun Java System Policy Agents.
4 Getting Started	Contains the procedures for changing default data stores and introductory material on the samples included with Federation Manager.

TABLE P–1 Chapters in the Federation Manager User's Guide *(Continued)*

Chapter	Description
5 System Administration	Contains information on how to configure your Federation Manager server and the non-Liberty Authentication Service.
6 Authentication	Contains information on how to configure your Federation Manager server for authentication.
7 Federation, Authentication Domains and Entities	Contains administrative information about federations, entities and providers.
8 SAML Administration	Contains information about the SAML component.
9 Web Services	Provides information regarding the Liberty-based web services.

Related Books

The following sections describe the documentation for Federation Manager and Sun Java System products:

- "Federation Manager Core Documentation" on page 17
- "Sun Java System Product Documentation" on page 18

Federation Manager Core Documentation

The Federation Manager documentation set contains the following titles:

- The *Sun Java System Federation Manager 7 Release Notes* will be available online after the product is released. It gathers an assortment of last-minute information, including a description of what is new in this current release, known problems and limitations, installation notes, and how to report issues with the software or the documentation.
- The *Sun Java System Federation Manager 7 User's Guide* (this guide) is the core manual for Federation Manager. It includes instructions on how to install Federation Manager and information on how to use the console and manage the meta data. It also includes some basic deployment information and describes the samples included with Federation Manager.
- The *Sun Java System Federation Manager 7 Java API Reference* are generated from Java code using the `JavaDoc` tool. The pages provide information on the implementation of the Java packages in Federation Manager.

Note – The *Sun Java System Access Manager 7 2005Q4 Federation and SAML Administration Guide* provides information about the features in Access Manager that are based on the Liberty Alliance Project and SAML specifications. Because Access Manager is also based on these specifications and might be used with Federation Manager, this guide provides useful information for a Federation Manager deployment.

Updates to the *Release Notes* and links to modifications of the core documentation can be found on the Federation Manager documentation page. Updated documents will be marked with a revision date.

Sun Java System Product Documentation

Useful information can be found in the documentation for the following Sun Java System products:

- Sun Java System Access Manager
- Sun Java System Directory Server
- Sun Java System Web Server
- Sun Java System Application Server
- Sun Java System Web Proxy Server

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in the product documentation, contact Sun Support Services.

Related Third-Party Web Site References

Third-party URLs are referenced in this documentation set and provide additional, related information. Sun is not responsible for the availability of third-party Web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Sun Welcomes Your Feedback

Sun Microsystems is interested in improving its documentation and welcomes your comments and suggestions. To share your thoughts, go to `http://docs.sun.com` and click the Send Comments link at the top of the page. In the online form provided, include the document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document. For example, the title of this book is *Sun Java System Federation Manager 7 User's Guide*, and the part number is 819–2357.

Accessing Sun Documentation Online

The docs.sun.com℠ web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is `http://docs.sun.com`.

Ordering Sun Documentation

Sun Microsystems offers select product documentation in print. For a list of documents and how to order them, see "Buy printed documentation" at `http://docs.sun.com`.

Typographic Conventions

The following table describes the typographic changes that are used in this book.

TABLE P–2 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.

TABLE P–2 Typographic Conventions *(Continued)*

Typeface or Symbol	Meaning	Example
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. Perform a *patch analysis*. Do *not* save the file. [Note that some emphasized items appear bold online.]

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–3 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

Additional Sun Resources

For product downloads, professional services, patches, support, and additional developer information, go to the following locations:

- Download Center (`http://wwws.sun.com/software/download/`)
- Technical Support (`http://www.sun.com/service/support/software/`)
- Sun Java Systems Services Suite (`http://www.sun.com/service/sunjavasystem/sjsservicessuite.html`)
- Sun Enterprise Services, Solaris Patches, and Support (`http://sunsolve.sun.com/`)
- Developer Information (`http://developers.sun.com/prodtech/index.html`)

If you have technical questions about any Sun products, contact Sun Support and Services (`http://www.sun.com/service/contacting`).

◆ ◆ ◆ CHAPTER 1

Introducing Federation Manager

Sun™ Java System Federation Manager is a server product that helps companies to quickly build federated identity and authentication services that work with existing federation hub technologies. The following sections contain overview material about Federation Manager: why the product is needed, the standards on which it is based, and how it can be used.

This chapter covers the following topics:

Sun Java System Federation Manager

Sun Java System Federation Manager delivers a solution to establish and share trusted information for single sign-on. The ability to form these trust relationships across security domains allows an organization to:

- Engage in relationships with cooperating business partners offering a variety of complementary services.
- Integrate applications offered by different departments and divisions within an enterprise.

There are many products available today, including Sun Java System Access Manager, that can be deployed for these purposes. Federation Manager is one of them: a lightweight server application that helps companies to quickly build interoperable, federated identity and authentication services. These services will work with and complement existing or newly deployed federation technologies, such as web access management solutions and authentication authorities. By leveraging these capabilities, Federation Manager can be used to build a reusable, standards-based framework to exchange security assertions, user attributes, and policies across a distributed network of partners.

Note – This *User's Guide* assumes familiarity with the Liberty Alliance Project and Security Assertions Markup Language (SAML) specifications. For an introduction, see the *Sun Java System Access Manager 7 2005Q4 Federation and SAML Administration Guide*. For more detailed information, see the Liberty Alliance Project web site or the Organization for the Advancement of Structured Information Standards (OASIS) web site.

Federation Manager provides support for heterogeneous IT environments and can integrate with all common identity repositories, application servers, and critical enterprise applications as well as with existing identity management infrastructures. Federation Manager adheres to accepted industry-standard federated identity protocols, such as Security Assertions Markup Language (SAML) and the Liberty Alliance Project specifications. It can be adapted to work with proprietary federation mechanisms and deploys easily because it leverages the core capabilities of an existing identity provider. It can also be deployed on a partner site as a single web archive (`WAR`), reducing the complexity and time for configuring a typical scenario.

Key Features

Federation Manager creates a comprehensive security and identity management framework optimized to work with and extend an identity provider's existing security infrastructure. The following list describes some key features of Federation Manager:

- Lightweight web archive (WAR) accelerates deployment of Federation Manager for service providers.
- Lightweight WAR also allows for flexibility in customizing your deployment.
- Exchange of credentials and security tokens across authentication domain partners for purposes of authentication and single sign-on.
- Automatic federation of user accounts across multiple security domains.
- Session management across authentication domains to determine when user interactions must be terminated (single logout).
- Import or export the data required to establish basic federated communication between hubs and spokes.
- Manages and links providers that are available to participate in an authentication domain.
- Searches for available end points and identifies each provider's federation capabilities.
- Exchanges SAML security assertions among providers in the authentication domain.
- Provides the tools and APIs to quickly develop, register, and enable web services on the consumer and provider sides.
- Data management choices include a proprietary flat file format (by default), and an LDAPv3 directory (Sun Java System Directory Server or Microsoft™ Active Directory).
- Separate service configuration and user data stores.

- Included service provider interfaces (SPIs) to allow customized logic during the federation process.
- Support for bulk federation and auto federation.
- Option to preload included samples.
- Sun Java System Policy Agents 2.2 can be used in SSO mode.

Federated Models

Federation Manager with Sun Java System Access Manager can provide a *hub and spoke* model of federation. Access Manager would typically be the hub, an identity provider trusted by many instances of Federation Manager acting as service providers. The following figure illustrates this hub and spoke model of federation.

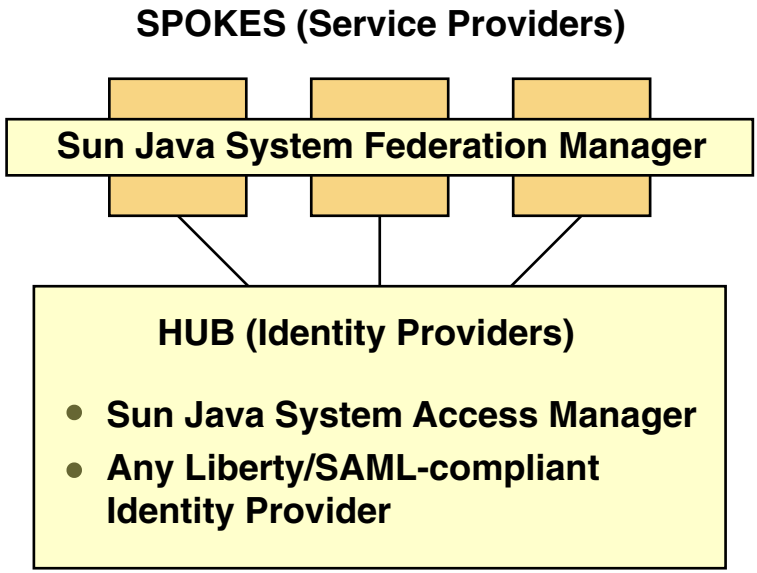

FIGURE 1–1 Hub and Spoke Model of Federation

Generally speaking, spoke service providers trust one hub identity provider. Within one organization, the hub identity provider might be administered by a human resources department using Access Manager. The spoke service providers might include other departments (legal, accounting, and the like) that need to communicate identity and session information with the hub Access Manager. Federation Manager allows the spoke service provider to enable this communication quickly and efficiently.

The hub and spoke is one model of federation. Other models that can be established using Federation Manager include a transitive trust model or a point—to—point model. The

transitive trust model assumes that because A trusts C and B trusts C, A will trust B. The point-to-point model assumes one point as an aggregation of services, service providers, or identity providers.

Federation Manager and Sun Java System Access Manager

It is not necessary to install Sun Java System Access Manager in order to use Federation Manager. Federation Manager is a standalone product that can work with any Liberty or SAML-compliant product.

Supported Standards and Components

Federation Manager installs as a single web archive (WAR), making it easy to deploy and integrate. It runs on a simple web container and requires no complex integration with data stores or application server environments. Federation Manager extends an identity provider federation framework to partners with ease, leveraging open standards and existing IT investments to help you efficiently secure your service oriented architectures. Federation Manager can also be used to create infinitely reusable application security mechanisms as it is also a Java software development kit (SDK) for Liberty and SAML—based application development.

Supported Standards

Sun Java System Federation Manager was developed using the specifications defined by these standards bodies:

- Liberty Alliance Project
- Organization for the Advancement of Structured Information Standards (OASIS) Security Services Technical Committee

It supports:

- Security Assertion Markup Language (SAML) 1.0/1.1
- Liberty Identity Federation Framework (Liberty ID-FF) 1.1/1.2
- Liberty Identity Web Services Framework (Liberty ID-WSF) 1.0

The following sections contain background information regarding these bodies and the specifications they have developed.

The Liberty Alliance Project Specifications

The goal of the Liberty Alliance Project is to define standards for developing interoperable, identity-based infrastructures, software, and web services, and to promote adoption of these standards. It does not deliver products or services. The standards provide a solution for enforcing authorized access to network services and resources. They integrate access control, identity management and service management to simplify the administration of users and organizations with regards to federation and its associated web services. A *federation* is defined as "an association formed by merging several groups or parties." The Liberty ID-FF describes more about federation and how it can be implemented. The Liberty ID-WSF describes related web services that can be implemented for use within a federated model. Among other services, Federation Manager has implemented a discovery service and a SOAP binding service.

Note – For more information on the Liberty Alliance Project, go to `http://www.projectliberty.org`.

In terms of the Liberty Alliance Project specifications, federation encompasses both *identity federation* and *provider federation* as detailed in the following sections.

Identity Federation

The concept of federation (as it has evolved with regards to the World Wide Web) begins with the notion of identity. Sending and receiving email, logging in to a news portal, checking bank balances, finalizing travel arrangements, bidding on auction items, accessing utility accounts, and shopping are all possible online services for which you might define a identity. Each time you want to access one of these services, you identify yourself by logging in to the service provider. If you use all of the mentioned services, you've configured a multitude of separate accounts to which you must log in and log out. This virtual circumstance offers the opportunity to fashion a system for computer users to correlate (or *federate*) their disparate service provider identities. This concept of *identity federation* allows the user to link, connect or bind the local identities that they have created for multiple service providers. The linked local identities, referred to as a *federated identity*, allow the user to log in to one service provider site and click through to an affiliated service provider without having to re-authenticate or re-establish their identity.

Provider Federation

The concept of federation as defined by the Liberty Alliance Project begins with the notion of a circle of trust. A *circle of trust* (referred to as an *authentication domain* in the Federation Manager Console) is a group of service providers (with at least one identity provider) who agree to join together to exchange user authentication information using Liberty-based technologies. Once a group of providers has been federated within a circle of trust, authentication accomplished by the identity provider in that circle is honored by all affiliated service providers. Thus, single sign-on can be enabled amongst all membered providers as well as identity federation among users.

The OASIS Security Services and SAML

SAML is an XML-based standard for communicating authentication, authorization and attribute information among online partners. SAML allows organizations to securely send assertions between partnered organizations regarding the identity and entitlements of a principal. The OASIS Security Services Technical Committee is in charge of defining, enhancing, and maintaining the specifications that define SAML. They incorporate XML protocols such as SOAP, XML Signature (XMLSIG), and XML Encryption (XMLENC) to define a single sign-on framework that can be used between domains. For more information on SAML, visit the OASIS web site.

Data Stores

Federation Manager configuration data, user authentication data and user federation data can be managed and retrieved from a database of the following type:

- Lightweight Directory Access Protocol (LDAP) version 3 compliant directories (for example, Sun Java System Directory Server or Microsoft® Active Directory)
- Relational Database Management Systems (through customer plug-in implementations only)
- Flat files (default text file format for configuration data)

Note – Federation Manager does not come with a user administration system.

Platforms and Operating Systems

You can install Federation Manager on the following platforms running the applicable operating systems.

TABLE 1–1 Operating Systems

Platform	Operating System	Version
Sparc®	Solaris	8 / 9 / 10
x86	Solaris	9 / 10

TABLE 1–1 Operating Systems *(Continued)*

Platform	Operating System	Version
x86	Windows	■ Windows 2000 Advanced Server SP4 or above ■ Windows 2000 Server SP4 or above ■ Windows 2000 Professional Edition SP4 or above ■ Windows XP Professional Edition SP2 ■ Windows 2003 Enterprise Server **Note –** Federation Manager was only tested on Windows 2003 Enterprise Server.
x86	Linux	Red Hat™ Enterprise Linux 2.1, 3.0, and 4.0

Shared Components

Federation Manager supports the following shared components.

Note – If you are running the Sun Java Enterprise System some of these components may already be installed.

TABLE 1–2 Shared Components

Component	Package Name	Version
Java Development Kit	N/A	1.4.2/1.5.0
Java Activation Framework	SUNWjaf	1.0.3
Java Studio Enterprise Web Application Framework	SUNWjato	2.1.4
Java Architecture for XML Binding	SUNWjaxb	1.0.3
Java API for XML Processing	SUNWjaxp	1.2.6
Sun Java System LDAP Java Development Kit	SUNWljdk	1.0
Common libraries for web service components	SUNWwscl	1.0
Java API for XML-based RPC	SUNWxrpcrt	1.1.2
SOAP with Attachments API for Java	SUNWxsrt	1.2.1

TABLE 1–2 Shared Components *(Continued)*

Component	Package Name	Version
Message Queue Java API for XML Messaging (JAXM)	SUNWiqjx	3.0.1
JavaHelp packages	SUNWjhrt	1.1.3

Supported Web Containers

Federation Manager can be deployed in the following web containers. CPU and memory requirements are based on the needs of the web container.

TABLE 1–3 Supported Web Containers

Web Container	Minimum Version
Sun Java System Web Server	6.1sp4
Sun Java System Application Server	8.1
BEA WebLogic® Server	8.1
WebSphere® Application Server	5.1

Supported Policy Agents

Federation Manager supports the use of Sun Java System Access Manager Policy Agents 2.2. For example, with 2.2 agents, user profile attributes asserted by an identity provider and SAML producer are made available as HTTP headers and for cookies. With 2.2 J2EE agents, J2EE declarative policies can map to user roles asserted by a remote identity provider and SAML producer. For more information on Federation Manager and policy agents, see "Configuring Federation Manager for Sun Java System Policy Agents" on page 66. For more information on available policy agents, see *Sun Java System Access Manager Policy Agent 2.2 User's Guide*.

Federation Manager Architecture

Federation Manager consists of web-based services [using SOAP, XML over HTTP(S) or HTML over HTTP(S)], and Java—based application provider interfaces (APIs) and service provider interfaces (SPIs). The figure below illustrates this architecture. Additionally, the figure shows an agent embedded into a web container. This agent enables the service provider applications to participate in the SAML or Liberty-based protocols. The darker boxes are components provided by Federation Manager.

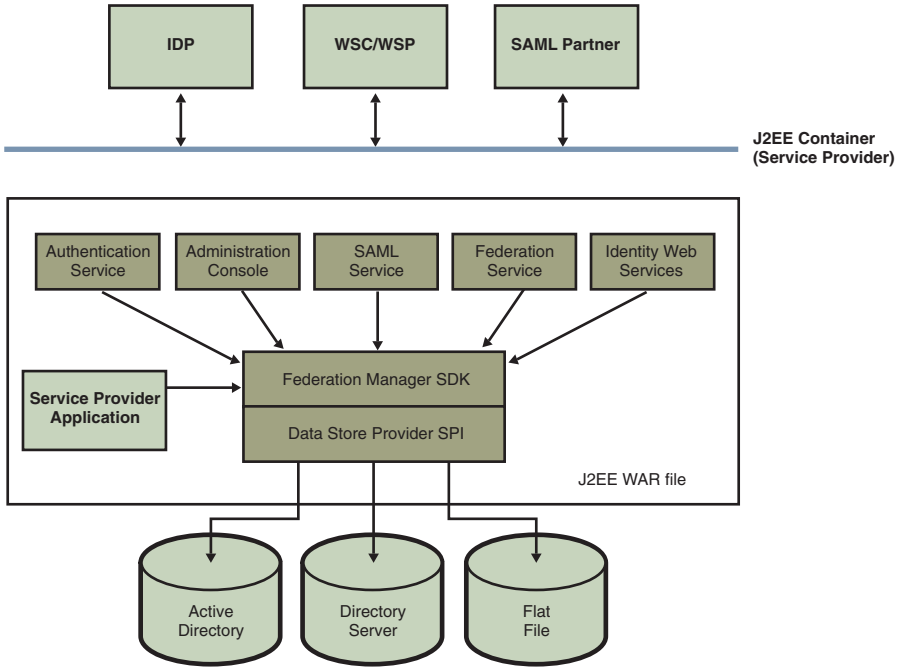

FIGURE 1–2 Architecture of Federation Manager

The Federation Manager components include:

Federation Manager Console	A web interface for managing authentication domains, provider meta data, and authentication.
SAML	Federation Manager provides SAML related services including artifact and POST profile support, and assertion query support.
Federation and associated web services	Federation Manager provides services based on the Liberty ID-FF and the Liberty ID-WSF specifications. Federation features include federation and single sign-on, single logout, federation termination, name registration, and support for the Common Domain. Implemented web services include a SOAP binding service, a discovery service, a personal profile service, and an authentication service.
Authentication	Federation Manager provides a JAAS-based authentication framework.

Session	Federation Manager provides session management for service provider applications.
Logging	Federation Manager provides a logging service. It also provides activity logs for auditing. Audit logs can be stored in flat files or JDBC-compliant databases.
Agents	Federation Manager allows service provider applications to participate in the federation protocol.
APIs	Federation Manager includes a set of APIs for interaction between the SSO, logging, SAML, Liberty ID-FF, and authentication components. Also included are APIs to build web services (Liberty ID-WSF) for clients and provider.
SPIs	Federation Manager includes a set of Service Provider Interfaces (SPIs) into which applications can insert their custom logic. For instance, there is an SPI to do post federation processing, and an SPI for post processing after a successful single logout.

◆ ◆ ◆ CHAPTER 2

Installing and Deploying Federation Manager

This chapter provides information and instructions for installing Sun Java System Federation Manager 7.0. It is recommended that you read this entire chapter before beginning with your installation.

This chapter covers the following topics:

Preparing for Installation

The following sections contain information you should know as you prepare your environment for installing Federation Manager.

Installation Requirements

Following are the minimum requirements for your Federation Manager installation:

- CPU and memory requirements are based on the needs of the web container.
- 100 MB of disk space for the working files used to create the WAR file, which is deployed to the web container.
- 100 MB of disk space where the application is deployed.
- You must have system administrator privileges to install and run Federation Manager.

Selecting and Installing a Web Container

Before deploying Federation Manager, you must install one of the supported web containers. See "Supported Web Containers" on page 28 to choose an appropriate server. See your chosen container's documentation for installation instructions.

Default Installation Directories

During a Federation Manager installation, two default directories are configured: `/opt` and `/var/opt`.

`/opt`	The default installation directory where the Federation Manager package, `SUNWamfm`, is physically copied. **Note –** The default installation directory can be changed before installation by modifying the file you create from the `fmsilent-template`. For more information, see "The Silent Installation File" on page 33.
`/var/opt`	The default runtime directory which contains debug and log file directories, a staging directory (containing the files used to generate the WAR), and default user and configuration data files.

Staging Directory

Before a Federation Manager web archive (WAR) is generated, the files in the WAR are placed in a *staging directory*. The staging directory should be kept in a permanent directory as it will be used whenever Federation Manager is redeployed or, the SAML v2 Plug-in for Federation Services is installed and its operational commands used. The recommended path is `/var/opt/SUNWam/fm/war_staging`. The staging directory can not be moved after it is created without reinstalling the product. From the staging directory, the web container deploys an instance of the web application. Thus, any modifications to the Federation Manager web application must first be made to the staging directory which is then re-archived and the resulting WAR, redeployed.

Caution – Do not put the staging directory in a volatile place such as `/tmp`.

See "Detailing the `fmwar` Syntax" on page 50 for more information on creating a Federation Manager WAR and staging directory for the first time.

Installation Sequence

This sequence of steps needs to be followed to complete your installation of Federation Manager. For more information, click the appropriate link.

1. Create a silent installation file based on your deployment.

   See "The Silent Installation File" on page 33.

2. Select an installation option for instructions on how to download and unpack the Federation Manager binary, add the Federation Manager packages to your server and generate the Federation Manager web archive (WAR) from the added packages.

   See "Federation Manager Installation Options" on page 35.

3. Deploy the generated WAR to a web container.

   See "Deploying Your Federation Manager WAR" on page 38.

The Silent Installation File

Before installing Federation Manager you must have a silent installation file for use as input to the installer. Included with the Federation Manager application is a template called `fmsilent-template`. After installation, `fmsilent-template` is located in the */FederationManager-base*`/SUNWam/fm/` directory. It is reproduced below and can be modified to create a silent installation file based on your deployment.

EXAMPLE 2–1 `fmsilent-template` Silent Installation File Template

```
XML_ENCODING=iso-8859-1
BASEDIR=/opt
IS_INSTALL_VARDIR=/var/opt/SUNWam/fm
COOKIE_ENCODE=false
SERVER_PROTOCOL=http
SERVER_DEPLOY_URI=/federation
DEPLOY_SAMPLES=true

# The following entries contain sample values!
# These should be modified for your specific installation
# and then uncommented (remove the # from the line)
#

#FM_PROCESS_USER=sample_user
#FM_PROCESS_GROUP=sample_group
#INST_ORGANIZATION=dc=sun,dc=com
#SERVER_HOST=sample.sun.com
#SERVER_PORT=58080
#ADMINPASSWD=11111111
```

Delete the # before the variables you want to modify and enter the appropriate values. Save the file and install Federation Manager according to the instructions in "Federation Manager Installation Options" on page 35. Following are definitions of the variables in the `fmsilent-template` file.

TABLE 2–1 Silent Installation File Variable Definitions

Silent Installation File Variable	Definition
`XML_ENCODING`	Defines how the XML parser will read your character set. `ISO 8859-1` defines the Latin 1 character set.
`BASEDIR`	Defines the default installation directory for Federation Manager. By default, this is `/opt`. See "Default Installation Directories" on page 32 for additional information.
`IS_INSTALL_VARDIR`	Defines the default runtime directory for Federation Manager. This directory is created by `fmwar` with the contents assigned the user and group ownership specified by the values in `FM_PROCESS_USER` and `FM_PROCESS_GROUP`. For more information, see "Default Installation Directories" on page 32.
`COOKIE_ENCODE`	Defines whether the cookie should be encoded or not. By default, this value should be set to `false`. This attribute will update the `com.iplanet.am.cookie.encode` property in `AMConfig.properties`. **Caution –** The value of this property is dependent on the web container in which Federation Manager is deployed. For all currently supported web containers, the value must be left `false`.
`SERVER_PROTOCOL`	Defines the protocol used by the web container on which Federation Manager will be deployed.
`SERVER_DEPLOY_URI`	Defines the Federation Manager WAR deployment URI. It is also used as the name of the generated `.WAR` file by default.
`DEPLOY_SAMPLES`	Defines whether the Liberty and SAML samples will be deployed as part of the Federation Manager installation.
`FM_PROCESS_USER`	Defines the user under which Federation Manager runs. **Note –** It is recommended that `FM_PROCESS_USER` specifies a user ID that is different from the super user like `root`. Typically this user is the same as the user under which the underlying web container runs.
`FM_PROCESS_GROUP`	Defines the group to which the `FM_PROCESS_USER` belongs.

TABLE 2–1 Silent Installation File Variable Definitions *(Continued)*

Silent Installation File Variable	Definition
`INST_ORGANIZATION`	Defines the root node for the Federation Manager configuration data. The value of this variable is written as `dc=orgname,dc=com`. If Federation Manager uses an LDAP directory to store its configuration, this becomes the root suffix. If Federation Manager uses a flat file to store its configuration, this value becomes the top level directory name of the file system by inverting the sequence and using the comma as a delimiter. Using the values from the template in Example 2–1, the flat file directory structure would be `/var/opt/SUNWam/fm/federation/dc=com/dc=orgname`.
`SERVER_HOST`	Defines the host of the web container on which Federation Manager will be deployed.
`SERVER_PORT`	Defines the port of the web container on which Federation Manager will be deployed.
`ADMINPASSWD`	Defines the password for the Federation Manager administrator. The default administrator is `amadmin`.

Federation Manager Installation Options

There are two different options that can be used to accomplish the actual installation step defined in the "Installation Sequence" on page 33. The first option, *standard installation*, is completed in two steps. First, you add the Federation Manager packages to your machine using `fmsetup` and then you create a WAR from these packages using `fmwar`. The second option, *self-extracting installation*, uses `fmsetup` and a `tar.gz` file to add the packages to your machine and create a WAR in one step. For instructions, see:

- "Self-Extracting Installation" on page 35
- "Standard Installation" on page 36

Tip – It is recommended that you use the "Self-Extracting Installation" on page 35 procedure.

After completing either of these installation procedures, the Federation Manager WAR must still be deployed in your web container. For instructions, see "Deploying Your Federation Manager WAR" on page 38.

Self-Extracting Installation

The Federation Manager self-extracting installation option installs the packages, and creates the Federation Manager WAR in one procedure. After completing the self-extracting installation option, see "Deploying Your Federation Manager WAR" on page 38 for further instructions.

▼ To Add Federation Manager Packages and Create a WAR

1 **Log in as root.**

2 **Create a new directory.**

```
# mkdir fm1.0

# cd fm1.0
```

3 **Download the** *file-name*`.tar.gz` **file into the new directory.**

Go to `http://www.sun.com/download/products.xml?id=44a5bbb5` to download Federation Manager.

4 **Unpack the product binaries by typing:**

```
# gunzip –dc file-name.tar.gz | tar -xvof -
```

where *file-name*`.tar.gz` is the name of the downloaded file.

5 **Run** `fmsetup` **to add the Federation Manager packages and create a WAR.**

```
#fmsetup install -s silent-installation-file
```

where *silent-installation-file* is the name of the file created from the silent installation template discussed in "The Silent Installation File" on page 33. See "Detailing the `fmsetup` Syntax" on page 49 for more information.

Caution – `fmsetup` only installs shared packages if they are not already present. If an older package is detected, the installer script will quit and display an error message identifying the older package name. To continue, you must manually remove that package using `pkgrm(1M)` and run `fmsetup` again.

6 **Follow the instructions in "Deploying Your Federation Manager WAR" on page 38 to complete the deployment.**

Standard Installation

The Federation Manager standard installation option encompasses two separate procedures.

1. The first procedure adds the Federation Manager packages to your server. See "To Add Federation Manager Packages" on page 37.
2. The second procedure creates the Federation Manager `WAR` from the added packages. See "To Archive the Federation Manager Packages" on page 38.

To Add Federation Manager Packages

This section details the first part of the standard installation procedure in which the Federation Manager packages are added to a directory on your machine. Following the package add, you must still create a staging directory and the WAR. See "To Archive the Federation Manager Packages" on page 38.

1. **Log in as root.**

2. **Create a new directory.**

```
# mkdir fm1.0

# cd fm1.0
```

3. **Paste the download URL into a browser window to download the installation bundle into the new directory.**

   Go to `http://www.sun.com/download/products.xml?id=44a5bbb5` to download Federation Manager.

4. **Unpack the product binaries by typing:**

```
# gunzip -dc file-name.tar.gz | tar -xvof -
```

   where *file-name*`.tar.gz` is the name of the downloaded file.

5. **Edit the silent installation file template as described in "The Silent Installation File" on page 33.**

6. **Run `fmsetup` to add the Federation Manager packages using the `-p` option.**

   The `-p` option is used for package installations only.

```
# fmsetup install -p -s silent-installation-file
```

   where *silent-installation-file* is the name of the file created from the silent installation template discussed in "The Silent Installation File" on page 33. See "Detailing the `fmsetup` Syntax" on page 49 for more information.

Caution – `fmsetup` only installs shared packages if they are not already present. If an older package is detected, the installer script will quit and display an error message identifying the older package name. To continue, you must manually remove that package using `pkgrm(1M)` and run `fmsetup` again.

7. **Follow the instructions in "To Archive the Federation Manager Packages" on page 38 to complete the standard installation option.**

To Archive the Federation Manager Packages

This section details the second part of the standard installation option: how to create a staging directory and archive the packages added to your machine in "To Add Federation Manager Packages" on page 37. This procedure will create the WAR used to deploy Federation Manager. After generating a WAR, see "Deploying Your Federation Manager WAR" on page 38.

1. **Log in as root.**

2. **Run `fmwar` to archive the Federation Manager packages into a WAR file.**

   ```
   # /FederationManager-base/SUNWam/fm/bin/fmwar -n name-for-WAR
   -d staging-directory -s silent-installation-file -g
   ```

   where:

name-for-WAR	Defines the name given to the created WAR.
staging-directory	Defines the path to the directory which contains the files that will be used to generate the WAR. **Caution –** The `-g` option in the example above is used only when creating a Federation Manager WAR for the first time. In these cases, the `-g` option must be specified and the `-d` option must point to a directory which does not yet exist. This directory will be the *staging directory*. A recommended path is `/var/opt/SUNWam/FM/war_staging`. If the `-g` option is not specified, `-d` must point to an existing staging directory to regenerate the WAR. See "Detailing the `fmwar` Syntax" on page 50 for more information.
silent-installation-file	Defines the name of the file created in "The Silent Installation File" on page 33.

3. **Follow the instructions in "Deploying Your Federation Manager WAR" on page 38 to complete the product installation.**

Deploying Your Federation Manager WAR

There are four web containers in which Federation Manager can be deployed. They are:

- Sun Java System Web Server 6.1sp4
- Sun Java System Application Server 8.1
- BEA WebLogic Server 8.1 (minimum version supported)
- WebSphere Application Server 5.1 (minimum version supported)

Instructions for deploying the Federation Manager `WAR` in each of these web containers can be found in the following sections.

- "Deploying Federation Manager in Sun Java System Web Server" on page 39

Deploying Federation Manager in Sun Java System Web Server

Sun Java System Web Server has the `wdeploy` command line utility to deploy a WAR file. The syntax is:

```
wdeploy deploy -u /deployment-URI -i instance-name -v vs-id -d deployment-directory WAR-file-location
```

where:

deployment-URI	Defines the URI for the application. This value must be the same as that defined for the `SERVER_DEPLOY_URI` property in "The Silent Installation File" on page 33. **Note –** Note the inclusion of the leading slash.
instance-name	Defines the instance of Web Server to which you are deploying the web application.
vs-id	Defines the virtual server ID of the instance of Web Server to which you are deploying the web application.
deployment-directory	Defines the directory to which the WAR will be deployed.
WAR-file-location	Defines the full path to the WAR file being deployed.

When you execute `wdeploy`, a web application with the defined URI and directory is added to the `server.xml` file. The WAR file is also extracted from its location and deployed in the deployment directory.

Note – More information on the `wdeploy` utility can be found in the *Sun ONE Web Server 6.1 Programmer's Guide to Web Applications*.

This section contains the following procedures:

▼ To Deploy an Instance of Federation Manager in Web Server

1 Type the following command replacing the variables with information regarding your deployment and hit Enter.

```
# WebServer-base/SUNWwbsvr/bin/https/httpadmin/bin/wdeploy deploy
-u /deployment-URI -i instance-name -v vs-id
-d WebServer-base/instance-name/deployment-URI war-file-location
```

For example:

```
# /WebServer-base/SUNWwbsvr/bin/https/httpadmin/bin/wdeploy deploy -
u /federation -i niceday.red.sun.com -v https-niceday.red.sun.com
-d /opt/SUNWwbsvr/https-niceday.red.sun.com/federation
/var/opt/SUNWam/fm/fedx/fedx.war
```

2 Restart the Web Server.

▼ To View the Federation Manager Java API Specifications

1 To view the Federation Manager Java API Specifications when it is deployed in Web Server, you must first create a symbolic link by typing:

```
# ln -s /FederationManager-base/SUNWam/fm/docs /Web Server-doc-root/javadocs
```

where:

FederationManager-base	Defines the Federation Manager installation directory.
Web Server-doc-root	Defines the Web Server directory in which you store publicly available documentation files.

2 Once the symbolic link is defined, you can access the Java API Reference via a web browser by typing a URL in the format *protocol*`://`*host*`:`*port*`/javadocs`. For example, `http://host1.sun.com:81/javadocs`.

▼ To Delete an Instance of Federation Manager from Web Server

1 Type the following command replacing the variables with information regarding your deployment:

```
# /WebServer-base/bin/https/httpadmin/bin/wdeploy delete -u /deployment-URI
-i instance-name -v vs-id -n hard
```

where:

WebServer-base	Defines the Web Server installation directory.
deployment-URI	Defines the Federation Manager URI (with leading slash).
instance-name	Defines the instance of Web Server to which the web application is deployed.
vs-id	Defines the virtual server ID of the instance of Web Server to which you are deploying the web application.

For example:

```
# /WebServer-base/SUNWwbsvr/bin/https/httpadmin/bin/wdeploy delete -u /federation
-i https-niceday.red.sun.com -v https-niceday.red.sun.com -n hard
```

2 **Hit Enter.**

Deploying Federation Manager in Sun Java System Application Server

With Sun Java System Application Server, you can use the `deploy` subcommand of the `asadmin` utility to deploy a WAR file. The syntax is:

```
# asadmin deploy --user admin-user --passwordfile filename --port port
--contextroot deployment-URI --name deployment-URI
--target instance-name WAR-file-location
```

where:

admin-user	Defines the ID of the Application Server administrator.
filename	Defines the full path to the file that stores the password of the Application Server administrator. You must manually edit this file so it can be understood by the `asadmin` utility. The password must be in the form: `AS_ADMIN_PASSWORD=`*password* where *password* is the password in text used during the installation of Application Server.
port	Defines the port for the Application Server Domain Administration Server. The default is 4849.
deployment-URI	Defines the URI for the application. This value must be the same as that defined for the `SERVER_DEPLOY_URI` property in "The Silent Installation File" on page 33.
instance-name	Defines the instance of Application Server to which the WAR will be deployed.
WAR-file-location	Defines the full path to the WAR file being deployed.

Note – The `asadmin` options listed above are those relevant to deploying Federation Manager. For more information (including the full set of options), see the *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Reference Manual*.

This section contains the following procedures:

- "To Deploy an Instance of Federation Manager in Application Server" on page 42
- "To View the Federation Manager Java API Specifications" on page 43
- "To Delete an Instance of Federation Manager from Application Server" on page 44

▼ To Deploy an Instance of Federation Manager in Application Server

1 Type the following command replacing the variables with information regarding your deployment and hit Enter.

```
# ApplicationServer-base/bin/asadmin deploy --user AS-administrator
--passwordfile filename --port port-number
--contextroot  deployment-URI --name deployment-URI
--target instance-name  war-file-location
```

For example:

```
# /opt/SUNWappserver/appserver/bin/asadmin deploy --user admin
--passwordfile /tmp/pwdfile --port 4849 --contextroot fm --name fm
--target server1 /var/opt/SUNWam/fm/war_staging/fm.war
```

2 Following the deployment, modify the Application Server `server.policy` file.

By default, `server.policy` is located in the `/var/opt/SUNWappserver/domains/`*domain-name*`/` directory. In the `server.policy` sample below, all of the capitalized contents (except `WEB-INF`) must be replaced with information applicable to your deployment.

```
// Federation Manager RELATED ADDITIONS
   grant {
     permission java.util.PropertyPermission "user.language", "write";
   };
   grant codeBase "file:${BASEDIR}/${PROD_DIR}/fm/web-src/WEB-INF/lib/am_sdk.jar" {
     permission java.net.SocketPermission "*", "connect,accept,resolve";
   };
   grant codeBase "file:${BASEDIR}/${PROD_DIR}/fm/web-src/WEB-INF/lib/am_services.jar" {
        permission java.net.SocketPermission "*", "connect,accept,resolve";
   };
   grant codeBase "file:$AS81_VARDIR/domains/$AS81_DOMAIN/applications/
   j2ee-modules/${DEPLOY_WARPREFIX}/-" {
        permission java.net.SocketPermission "*", "connect,accept,resolve";
   };
   grant {
    permission java.lang.RuntimePermission "modifyThreadGroup";
    permission java.lang.RuntimePermission "setFactory";
    permission java.lang.RuntimePermission "accessClassInPackage.*";
    permission java.util.logging.LoggingPermission "control";
    permission java.lang.RuntimePermission "shutdownHooks";
    permission javax.security.auth.AuthPermission "insertProvider.Mozilla-JSS";
    permission java.security.SecurityPermission "putProviderProperty.Mozilla-JSS";
    permission javax.security.auth.AuthPermission "getLoginConfiguration";
    permission javax.security.auth.AuthPermission "setLoginConfiguration";
    permission javax.security.auth.AuthPermission "modifyPrincipals";
    permission javax.security.auth.AuthPermission "createLoginContext.*";
    permission java.security.SecurityPermission "insertProvider.Mozilla-JSS";
```

```
 permission javax.security.auth.AuthPermission "putProviderProperty.Mozilla-JSS";
 permission java.io.FilePermission "ALL FILES", "execute,delete";
 permission java.io.FilePermission "$VAR_SUBDIR/logs/*", "delete,write";
 permission java.util.PropertyPermission "java.util.logging.config.class", "write";
 permission java.security.SecurityPermission "removeProvider.SUN";
 permission java.security.SecurityPermission "insertProvider.SUN";
 permission java.security.SecurityPermission "removeProvider.Mozilla-JSS";
 permission javax.security.auth.AuthPermission "doAs";
 permission java.util.PropertyPermission "java.security.krb5.realm", "write";
 permission java.util.PropertyPermission "java.security.krb5.kdc", "write";
 permission java.util.PropertyPermission "java.security.auth.login.config", "write";
 permission javax.security.auth.kerberos.ServicePermission "*", "accept";
 permission javax.net.ssl.SSLPermission "setHostnameVerifier";
};
```

Post installation modifications to `server.policy` are made as follows:

Replaceable Content	Default Value
`$BASEDIR`	`/opt`
`$PROD_DIR`	`SUNWam`
`$AS81_VARDIR`	`/var/opt/SUNWappserver`
`$AS81_DOMAIN`	`domain1`
`$VAR_SUBDIR`	`/var/opt/SUNWam`
`$DEPLOY_WARPREFIX`	`federation`

3 Restart the Application Server.

▼ To View the Federation Manager Java API Specifications

1 To view the Federation Manager Java API Specifications when it is deployed in Application Server, you must first create a symbolic link by typing:

`# ln -s` */FederationManager-base*`/SUNWam/fm/docs` */Application Server-doc-root*`/javadocs`

where:

FederationManager-base	Defines the Federation Manager installation directory.
Application Server-doc-root	Defines the Application Server directory in which you store publicly available documentation files.

2 **Once the symbolic link is defined, you can access the Java API Reference via a web browser by typing a URL in the format** *protocol*`://`*host*`:`*port*`/javadocs`**. For example,** `http://host1.sun.com:81/javadocs`.

▼ To Delete an Instance of Federation Manager from Application Server

1 **Type the following command replacing the variables with information regarding your deployment:**

```
# ApplicationServer-base/bin/asadmin undeploy --user AS-administrator
--passwordfile filename --host host --port port
 --target instance-name deployment-URI
```

where:

ApplicationServer-base	Defines the Application Server installation directory.
AS-administrator	Defines the administrator of the Application Server
filename	Defines the file that holds the Application Server administrator password.
host	Defines the server that is hosting Federation Manager.
port	Defines the port number used.
instance-name	Defines the instance of Application Server to which the application is deployed.
deployment-URI	Defines the Federation Manager URI (with leading slash).

For example:

```
# /opt/SUNWappserver/appserver/bin/asadmin undeploy --user admin
--passwordfile /tmp/pwdfile --host samplehost.sun.com
--port 4849 --target server1 /fm
```

2 **Hit Enter.**

Deploying Federation Manager in BEA WebLogic Server

With BEA WebLogic Server 8.1 (minimum version supported), you can use the `weblogic.Deployer` utility on the command line to deploy a WAR file. More information on this utility can be found in the BEA WebLogic Server documentation.

This section contains the following procedures:

- "To Deploy an Instance of Federation Manager in BEA WebLogic Server" on page 45
- "To Delete an Instance of Federation Manager from BEA WebLogic Server" on page 46

▼ To Deploy an Instance of Federation Manager in BEA WebLogic Server

1 Type the following command replacing the variables with information regarding your deployment and hit Enter.

```
# WebLogic-base/bin/java -classpath WebLogic-classpath weblogic.Deployer
-adminurl http://server-host:server-port -name deployment-URI
-username WebLogic-admin -password WebLogic-admin-password -target instance-name
-deploy WAR-file-location
```

where:

WebLogic-base	Defines the WebLogic Server installation directory.
WebLogic-classpath	Includes the JDK path and weblogic.jar.
server-host	Defines the WebLogic Server host to which Federation Manager will be deployed.
server-port	Defines the port of the WebLogic Server host to which Federation Manager will be deployed.
deployment-URI	Defines the URI for the application. This value must be the same as that defined for the `SERVER_DEPLOY_URI` property in "The Silent Installation File" on page 33.
WebLogic-admin	Defines the ID of the WebLogic Server super user (weblogic, by default).
WebLogic-admin-password	Defines the password of the WebLogic Server super user.
instance-name	Defines the instance of WebLogic Server to which the WAR will be deployed.
WAR-file-location	Defines the full path to the WAR file being deployed.

For example:

```
/export/bea8/jdk142_06/bin/java -classpath /export/bea8/weblogic8/lib/weblogic.jar:.
weblogic.Deployer -adminurl http://samplehost.sun.com:7001 -name /fm
-username weblogic -password 11111111 -target myserver
-deploy /var/opt/SUNWam/fm/war_staging/fm.war
```

2 Restart the WebLogic Server.

▼ To Delete an Instance of Federation Manager from BEA WebLogic Server

1 Type the following command replacing the variables with information regarding your deployment:

```
# WebLogic-base/bin/java -classpath WebLogic-classpath weblogic.Deployer -undeploy
-adminurl http://server-host:server-port -name deployment-URI
-username WebLogic-admin -password WebLogic-admin-password
-target WebLogic-Server
```

where:

WebLogic-base	Defines the WebLogic Server installation directory.
WebLogic-classpath	Includes the JDK path and weblogic.jar.
server-host	Defines the WebLogic Server host to which Federation Manager is deployed.
server-port	Defines the port of the WebLogic Server host to which Federation Manager is deployed.
deployment-URI	Defines the Federation Manager URI.
WebLogic-admin	Defines the ID of the WebLogic Server super user (weblogic, by default).
WebLogic-admin-password	Defines the password of the WebLogic Server super user.
WebLogic-Server	Defines the instance of WebLogic Server in which Federation Manager is deployed.

For example:

```
/export/bea8/jdk142_06/bin/java -classpath /export/bea8/weblogic8/lib/weblogic.jar:.
weblogic.Deployer -undeploy -adminurl http://samplehost.sun.com:7001
-name /fm -username weblogic -password 11111111 -target myserver
```

2 Hit Enter.

Deploying Federation Manager in WebSphere Application Server

Before deploying an application or deleting an instance in WebSphere Application Server 5.1 (minimum version supported), you must modify the `Jacl` (Java Action Command Language) descriptor file. Following this you may run the wsadmin.sh file from the command line. More information on both of these steps can be found in the WebSphere Application Server documentation.

This section contains the following procedures:

- "To Deploy an Instance of Federation Manager in WebSphere Application Server" on page 47
- "To Delete an Instance of Federation Manager from WebSphere Application Server" on page 48

▼ To Deploy an Instance of Federation Manager in WebSphere Application Server

1 Create a Jacl descriptor (for example, `fmsetup.jacl`) with the following content:

```
# $AdminApp install WAR-file-location {-contextroot
deployment-URI -usedefaultbindings -nopreCompileJSPs -distributeApp
-nouseMetaDataFromBinary -node WebSphereAS-node
-cell WebSphereAS-cell -server WebSphereAS-instance
-nodeployejb -appname deployment-URI -createMBeansForResources
-noreloadEnabled -reloadInterval 0 -nodeployws}
```

where:

WAR-file-location	Defines the full path to the WAR file being deployed.
deployment-URI	Defines the Federation Manager URI. This is generally the name of the WAR file without the `.WAR` suffix.
WebSphereAS-node	Defines the node under which the Application Server instance is configured.
WebSphereAS-cell	Defines the cell under which the Application Server node is configured.
WebSphereAS-instance	Defines the instance of the Application Server to which Federation Manager will be deployed.

2 Save the configuration with the following command:

```
# $AdminConfig save
```

3 After editing the Jacl descriptor, run the following command to deploy the WAR:

```
# WebSphereAS-base/bin/wsadmin.sh -f Jacl-descriptor-file
```

where:

Jacl-descriptor-file	Defines the full path to the Jacl descriptor previously created.

▼ To Delete an Instance of Federation Manager from WebSphere Application Server

1 Edit the Jacl descriptor by adding the following:

\$AdminApp uninstall *deploy-tag* **{-node** *WebSphereAS-node* **-cell** *WebSphereAS-cell* **-server** *WebSphereAS-instance***}**

where:

deploy-tag	Defines the Federation Manager URI. This is generally the name of the WAR file without the `.WAR` suffix.
WebSphereAS-node	Defines the node under which the Application Server instance is configured.
WebSphereAS-cell	Defines the cell under which the Application Server node is configured.
WebSphereAS-instance	Defines the instance of the Application Server to which Federation Manager will be deployed.

2 After editing the Jacl descriptor, run the following command to delete the instance:

*WebSphereAS-base***/bin/wsadmin.sh -f** *Jacl-descriptor-file*

Accessing the Federation Manager Console

After deploying the WAR file and restarting the web container, you can access the Federation Manager Console by typing the following URL in a web browser:

http://*host***:***port***/***deployment-URI***/UI/Login**

where:

host	Defines the server that is hosting Federation Manager.
port	Defines the port number used.
deployment-URI	Defines the Federation Manager URI.

When the login page appears, authenticate to Federation Manager by entering a user ID and password.

Installation Tools Reference

This section covers information you should know and tasks you should finish before beginning with the Federation Manager installer.

Detailing the `fmsetup` Syntax

`fmsetup` is the command-line application used to install or uninstall Federation Manager. `fmsetup` is located in the */FederationManager-base*`/SUNWam/fm/bin` directory. The syntax is:

```
fmsetup <install | uninstall> [-p]  -s  silent-installation-file-name
fmsetup -V
fmsetup -?
```

where:

`install | uninstall`	Defines whether you are installing or uninstalling Federation Manager. ■ Choose `install` to install all Federation Manager packages and supported components. ■ Choose `uninstall` to remove all Federation Manager packages and supported components.
`-p` or `--package`	Defines whether the installer will just add the Federation Manager packages or add the packages and create a WAR from them. When specified, this option only installs the Federation Manager packages. No WAR is created. **Note –** To complete installation after running `fmsetup` with the `-p` option, `fmwar` must be run to create a WAR for deployment. See "Standard Installation" on page 36 for more information. If the `-p` option is not specified, `fmsetup` will install the Federation Manager packages, create a staging directory at `/var/opt/SUNWam/FM/war_staging`, and generate a Federation Manager WAR. See "Self-Extracting Installation" on page 35 for more information.
`-s` or `--silent`	Defines the location and name of the file created from the template discussed in "The Silent Installation File" on page 33.
`-V`	Displays version information.
`-?`	Displays help information.

Detailing the `fmwar` Syntax

`fmwar` is the command-line application used to create (or recreate) the WAR for Federation Manager from packages that have already been added. (See "Detailing the `fmsetup` Syntax" on page 49.) `fmwar` is located in the */FederationManager-base*`/SUNWam/fm/bin` directory. By default, `fmwar` can be used for either of the following scenarios:

- Create a new staging directory and generate a WAR.

  Note – This scenario is used when Federation Manager is being installed for the first time. The `-g` option must be specified and the `-d` option must point to a directory which does not yet exist. This directory will be the staging directory.

- Regenerate a WAR from files in a staging directory that already exists.

  Note – This scenario is used when Federation Manager has already been deployed. When files in an already existing staging directory are modified, a new WAR must be archived for redeployment. In this case, since the staging directory already exists, the `-g` option is not specified.

From the installed packages, `fmwar` will create a WAR used to deploy Federation Manager. The syntax for using `fmwar` is:

`fmwar -n` *name-of-WAR* `-d` *staging-directory* `-s` *silent-installation-file-name* `[-g]`

`fmwar -V`

`fmwar -?`

where:

`-n` or `--name`	Defines the name of the WAR to be generated. The value does not include the `.WAR` suffix. This is referred to as the application's URI.
`-d` or `--directory`	Defines the path to the staging directory which contains the files that will be used to generate the WAR. A recommended path is `/var/opt/SUNWam/FM/war_staging`. **Note –** The `-d` option is used in tandem with the `-g` option. If the `-g` option is specified, `-d` must point to a staging directory which does not yet exist. If the `-g` option is not specified, `-d` must point to an existing staging directory. The latter scenario is used to regenerate the WAR.
`-s` or `--silent`	Defines the path to, and name of, the file created from the template discussed in "The Silent Installation File" on page 33.

`-g`	Creates a new staging directory, generates a WAR, and configures the flat file data store. **Note –** The `-g` option is used only when creating a Federation Manager WAR for the first time. In these cases, the `-g` option must be specified and the `-d` option must point to a directory which does not yet exist. This directory will be the staging directory. A recommended path is `/var/opt/SUNWam/FM/war_staging`. If the `-g` option is not specified, `-d` must point to an existing staging directory to regenerate the WAR.
`-V`	Displays version information.
`-?`	Displays help information.

◆ ◆ ◆ CHAPTER 3

Customizing Federation Manager

This chapter contains procedures to customize the default installation of Sun Java™ System Federation Manager. It includes procedures on how to change the default data store, enable Secure Sockets Layer (SSL), and work with Sun Java System Policy Agents.

This chapter contains the following topics:

General Customizations

Federation Manager deploys as a single web archive (WAR) making it easy to install, integrate and customize. If there is a need to customize Federation Manager, you can go to the staging directory, make changes to the relevant files, archive the WAR again, and redeploy the new WAR file to your web container. The default staging directory is `/var/opt/SUNWam/fm/war_staging`. See "Detailing the `fmwar` Syntax" on page 50 for information on regenerating a WAR.

Note – In addition to the customizations described in this chapter, 8 Authentication contains a section on customizing the Authentication component of Federation Manager. For information, see "Customizing Authentication" on page 87.

Using an LDAPv3–compliant Directory

When Federation Manager is deployed, all server configuration and user data is stored in a flat file. After installation, Federation Manager can be reconfigured to retrieve this data from an LDAPv3–compliant directory. This section contains information on how this is done as well as information about the `serverconfig.xml` file which provides configuration information regarding the directory.

Changing the Default Data Store for Configuration Data to an LDAPv3–compliant Directory

When Federation Manager is deployed, the server configuration data is stored in a flat file. After installation, Federation Manager can be reconfigured to retrieve this data from an LDAPv3–compliant directory. To proceed, you must follow this sequence:

1. Set up your LDAPv3–compliant directory.

   See "Setting Up Your LDAPv3–Compliant Directory" on page 54.
2. Install your instance of Federation Manager.

   See Chapter 2.
3. Reconfigure the Federation Manager configuration data.

   See "Modifying Federation Manager Configuration Data to Recognize an LDAPv3–compliant Directory" on page 56.
4. Migrate the new data to the directory.

   See "Building and Loading LDIF Configuration Data Using `fmff2ds`" on page 58.

Setting Up Your LDAPv3–Compliant Directory

The following sections contain procedures for setting up the supported LDAPv3–compliant directories.

Note – Although Federation Manager has only been tested on supported LDAPv3–compliant directories, it should work with any LDAPv3-compliant directory server.

This section contains the following procedures:

- "To Set Up Sun Java System Directory Server as a Configuration Data Store" on page 55
- "To Set Up Microsoft Active Directory as a Configuration Data Store" on page 55

To Set Up Sun Java System Directory Server as a Configuration Data Store

In order to change the Federation Manager data store for configuration data to Sun Java System Directory Server, follow the procedure described below in "To Set Up Sun Java System Directory Server as a Configuration Data Store" on page 55.

1. **Install Directory Server based on the instructions in the** *Sun Java Enterprise System 2005Q4 Installation Guide for UNIX*.
2. **Install Federation Manager based on the instructions in Chapter 2.**
3. **Configure Federation Manager to communicate with Directory Server based on the instructions in "Modifying Federation Manager Configuration Data to Recognize an LDAPv3–compliant Directory" on page 56.**
4. **Build the new LDAP-based configuration data from the flat file data, and load the data and accompanying schema into Directory Server based on the instructions in "Building and Loading LDIF Configuration Data Using `fmff2ds`" on page 58.**

To Set Up Microsoft Active Directory as a Configuration Data Store

In order to change the Federation Manager data store for configuration data to Microsoft Active Directory, you must set up the directory and load the Federation Manager LDIF schema. The procedure is described in "To Set Up Microsoft Active Directory as a Configuration Data Store" on page 55.

Note – When the Active Directory installation wizard asks you to type a new domain, you may type a non-existent domain as in `xyz.com`. In this example, the root suffix will be `dc=xyz,dc=com`.

---

1. **Install Microsoft Active Directory in either a Microsoft Windows 2000 Advanced Server or a Microsoft Windows 2003 Advanced Server.**

   The procedures for these installations can be found in your Active Directory documentation or on the Microsoft web site.

2. **Install the Active Directory Schema Snap-in.**

   Instructions for installing the Active Directory Schema Snap-in can also be found on the Microsoft web site

3. **Open the Microsoft Management Console (MMC).**

   Using this console you can load the LDIF schema into Active Directory.

4. **Point your cursor to Active Directory Schema and hold the right mouse button down.**

5. **Select Operations Master... from the drop-down menu.**

6. **Check The Schema may be modified on this Domain Controller from the Change Schema Master" window and click OK.**

   This enables schema modification. The administrator DN is `cn=administator,cn=users,`*ROOT-SUFFIX*`.`

7. **Install and configure Federation Manager according to the information in "Modifying Federation Manager Configuration Data to Recognize an LDAPv3–compliant Directory" on page 56.**

Modifying Federation Manager Configuration Data to Recognize an LDAPv3–compliant Directory

After installing your LDAPv3–compliant directory you must reconfigure certain of the Federation Manager configuration data before migrating it to the directory. If the Federation Manager WAR is exploded, you must restart the web container after making these changes. If the Federation Manager WAR is not exploded, make your changes in the staging directory (located by default in the *IS_INSTALL_VARDIR*`/war_staging` directory where *IS_INSTALL_VARDIR* is defined in the silent installation file detailed in "The Silent Installation File" on page 33), regenerate the WAR, and deploy the modified WAR. The following instructions assume that the WAR has been exploded.

▼ To Set Up Federation Manager for an LDAPv3–compliant Directory

Before You Begin If Federation Manager is working solely against an LDAPv3–compliant directory, you must create two users in the directory with the correct read and write privileges to the `ou=services` tree: `amadmin` and `dsameuser`. See "`serverconfig.xml` Users" on page 66.

1. **Install Federation Manager according to the instructions in Chapter 2.**

2. **Edit the default `ServerGroup` in the `serverconfig.xml` file as follows:**

   - Change the `host`, `port`, and `type` attributes of the `Server` tag to reflect your directory's installation.

- Change the `DirDN` and `DirPassword` attributes of the `User` tag in both the `proxy` and `admin` entries to reflect an existing user `DN` and password (encrypted using `ampassword`). Alternately, you can create a new administrator in the directory. This new user must have read, search, write and delete permission on the `ou=services` subtree of the directory information tree (DIT) containing the Federation Manager configuration data once the data store has been changed to Open LDAP.

  Note – Ensure the proper user permissions have been allocated. This should be done after running `fmff2ds`.

- Change the values of the `BaseDN` to that of the parent DN containing the configuration data. For example, `dc=sun,dc=com`.

See

3 Edit the `AMConfig.properties` file as follows:

- Change the value of the `com.sun.identity.sm.sms_object_class_name` property to `com.sun.identity.sm.ldap.SMSLdapObject`.
- If the `DirDN` specified in the step above is different from the default `amadmin`, you need to modify the `com.sun.identity.authentication.special.users` property by adding (or replacing) the specified DN of the directory's super user. This property may contain a pipe-separated list of user DNs as in:
  ```
  com.sun.identity.authentication.special.users=
  cn=dsameuser,ou=DSAME
  Users,dc=sun,dc=com|cn=administrator,cn=users,dc=sun,dc=com.
  ```

`AMConfig.properties` is located in the */exploded-FM-WAR-directory*`/WEB-INF/classes` directory where *exploded-FM-WAR-directory* is the directory to which the Federation Manager WAR was deployed.

4 Run `fmff2ds` according to the information in "Building and Loading LDIF Configuration Data Using `fmff2ds`" on page 58.

5 Restart the web container.

Federation Manager is now communicating with Directory Server.

▼ To Add a Second Instance of Federation Manager to the Server List

This procedure is useful when an LDAPv3–compliant directory is used for storing configuration data. If creating a *second* instance of Federation Manager by running `fmwar`, you will need to add the new instance to the Server List attribute of the *original* instance of Federation Manager before starting the new instance. The following procedure describes how to do this.

1. **Login to the console of the original instance of Federation Manager as administrator.**
2. **Select the Configuration tab.**
3. **Select Platform under System Properties.**
4. **Enter the new instance in the Server List global attribute and click Add.**

   The value is entered in the form *protocol*://*host-name*:*port*|*instance-name*. For example, **`http://host2.sun.com:81|02`**.

Building and Loading LDIF Configuration Data Using `fmff2ds`

`fmff2ds` is a command-line utility used to convert service configuration data from a flat file to an LDIF file. This utility does not work with user data. `fmff2ds` performs the following functions:

- Builds and loads an LDAP schema for your directory based on included templates.
- Builds an LDIF file from a configuration data flat file and loads it into the directory.

The syntax for `fmff2ds` is:

`fmff2ds [-a] -h` *directory-server-host* `-p` *directory-server-port* `-r` *root-suffix* `-f` *flat-file* `-u` *userDN* `[-w` *userPW*`] -j` *Java-directory*

`fmff2ds -V`

`fmff2ds -?`

where:

`-a, --activedirectory`	By default, `fmff2ds` assumes Sun Java System Directory Server is the underlying directory server. By specifying `-a`, `fmff2ds` will load data into Microsoft Active Directory.
`-r, --rootsuffix`	Defines the root suffix of the underlying directory server.
`-f, --flatfile`	Defines the location of the flat file directory. By default, `/var/opt/SUNWam/fm/`*URI*.
`-u, --username`	Defines the distinguished name of the user connecting to the directory server.
`-w, --password`	Defines the password of the user connecting to the directory server. If this option is not specified, the user will be prompted to type the bind password for the specified userDN. When the option is not given and the DSHOME property is not set to the Sun Directory Server installation root directory, user will be prompted to enter bind password for three times during the process.

`-j, --java`	*Java-directory* defines the directory where the Java Development Kit (JDK) is located. The JDK must be version 1.4.2 or higher.
`-V`	Displays version information.
`-?`	Displays help information.

Changing the Default Data Store for User Data to an LDAPv3–compliant Directory

After installing Federation Manager, its components are configured to work with user data that is stored in flat files. Following are the steps to change the default data store for user data from a flat file to an LDAPv3–compliant directory. User data can then be migrated to the directory based on the loaded LDIF schema. The following schemas are included with Federation Manager for this purpose.

TABLE 3–1 LDIF Schema Files for LDAPv3–compliant Directory

Schema File	LDAP Directory	Location
`fm_liberty_sds_schema.ldif`	Directory Server	*/FederationManager-base*`/SUNWam/fm/ldif/`
`fm_liberty_ad_schema.ldif`	Active Directory	*/FederationManager-base*`/SUNWam/fm/ldif/`

Note – Federation Manager does not contain scripts to migrate existing user data. Transferring any existing user data is outside the scope of this guide.

▼ To Change the Default Data Store for User Data

If the Federation Manager WAR is exploded, you must restart the web container after making these changes. If the Federation Manager WAR is not exploded, make your changes in the staging directory, regenerate the WAR, and deploy the modified WAR. The following instructions assume that the WAR has been exploded.

1. **Install your LDAPv3–compliant directory according to the product's documentation.**
2. **Locate the LDIF schema file for your directory server under** */FederationManager-base*`/SUNWam/fm/ldif/`**.**
   - **Use** `fm_liberty_sds_schema.ldif` **if migrating data to Sun Java System Directory Server.**

- **Use `fm_liberty_ad_schema.ldif` if migrating data to Microsoft Active Directory.**

  You will need to replace the value of the `ORG_ROOT_SUFFIX` property with the appropriate root suffix before loading.

- **For all other LDAPv3–compliant directories, create an LDIF schema file accordingly.**

3. **Load the appropriate LDIF schema file into your directory by typing:**

   `ldapmodify -a -c -h` *host* `-p` *port* `-D` *bindDN* `-w` *bindpwd* `-f` *ldif-file*

4. **Enable the equality index for the `iplanet-am-user-federation-info-key` property in your directory server.**

   If using Active Directory, indexing is already enabled through the attribute schema definition.

5. **Edit the `AMConfig.properties` file by changing the value of the `com.sun.identity.common.datastore.provider.default` property from `com.sun.identity.common.FileDataStoreProvider` to `com.sun.identity.common.LDAPDataStoreProvider`**

   `AMConfig.properties` is located in the */FederationManager-base*`/fmwar/web-src/WEB-INF/classes` directory.

6. **Modify the `userdefault` ServerGroup in the `serverconfig.xml` file as follows:**

   - Change the `host`, `port`, and `type` attributes of the `Server` tag to reflect your directory's configuration.
   - Change the `DirDN` and `DirPassword` attributes in both the `proxy` and `admin` User tags to reflect an existing user `DN` and password (encrypted using `ampassword`). Alternately, you can create a new administrator in the directory. This new user must have read, search, write and delete permission on all users to be managed.
   - Change the value of the `BaseDN` to that of the people container of your directory. For example, `ou=People,dc=sun,dc=com`.

   `serverconfig.xml` is located in the */exploded-FM-WAR-directory*`/WEB-INF/config` directory where *exploded-FM-WAR-directory* is the directory to which the Federation Manager WAR was deployed.

7. **Restart the web container.**

The `serverconfig.xml` File

The `serverconfig.xml` file provides configuration information to Federation Manager regarding the LDAPv3–compliant directory being used as its data store. `serverconfig.xml` is located in the */exploded-FM-WAR-directory*`/WEB-INF/config` directory where

exploded-FM-WAR-directory is the directory to which the Federation Manager WAR was deployed. It contains the parameters used to establish the LDAP connection pool to the LDAPv3–compliant directory.

EXAMPLE 3–1 serverconfig.xml

```
<?xml version="1.0" encoding="XML_ENCODING" standalone="yes"?>
<!--
    Copyright (c) 2004 Sun Microsystems, Inc. All rights reserved
    Use is subject to license terms.
-->

<iPlanetDataAccessLayer>
    <ServerGroup name="default" minConnPool="1" maxConnPool="10">
        <Server name="Server1" host="DIRECTORY_SERVER" port="DIRECTORY_PORT" type="SIMPLE" />
        <User name="User1" type="proxy">
            <DirDN>
                uid=amadmin,ou=people,NORMALIZED_ORGBASE
            </DirDN>
            <DirPassword>
                ENCADMINPASSWD
            </DirPassword>
        </User>
        <User name="User2" type="admin">
            <DirDN>
                uid=amadmin,ou=people,NORMALIZED_ORGBASE
            </DirDN>
            <DirPassword>
                ENCADMINPASSWD
            </DirPassword>
        </User>
        <BaseDN>
            NORMALIZED_RS
        </BaseDN>
    </ServerGroup>
    <ServerGroup name="internalauthentication" minConnPool="1" maxConnPool="10">
        <Server name="Server1" host="DIRECTORY_SERVER" port="DIRECTORY_PORT" type="SIMPLE" />
        <User name="User1" type="proxy">
            <DirDN>
                uid=amadmin,ou=people,NORMALIZED_ORGBASE
            </DirDN>
            <DirPassword>
                ENCADMINPASSWD
            </DirPassword>
        </User>
        <User name="User2" type="admin">
            <DirDN>
                uid=amadmin,ou=people,NORMALIZED_ORGBASE
```

EXAMPLE 3–1 serverconfig.xml *(Continued)*

```
            </DirDN>
            <DirPassword>
                ENCADMINPASSWD
            </DirPassword>
        </User>
        <BaseDN>
            NORMALIZED_RS
        </BaseDN>
    </ServerGroup>
    <!--
    Modify this ServerGroup to point to your LDAP server for user data.
    If your component needs to use a different LDAP server, add a
    ServerGroup for that component. Some of the possible ServerGroup names
    you can use are: saml, federation, disco, idpp, and authnsvc.

    Make sure user dn in your LDAP server uses the same naming attribute as
    your "admin" user; and baseDN is the people container LDAP server uses.
    -->
    <ServerGroup name="userdefault" minConnPool="1" maxConnPool="10">
        <Server name="Server1" host="DIRECTORY_SERVER" port="DIRECTORY_PORT" type="SIMPLE" />
        <User name="User1" type="proxy">
            <DirDN>
                uid=amadmin,ou=people,NORMALIZED_ORGBASE
            </DirDN>
            <DirPassword>
                ENCADMINPASSWD
            </DirPassword>
        </User>
        <User name="User2" type="admin">
            <DirDN>
                uid=amadmin,ou=people,NORMALIZED_ORGBASE
            </DirDN>
            <DirPassword>
                ENCADMINPASSWD
            </DirPassword>
        </User>
        <BaseDN>
            ou=people,NORMALIZED_RS
        </BaseDN>
    </ServerGroup>
</iPlanetDataAccessLayer>
```

This section contains information on the following:

`server-config.dtd` Definition Type Document

The `server-config.dtd` defines the structure for `serverconfig.xml`. It is located in */FederationManager-base/*XXXXXXXX.

EXAMPLE 3–2 `server-config.dtd`

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<--
    Copyright (c) 2002 Sun Microsystems, Inc. All rights reserved.

    Use is subject to license terms.
-->

<!-- The root.-->
<!ELEMENT iPlanetDataAccessLayer (ServerGroup+) >


<!ELEMENT ServerGroup Server+ User+ BaseDN MiscConfig*>
<!ATTLIST ServerGroup
    name    ID        #REQUIRED
    minConnPool    NMTOKEN        "1"
    maxConnPool    NMTOKEN        "10"
>


<!ELEMENT BaseDN (#PCDATA) >

<!-- An Server contains an id, host name and port. -->
<!ELEMENT Server>
<!ATTLIST Server
          name    ID    #REQUIRED
          host    CDATA    #REQUIRED
          port    NMTOKEN    "389"
          type    (SIMPLE|SSL)    "SIMPLE"
>

<!--An User contains an ID, the type of privileges the DN and Password
provides. The type of connection of a DirInstance is realized from the type
of User it is associated with or it contains.-->
<!ELEMENT User (DirDN, DirPassword)>
<!ATTLIST User
          name    ID    #REQUIRED
          type    (auth|proxy|rebind|admin)    "auth"
>

<!ELEMENT DirDN (#PCDATA)>
```

EXAMPLE 3–2 server-config.dtd *(Continued)*

```
<!ELEMENT DirPassword (#PCDATA)>
<!ELEMENT MiscConfig EMPTY>
<!ATTLIST MiscConfig
          name    CDATA   #REQUIRED
          value    CDATA   #IMPLIED
>
```

This section defines the main elements of the DTD.

iPlanetDataAccessLayer Element

iPlanetDataAccessLayer is the root element. It allows for the definition of multiple server groups per XML file. Its immediate sub-element is the ServerGroup. It contains no attributes.

ServerGroup Element

ServerGroup defines a pointer to one or more LDAPv3–compliant directories. They can be master or replica servers. The sub-elements that qualify ServerGroup include:

- Server
- User
- BaseDN
- MiscConfig

The ServerGroup attributes are the name of the server group, and values for minConnPool and maxConnPool which define the minimum (1) and maximum (10) connections that can be opened for the LDAP connection pool respectively. More than one defined ServerGroup element is not supported.

Note – Federation Manager uses a connection pool to access the LDAPv3–compliant directory. All connections are opened when Federation Manager starts and are not closed. They are reused.

`Server` Element

`Server` defines a specific LDAPv3–compliant directory instance. It contains no sub-elements. The required XML attributes are a user-friendly name for the server, the host name, the port number on which the LDAPv3–compliant directory runs, and the type of LDAP connection that must be opened (either simple or SSL).

`User` Element

`User` contains sub-elements that define the user configured for the instance of the LDAPv3–compliant directory. The `User` sub-elements are `DirDN` and `DirPassword`. It's required XML attributes are the name of the user, and the type of user. The values for type identify the user's privileges and the type of connection that will be opened to the directory instance. Options include:

- `auth` defines a user authenticated to the LDAPv3–compliant directory.
- `proxy` defines a proxy user for the LDAPv3–compliant directory. See "Proxy User" on page 66.
- `rebind` defines a user with credentials that can be used to rebind to the LDAPv3–compliant directory.
- `admin` defines a user with administrative privileges for the LDAPv3–compliant directory. See "Admin User" on page 66.

`DirDN` Element

`DirDN` contains the LDAP Distinguished Name (DN) of the defined user.

`DirPassword` Element

`DirPassword` contains the defined user's encrypted password.

Caution – It is important that passwords and encryption keys are kept consistent throughout the deployment. For example, the passwords defined in this element are stored in the LDAPv3–compliant directory. If the password is changed in one place, it must be updated in all places where it is defined. Additionally, this password is encrypted. If the encryption key defined in the `am.encryption.pwd` property is changed, all passwords in `serverconfig.xml` must be re-encrypted using `ampassword`.

`BaseDN` Element

`BaseDN` defines the base DN for the server group. It contains no sub-elements and no XML attributes.

`MiscConfig` Element

`MiscConfig` is a placeholder for defining any LDAP JDK features like cache size. It contains no sub-elements. It's required XML attributes are the name of the feature and its defined value.

`serverconfig.xml` Users

Two users are defined in `serverconfig.xml`:

- "Proxy User" on page 66
- "Admin User" on page 66

Proxy User

`serverconfig.xml`

The Proxy User can take on any user's privileges (for example, the organization administrator or an end user). The connection pool is created with connections bound to the proxy user. Federation Manager creates a proxy user with the DN of `cn=puser,ou=DSAME Users,dc=example,dc=com`. This user is used for all queries made to the LDAPv3–compliant directory. It benefits from a proxy user ACI already configured in the LDAPv3–compliant directory and, therefore, can perform actions on behalf of a user, when necessary. It maintains an open connection through which all queries are passed (retrieval of service configurations, organization information, etc.). The proxy user password is always encrypted.

Admin User

`dsameuser`, the administrator user, is used for binding purposes when the Federation Manager SDK performs operations on the LDAPv3–compliant directory that are not linked to a particular user (for example, retrieving service configuration information). The Proxy User performs these operations on behalf of the `dsameuser`, but a bind must first validate the `dsameuser` credentials. During installation, Federation Manager creates `cn=dsameuser,ou=DSAME Users,dc=example,dc=com`.

Configuring Federation Manager for Sun Java System Policy Agents

Federation Manager supports both types of Sun Java System Policy Agents 2.2: web agents and J2EE agents. They can be deployed to work with Federation Manager in any web container on which the agents are supported although they will only work in SSO only mode. No policy or access control is supported.

With respect to passing attributes to an application using agents, there are some limitations. Agents can not retrieve user attributes stored directly in Federation Manager as the user management functionality is not enabled. Either of the following options will alleviate this issue.

1. User attributes can be passed from identity providers and delivered to service providers (using Federation Manager) in SAML assertions. Federation Manager can mark the attributes as session attributes that policy agents will consume and pass to the applications.
2. Implement a post authentication and/or federation adapter SPI for the service provider (using Federation Manager) to read user attributes and set them as session attributes. The policy agent will consume the session attributes and pass them to the applications.

The following steps must be taken in order for Sun Java System Policy Agents 2.2 to work correctly with Federation Manager. On the agent side, you must do the following:

1. When installing the agent, specify `/fm` for the following URI: `Primary Server Deployment URI [/amserver] = /fm`
2. After installation, modify the following values in the `AMAgents.properties` file:
   - `com.sun.am.policy.agents.config.do_sso_only = true`
   - `com.sun.am.policy.agents.config.profile.attribute.fetch.mode=NONE`
   - `com.sun.am.policy.agents.config.response.attribute.fetch.mode=NONE`
3. Restart the agent.

On the Federation Manager side, you must do the following:

1. Encrypt the shared secret password using the `ampassword` utility.
2. Modify the `com.iplanet.am.service.secret` property in `AMConfig.properties` by specifying the encrypted password as the value for the property.
3. Restart Federation Manager.

◆ ◆ ◆ CHAPTER 4

Getting Started

Sun™ Java System Federation Manager is installed with a number of samples that can be used to familiarize yourself with the product. This chapter contains general information on the samples. The `README` file included in each sample directory contains the procedures that describe how to run the sample.

This chapter contains the following topics:

SAML Samples

The SAML samples are located in the */FederationManager-base*`/SUNWam/fm/samples/saml` directory. They include:

All samples are compiled and deployed during installation.

Note – For general information about the SAML implementation (including definitions of the profile types and assertions), see Chapter 9, "SAML Administration," in *Sun Java System Access Manager 7 2005Q4 Federation and SAML Administration Guide*.

SAML Client Sample

This sample illustrates how the SAML client API in the `com.sun.identity.saml` package can be used. Federation Manager is deployed on two machines. The `AssertionArtifactSample`

servlet on machine #1 generates an assertion artifact which is received by the `SAMLClientSample` servlet on machine #2. This sample is located in the */FederationManager-base*`/SUNWam/fm/samples/saml/client` directory.

SAML Query Sample

This sample illustrates how to use Federation Manager to form a query, to write an `AttributeMapper`, and to send and process a SOAP message using the SAML API. This sample is located in the */FederationManager-base*`/SUNWam/fm/samples/saml/query` directory.

Single Sign On Using SAML

This sample illustrates how to achieve single sign on for a user between two instances of Federation Manager using either the Web Browser Artifact Profile or the Web Browser POST Profile. This sample is located in the */FederationManager-base*`/SUNWam/fm/samples/saml/sso` directory.

SAML XMLSIG Sample

This sample illustrates how to use the `com.sun.identity.saml.xmlsig` API in a SAML interaction. This sample is located in the */FederationManager-base*`/SUNWam/fm/samples/saml/xmlsig` directory.

Liberty-based Samples

The Liberty-based samples are located in the */FederationManager-base*`/SUNWam/fm/samples/liberty` directory. They include:

- "Federation SPI Sample" on page 70
- "Liberty Protocol Sample" on page 71
- "Web Service Sample" on page 71

Federation SPI Sample

This sample illustrates how to Liberty-enable a password—protected application without changing the application code. It uses the federation SPI to achieve password playback within the sample application, and also makes use of a service provider and an identity provider. This sample is located in the */FederationManager-base*`/SUNWam/fm/samples/liberty/spi` directory.

Liberty Protocol Sample

This sample highlights different Liberty—based protocol (such as Account Federation, SSO, Single Logout and Federation Termination) using the communication channel between a service provider and an identity provider. This sample is located in the */FederationManager-base*/SUNWam/fm/samples/liberty/sso directory.

Web Service Sample

This sample describes the development of a simple web service using Federation Manager. This web service sends stock data based on a defined user type. The sample extracts authentication information from an authenticated SAML Assertion (SAML Bearer Token). It is located in the */FederationManager-base*/SUNWam/fm/samples/liberty/webservices/stockticker directory.

◆ ◆ ◆ CHAPTER 5

System Administration

The deployment of Sun Java System Federation Manager can be customized by values you set in the system attributes. In addition, Federation Manager has a non-Liberty authentication service.

This chapter contains the following topics:

Organization Properties

The following attributes can be configured for your organization.

Domain Name

This defines the domain to which the organization belongs.

Note – This attribute is not currently supported.

Status

This defines whether the realm is active or inactive.

Aliases

This defines an alias for the realm.

DNS Alias Names

This defines DNS aliases for the realm.

Console Properties

The Console properties include settings to customize the console for different locales and character sets. This includes:

- "Globalization Settings" on page 74

Globalization Settings

The Globalization Settings contain attributes that enable you to configure Federation Manager for different locales and character sets. The attributes include:

- "Charsets Supported by Each Locale" on page 74
- "Charset Aliases" on page 74
- "Auto Generated Common Name Format" on page 75

Charsets Supported by Each Locale

This attribute lists the character sets supported for each locale, which indicates the mapping between locale and character set.

The following tasks are associated with character set support.

- "To Add Supported Character Sets" on page 75
- "To Edit Supported Character Sets" on page 75

Charset Aliases

This attribute lists the codeset names (which map to IANA names) that will be used to send the response. These codeset names do not need to match Java codeset names. Currently, there is a hash table to map Java character sets into IANA character sets and vice versa. The following tasks are associated with character set aliases.

The following tasks are associated with character set support.

- "To Add New Character Set Aliases" on page 76

- "To Edit Existing Character Set Aliases" on page 76

Auto Generated Common Name Format

This display option allows you to define the way in which a name is automatically generated to accommodate name formats for different locales and character sets. The default syntax is as follows (please note that including commas and/or spaces in the definition will display in the name format):

```
en_us = {givenname} {initials} {sn}
```

For example, if you wanted to display a new name format for a user (User One) with a uid (11111) for the Chinese character set, define:

```
zh = {sn}{givenname}({uid})
```

The display is:

```
OneUser 11111
```

▼ To Add Supported Character Sets

1 Click Add under Charsets Supported by Each Locale.

The New Supported Character Sets page is displayed.

2 Define the following attributes.

Locale
: The new locale you wish to add.

Supported Charsets
: Enter the supported character set for the specified locale. Character sets are delimited by a semicolon. For example, `charset=charset1;charset2;charset3;...;charsetn`

3 Click OK.

You are returned to the Globalization Settings page.

▼ To Edit Supported Character Sets

1 Click Edit next to the name of the character set you want to modify.

2 Modify the following parameters:

Locale
: The new locale you wish to add. For more information, see Supported Language Locales.

Supported Charsets
: Enter the supported character set for the specified locale. Character sets are delimited by a semicolon. For example, `charset=charset1;charset2;charset3;...;charsetn`

3 **Click OK.**

You are returned to the Globalization Settings page.

▼ To Add New Character Set Aliases

1 **Click Add under Charset Aliases.**

The New Charset Aliases page is displayed.

2 **Define the following attributes.**

MIME name	The IANA mapping name. For example, `Shift_JIS`
Java Name	The Java character set to map to the IANA character set.

3 **Click OK.**

You are returned to the Globalization Settings page.

▼ To Edit Existing Character Set Aliases

1 **Click Edit next to the name of the character set alias you want to modify.**

2 **Modify the following parameters:**

MIME name	The IANA mapping name. For example, `Shift_JIS`
Java Name	The Java character set to map to the IANA character set.

3 **Click OK.**

You are returned to the Globalization Settings page.

System Properties

The System properties include settings that affect the deployment of Federation Manager. This includes:

- “Logging” on page 77
- “Naming” on page 80
- “Platform” on page 83
- “Session” on page 84

Logging

The Logging service provides status and error messages related to Federation Manager administration. An administrator can configures values such as log file size and log file location. Federation Manager can record events in flat text files or in a relational database. The Logging service attributes are global attributes. The attributes are:

Maximum Log Size

This attribute accepts a value for the maximum size (in bytes) of a Federation Manager log file. The default value is 1000000.

Number of History Files

This attribute has a value equal to the number of backup log files that will be retained for historical analysis. Any integer can be entered depending on the partition size and available disk space of the local system. The default value is 3.

Note – Entering a value of 0 is interpreted to be the same as a value of 1, meaning that if you specify 0, a history log file will be created.

The value in this attribute is only used when the Logging Type attribute is set to `FILE`. If Logging Type is set to `DB` (Database), there are no history files.

Log File Location

The file-based logging function needs a location where log files can be stored. This field accepts a full directory path to that location. The default location is `/var/opt/SUNWam/fm/logs`.

If a non-default directory is specified, Federation Manager will create the directory if it does not exist. You should then set the appropriate permissions for that directory (for example, `0700`).

When configuring the log location for database logging (such as, Oracle or MySQL), part of the log location is case sensitive. For example, if you are logging to an Oracle database, the log location should be (note case sensitivity):

`jdbc:oracle:thin:@`*machine.domain*`:`*port*`:`*DBName*

To configure logging to a database, add the JDBC driver files to the web container's Java Virtual Machine (JVM) classpath. You need to manually add JDBC driver files to the classpath of the `amadmin` script or `amadmin` logging can not load the JDBC driver.

Note – Changes to logging attributes usually take effect after you save them. This does not require you to restart the server. If you are changing to secure logging, however, you should restart the server.

Logging Type

Enables you to specify either File, for flat file logging, or DB for database logging.

Caution – If either of the following attributes (Database User Name or Database User Password) is invalid, it will seriously affect Federation Manager processing. If Federation Manager or the Federation Manager Console becomes unstable, set the `com.iplanet.am.logstatus` property in `AMConfig.properties` to `INACTIVE`.

After setting the property, restart the server, log in to the console and reset the invalid attribute. Then, change the value of the *logstatus* property back to `ACTIVE` and restart the server.

Database User Name

This attribute accepts the name of the user that will connect to the database when the Logging Type attribute is set to DB.

Database User Password

This attribute accepts the database user password when the Logging Type attribute is set to DB.

Database User Password (confirm)

Confirm the database password.

Database Driver Name

This attribute enables you to specify the driver used for the logging implementation class.

Configurable Log Fields

Represents the list of fields that are to be logged. By default, all of the fields are logged. The fields are:

- DOMAIN
- HOSTNAME
- IPADDR
- LOGGED BY
- LOGLEVEL
- LOGINID
- MODULENAME

At minimum you should log CONTEXTID, DOMAIN, HOSTNAME, LOGINID and MESSAGEID.

Maximum Number of Records

This attribute sets the maximum number of records that the Java `LogReader` interface returns, regardless of how many records match the read query. By default, it is set to 500. This attribute can be overridden by the caller of the Logging API through the `LogQuery` class.

Number of Files per Archive

This attribute is only applicable to secure logging. It specifies when the log files and keystore need to be archived, and the secure keystore regenerated, for subsequent secure logging. The default is five files per logger.

Buffer Size

This attribute specifies the maximum number of log records to be buffered in memory before the logging service attempts to write them to the logging repository. The default is one record.

DB Failure Memory Buffer Size

This attribute defines the maximum number of log records held in memory if database logging fails. This attribute is only applicable when DB is specified as the Logging Type. When the

Logging Service loses connection to the database, it will buffer up to the number of records specified. This attribute defaults to two times of the value defined in the Buffer Size attribute.

Buffer Time

This attribute defines the amount of time that the log records will buffered in memory before they are sent to the logging service to be logged. This attribute applies if Enable Time Buffering is ON. The default is 3600 seconds.

Enable Time Buffering

When selected as ON, Federation Manager will set a time limit for log records to be buffered in memory. The amount of time is set in the Buffer Time attribute.

Naming

The Naming service is used to get and set URLs, plug-ins and configurations as well as request notifications for various other Federation Manager services such as session, authentication, logging, SAML and Federation. This service enables clients to find the correct service URL if the platform is running more than one instance of Federation Manager. When a naming URL is found, the naming service will decode the session of the user and dynamically replace the protocol, host, and port with the parameters from the session. This ensures that the URL returned for the service is for the host that the user session was created on. The Naming attributes are:

- "Profile Service URL" on page 80
- "Session Service URL" on page 81
- "Logging Service URL" on page 81
- "Policy Service URL" on page 81
- "Authentication Service URL" on page 81
- "SAML Web Profile/Artifact Service URL" on page 81
- "SAML SOAP Service URL" on page 81
- "SAML Web Profile/POST Service URL" on page 82
- "SAML Assertion Manager Service URL" on page 82
- "Federation Assertion Manager Service URL" on page 82
- "Security Token Manager URL" on page 82
- "JAXRPC Endpoint URL" on page 82

Profile Service URL

This field takes a value equal to *protocol*://*host*:*port*/*Server_DEPLOY_URI*/`profileservice`.

This syntax allows for dynamic substitution of the profile URL based on the server host, port number, and deployment URI.

Session Service URL

This field takes a value equal to *protocol*://*host*:*port*/*Server_DEPLOY_URI*/`sessionservice`.

This syntax allows for dynamic substitution of the session URL based on the server host, port number, and deployment URI.

Logging Service URL

This field takes a value equal to *protocol*://*host*:*port*/*Server_DEPLOY_URI*/`loggingservice`

This syntax allows for dynamic substitution of the logging URL based on the server host, port number, and deployment URI.

Policy Service URL

This field takes a value equal to *protocol*://*host*:*port*/*Server_DEPLOY_URI*/`policyservice`.

This syntax allows for dynamic substitution of the policy URL based on the server host, port number, and deployment URI.

Authentication Service URL

This field takes a value equal to *protocol*://*host*:*port*/*Server_DEPLOY_URI*/`authservice`.

This syntax allows for dynamic substitution of the authentication URL based on the server host, port number, and deployment URI.

SAML Web Profile/Artifact Service URL

This field takes a value equal to
protocol://*host*:*port*/*Server_DEPLOY_URI*/`SAMLAwareServlet`.

This syntax allows for dynamic substitution of the SAML web profile/artifact URL based on the server host, port number, and deployment URI.

SAML SOAP Service URL

This field takes a value equal to
protocol://*host*:*port*/*Server_DEPLOY_URI*/`SAMLSOAPReceiver`.

This syntax allows for dynamic substitution of the SAML SOAP URL based on the server host, port number, and deployment URI.

SAML Web Profile/POST Service URL

This field takes a value equal to *protocol*://*host*:*port*/*Server_DEPLOY_URI*/SAMLPOSTProfileServlet.

This syntax allows for dynamic substitution of the SAML web profile/POST URL based on the server host, port number, and deployment URI.

SAML Assertion Manager Service URL

This field takes a value equal to *protocol*://*host*:*port*/*Server_DEPLOY_URI*/AssertionManagerServlet/AssertionManagerIF.

This syntax allows for dynamic substitution of the SAML Assertion Manager Service URL based on the server host, port number, and deployment URI.

Federation Assertion Manager Service URL

This field takes a value equal to *protocol*://*host*:*port*/*Server_DEPLOY_URI*/FSAssertionManagerServlet/FSAssertionManagerIF.

This syntax allows for dynamic substitution of the Federation Assertion Manager Service URL based on the server host, port number, and deployment URI.

Security Token Manager URL

This field takes a value equal to *protocol*://*host*:*port*/*Server_DEPLOY_URI*/SecurityTokenManagerServlet/SecurityTokenManagerIF/.

This syntax allows for dynamic substitution of the Security Token Manager URL based on the server host, port number, and deployment URI.

JAXRPC Endpoint URL

This field takes a value equal to *protocol*://*host*:*port*/*Server_DEPLOY_URI*/jaxrpc/.

This syntax allows for dynamic substitution of the JAXRPC Endpoint URL based on the server host, port number, and deployment URI.

Platform

The Platform service is where additional servers can be added to the Federation Manager configuration as well as other options applied at the top level of the application. The Platform service attributes are global attributes. The attributes are:

Server List

This list contains the Federation Manager server instances. If the host specified in a request for a service URL is not in this list, the request is rejected.

Platform Locale

The platform locale value is the default language subtype that Federation Manager was installed with. The authentication, logging and administration services are administered in the language of this value. The default is `en_US`.

Cookie Domains

The list of domains that will be returned in the cookie header when setting a cookie to the user's browser during authentication. If empty, no cookie domain will be set. In other words, the Federation Manager session cookie will only be forwarded to the Federation Manager itself and to no other servers in the domain.

If SSO is required with other servers in the domain, this attribute must be set with the cookie domain. If you had two interfaces in different domains on one Federation Manager then you would need to set both cookie domains in this attribute. If a load balancer is used, the cookie domain must be that of the load balancer's domain, not the servers behind the load balancer. The default value for this field is the domain of the installed Federation Manager.

Login Service URL

This field specifies the URL of the login page. The default value for this attribute is */Service_DEPLOY_URI*`/UI/Login`.

Logout Service URL

This field specifies the URL of the logout page. The default value for this attribute is */Service_DEPLOY_URI*/UI/Logout.

Available Locales

This attribute stores all available locales configured for the platform. Consider an application that lets the user choose the user's locale. This application would get this attribute from the platform profile and present the list of locales to the user. The user would choose a locale and the application would set this in the user entry *preferredLocale*.

Client Character Sets

This attribute specifies the character set for different clients at the platform level. It contains a list of client types and the corresponding character sets.

Note – This attribute is not currently supported.

Session

Sessions module provides a solution for viewing user session information and managing user sessions. It keeps track of various session times as well as allowing the administrator to invalidate a session. The Session attributes are:

- "Maximum Number of Search Results" on page 84
- "Timeout For Search (Seconds)" on page 84
- "Max Session Time (Minutes)" on page 85
- "Max Idle Time (Minutes)" on page 85
- "Max Caching Time (Minutes)" on page 85

Maximum Number of Search Results

This attribute specifies the maximum number of results returned by a session search. The default value is 120.

Timeout For Search (Seconds)

This attributed defines the maximum amount of time before a session search terminates. The default value is 5 seconds.

Max Session Time (Minutes)

This attribute accepts a value in minutes to express the maximum time before the session expires and the user must reauthenticate to regain access. A value of 1 or higher will be accepted. The default value is 120. Max Session Time limits the validity of the session. It does not get extended beyond the configured value.

Tip – To balance the requirements of security and convenience, consider setting the Max Session Time interval to a higher value and setting the Max Idle Time interval to a relatively low value.

Max Idle Time (Minutes)

This attribute accepts a value (in minutes) equal to the maximum amount of time without activity before a session expires and the user must reauthenticate to regain access. A value of 1 or higher will be accepted. The default value is 30.

Tip – To balance the requirements of security and convenience, consider setting the Max Session Time interval to a higher value and setting the Max Idle Time interval to a relatively low value.

Max Caching Time (Minutes)

This attribute accepts a value (in minutes) equal to the maximum interval before the client contacts Federation Manager to refresh cached session information. A value of 0 or higher will be accepted. The default value is 3. It is recommended that the maximum caching time should always be less than the maximum idle time.

◆ ◆ ◆ CHAPTER 6

Authentication

Federation Manager is installed with a set of default authentication and service types. An authentication type instance is a plug-in that collects user information, checks the information against entries in a database, and allows or denies access to the user based on the information given. Multiple instances of the same type can be created and configured separately. A service type contains configurable information concerning the organization. The Authentication tab contains a listing of services added to the root organization. The default services are configured during installation. Additional services can be added after installation.

This chapter contains the following topics:

Customizing Authentication

This section contains tasks describing how you can reconfigure the default authentication method in certain situations. For example, you can enable your organization to get user authentication data from an LDAPv3 directory as opposed to the default flat file. The procedures include:

Configuring an Authentication Module for an Organization

This procedure describes how to configure an authentication module for the organization using the Federation Manager Console.

▼ To Configure an Authentication Module for an Organization

1. **In the Federation Manager Console, select the Organization tab.**
2. **Under Organization, select the Authentication tab.**
3. **Click Add.**

   A list of Authentication Modules is displayed.

   - Active Directory
   - Anonymous
   - Certificate
   - HTTP Basic
   - JDBC
   - LDAP
   - Membership
   - MSISDN
   - Password Playback
   - RADIUS
   - SafeWord
   - SecurID
   - Windows Desktop SSO
   - Windows NT

4. **Select a module from the list and click Next.**
5. **Configure the attributes for the authentication module.**
6. **Click Assign.**

Changing the Default Authentication Module from Flat File to LDAP

By default, users are authenticated by Federation Manager using the flat file with which it is deployed. The authentication component can be reconfigured to retrieve data from most LDAPv3–compliant directory (including Sun Java System Directory Server) rather than the default flat file.

Note – Although Microsoft Active Directory is an LDAPv3–compliant directory, the procedure has some differences. For more information, see "Changing the Default Authentication Module from Flat File to Active Directory" on page 91.

This section includes the following procedures:

- "To Set LDAP as the Default Authentication Module for an Organization" on page 89
- "To Enable an Organization to Use the LDAP Authentication Module" on page 90

▼ To Set LDAP as the Default Authentication Module for an Organization

1 Use the following template, modify *ROOT SUFFIX* to reflect that of the organization.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
    Copyright (c) 2005 Sun Microsystems, Inc. All rights reserved
    Use is subject to license terms.
-->

<!DOCTYPE Requests
    PUBLIC "-//iPlanet//Sun Java System Access Manager 2005Q4 Admin CLI DTD//EN"
    "jar://com/iplanet/am/admin/cli/amAdmin.dtd"

<!--  CREATE REQUESTS -->

<Requests>
<OrganizationRequests DN="ROOT SUFFIX">
   <ModifyServiceTemplate serviceName="iPlanetAMAuthService"
    schemaType="Organization">
     <AttributeValuePair>
       <Attribute name="iplanet-am-auth-org-config" />
         <Value>&lt;AttributeValuePair&gt;&lt;Value&gt;com.sun.identity.
          authentication.modules.ldap.LDAP REQUIRED&lt;/Value&gt;&lt;
          /AttributeValuePair&gt;</Value>
     </AttributeValuePair>
   </ModifyServiceTemplate>
</OrganizationRequests>
</Requests>
```

2 **Modify the following template to change the administrator's default authentication module to LDAP.**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
    Copyright (c) 2005 Sun Microsystems, Inc. All rights reserved
    Use is subject to license terms.
-->

<!DOCTYPE Requests
    PUBLIC "-//iPlanet//Sun Java System Access Manager 2005Q4 Admin CLI DTD//EN"
    "jar://com/iplanet/am/admin/cli/amAdmin.dtd">

<!--  CREATE REQUESTS -->

<Requests>
<OrganizationRequests DN="ROOT SUFFIX">
   <ModifyServiceTemplate serviceName="iPlanetAMAuthService"
    schemaType="Organization">
     <AttributeValuePair>
       <Attribute name="iplanet-am-auth-admin-auth-module" />
         <Value>
           &lt;AttributeValuePair&gt;&lt;Value&gt;com.sun.identity.
           authentication.modules.ldap.LDAP REQUIRED&lt;/Value&gt;&lt;
           /AttributeValuePair&gt;
         </Value>
     </AttributeValuePair>
   </ModifyServiceTemplate>
</OrganizationRequests>
</Requests>
```

3 **Load the modified XML files using** `amadmin` **and the format** */FederationManager-base/*`fm/bin/amadmin –u amadmin –w` *password* `–i` *war—staging—directory* `-t` *name-of-XML-file*.

▼ To Enable an Organization to Use the LDAP Authentication Module

1 **In the Federation Manager Console, select the Organization tab.**

2 **Under Organization, select the Authentication tab.**

3 **Click Add.**

A list of Authentication Modules is displayed.

4 **Select LDAP from the list and click Next.**

5 **Configure the attributes for the LDAP authentication module and click Assign.**

6 **Under Organization, select the Authentication tab.**

7 **Click the Edit button next to the Core authentication service.**

The Core attributes are displayed.

8 **Add LDAP to the Organization Authentication Modules attribute by holding down the Control key and selecting LDAP.**

9 **Click Save.**

LDAP is now enabled as an authentication module for the organization. To authenticate to Federation Manager through the LDAP module, use a URL in the format *protocol*`://`*host*`:`*port*`/`*deploy_URI*`/`*something*`?module=LDAP`.

Changing the Default Authentication Module from Flat File to Active Directory

Although Microsoft Active Directory is an LDAPv3–compliant directory, the procedure to change the default authentication module from Flat File to Active Directory is different from the procedure described in "Changing the Default Authentication Module from Flat File to LDAP" on page 89. The following sections describe the procedures:

- "To Set Active Directory as the Default Authentication Module for an Organization" on page 91
- "To Enable an Organization to Use the Active Directory Authentication Module" on page 92

▼ To Set Active Directory as the Default Authentication Module for an Organization

1 **Use `ldapsearch` in the following format to find values that begin with `iplanet-am-auth-org-config`.**

```
/usr/bin/ldapsearch -b OU=default,OU=OrganizationConfig,OU=1.0,
OU=iPlanetAMAuthService,OU=services,ROOTSUFFIX
-D admin-dn -w admin-password
-s base -h AD-host -p AD-port
"(objectclass=*)" sunkeyvalue
```

The search result would look like this:

```
sunkeyvalue=iplanet-am-auth-org-config=<AttributeValuePair>
<Value>com.sun.identity.authentication.modules.flatfile.
FlatFileREQUIRED</Value></AttributeValuePair>
```

Save the search result as it will be used in the following step as the value for the Delete entry.

2. **Save the following text as an Lightweight Directory Interchange Format (LDIF) file.**

```
dn: OU=default,OU=OrganizationConfig,OU=1.0,
OU=iPlanetAMAuthService,OU=services,ROOTSUFFIX
changetype:modify
delete:sunkeyvalue
sunkeyvalue:
iplanet-am-auth-org-config=<AttributeValuePair>
<Value>com.sun.identity.authentication.modules.
flatfile.FlatFileREQUIRED</Value>
</AttributeValuePair>

dn: OU=default,OU=OrganizationConfig,OU=1.0,
OU=iPlanetAMAuthService,OU=services,ROOTSUFFIX
changetype:modify
add:sunkeyvalue
sunkeyvalue:
iplanet-am-auth-org-config=<AttributeValuePair>
<Value>com.sun.identity.authentication.modules.
ldap.LDAPREQUIRED</Value></AttributeValuePair>
```

3. **Type the found values from the previous step into the Delete section of the saved LDIF file.**

4. **Type the new values into the Add section of the saved LDIF file.**

5. **Run `ldapmodify` using the LDIF file as input.**

   `/usr/bin/ldapmodify -h` *AD-host* `-p` *AD-port* `-D` *adminDN* `-w` *admin-password* `-f` *name-of-LDIF-file*

▼ To Enable an Organization to Use the Active Directory Authentication Module

1. **In the Federation Manager Console, select the Organization tab.**

2. **Under Organization, select the Authentication tab.**

3. **Click Add.**

   A list of Authentication Modules is displayed.

4. **Select Active Directory from the list and click Next.**

5. **Configure the attributes for the Active Directory authentication module and click Assign.**

6. **Under Organization, select the Authentication tab.**

7 **Click the Edit button next to the Core authentication service.**

The Core attributes are displayed.

8 **Add Active Directory to the Organization Authentication Modules attribute by holding down the Control key and selecting Active Directory.**

9 **Click Save.**

Active Directory is now enabled as an authentication module for the organization. To authenticate to Federation Manager through the Active Directory module, use a URL in the format *protocol*`://`*host*`:`*port*/*deploy_URI*/*something*`?module=AD`.

Changing the Default Administrator Authentication Module from a Flat File to LDAP

After you have configured and enabled Federation Manager to retrieve data from an LDAPv3–compliant data store, you must also change the default method of authentication for the Federation Manager administrator from the flat file to the LDAPv3–compliant directory. This section contains the following procedures:

- "To Change the Default Administrator Authentication Module from Flat File to Sun Java System Directory Server" on page 93
- "To Change the Default Administrator Authentication Module from Flat File to Microsoft Active Directory" on page 95

Note – Make sure the super user (by default, `amadmin`) has read, write and search permission to the `ou=services` branch of the directory information tree (DIT).

▼ To Change the Default Administrator Authentication Module from Flat File to Sun Java System Directory Server

1 **Use the following template, modify *ROOT SUFFIX* to reflect that of the organization.**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
    Copyright (c) 2005 Sun Microsystems, Inc. All rights reserved
    Use is subject to license terms.
-->

<!DOCTYPE Requests
    PUBLIC "-//iPlanet//Sun Java System Access Manager 2005Q4 Admin CLI DTD//EN"
    "jar://com/iplanet/am/admin/cli/amAdmin.dtd"

<!--  CREATE REQUESTS -->
```

```
<Requests>
<OrganizationRequests DN="ROOT SUFFIX">
   <ModifyServiceTemplate serviceName="iPlanetAMAuthService"
    schemaType="Organization">
     <AttributeValuePair>
       <Attribute name="iplanet-am-auth-admin-auth-module" />
         <Value>&lt;AttributeValuePair&gt;&lt;Value&gt;com.sun.
          identity.authentication.modules.ldap.LDAPREQUIRED&lt;
          Value&gt;&lt;AttributeValuePair&gt;</Value>
     </AttributeValuePair>
   </ModifyServiceTemplate>
</OrganizationRequests>
</Requests>
```

2. **Load the modified XML file using `amadmin` and the format**
   FederationManager-base`/fm/bin/amadmin –u amadmin –w` *password* `–i` *war—staging—directory* `-t` *name-of-XML-file*.

3. **Make the following changes in the `AMConfig.properties` file.**

   `AMConfig.properties` is located in `/FederationManager-base/web-src/WEB-INF/`. Multiple entries are pipe-separated.

   a. **Add the distinguished name (DN) of the administrator to the `com.sun.identity.authentication.special.users` property.**

   b. **(Optional) Change the DN of the administrator in the `com.sun.identity.authentication.super.user` property.**

      This step is included if you want to use an administrator who already exists in your directory and not the default Federation Manager administrator `amadmin`. To use the default `amadmin`, create the user in Directory Server and make sure the user is given read, write and search permissions to the `ou=services` branch of the directory information tree (DIT).

   c. **Change the value of the `admin.auth.classname` property to `com.sun.identity.authentication.internal.server.LocalLdapAuthModule`.**

4. **In the `serverconfig.xml` file, change the value of the server group `internalauthentication` property to reflect the administrator DN, password and correct base DN.**

   `serverconfig.xml` is located in `/FederationManager-base/web-src/WEB-INF/`.

5. **Regenerate and redeploy the WAR.**

6. **Restart the web container, if applicable.**

To Change the Default Administrator Authentication Module from Flat File to Microsoft Active Directory

1. **Use `ldapsearch` in the following format to find values in Active Directory that begin with `iplanet-am-auth-admin-auth-module`.**

   ```
   /usr/bin/ldapsearch -b OU=default,OU=OrganizationConfig,OU=1.0,
   OU=iPlanetAMAuthService,OU=services,ROOTSUFFIX
   -D admin-dn -w admin-password
   -s base -h AD-host -p AD-port
   "(objectclass=*)" sunkeyvalue
   ```

   The search result would look like this:

   ```
   sunkeyvalue=iplanet-am-auth-admin-auth-module=
   <AttributeValuePair><Value>com.sun.identity.authentication.
   modules.flatfile.FlatFileREQUIRED</Value></AttributeValuePair>
   ```

   Save the search result as it will be used in the following step as the value for the Delete entry.

2. **Save the following text as a Lightweight Directory Interchange Format (LDIF) file.**

   ```
   dn: OU=default,OU=OrganizationConfig,OU=1.0,
   OU=iPlanetAMAuthService,OU=services,<ROOTSUFFIX>
   changetype:modify
   delete:sunkeyvalue
   sunkeyvalue:
   iplanet-am-auth-admin-auth-module=<AttributeValuePair>
   <Value>com.sun.identity.authentication.modules.
   flatfile.FlatFileREQUIRED</Value>
   </AttributeValuePair>

   dn: OU=default,OU=OrganizationConfig,OU=1.0,
   OU=iPlanetAMAuthService,OU=services,<ROOTSUFFIX>
   changetype:modify
   add:sunkeyvalue
   sunkeyvalue:
   iplanet-am-auth-admin-auth-module=<AttributeValuePair>
   <Value>com.sun.identity.authentication.modules.
   ldap.LDAPREQUIRED</Value></AttributeValuePair>
   ```

3. **Type the found values from the previous step into the Delete section of the saved LDIF file.**

4. **Type the new values into the Add section of the saved LDIF file.**

5. **Run `ldapmodify` using the LDIF file as input.**

   `/usr/bin/ldapmodify -h` *AD-host* `-p` *AD-port* `-D` *adminDN* `-w` *admin-password* `-f` *name-of-LDIF-file*

6. **Make the following changes in the `AMConfig.properties` file.**

   `AMConfig.properties` is located in `/FederationManager-base/web-src/WEB-INF/`. Multiple entries are pipe-separated.

   a. **Add the distinguished name (DN) of the administrator to the `com.sun.identity.authentication.special.users` property.**

   b. **(Optional) Change the DN of the administrator in the `com.sun.identity.authentication.super.user` property.**

      This step is included if you want to use an administrator who already exists in your directory and not the default Federation Manager administrator `amadmin`. To use the default `amadmin`, create the user in Directory Server and make sure the user is given read, write and search permissions to the `ou=services` branch of the directory information tree (DIT).

   c. **Change the value of the `admin.auth.classname` property to `com.sun.identity.authentication.internal.server.LocalLdapAuthModule`.**

7. **In the `serverconfig.xml` file, change the value of the server group `internalauthentication` property to reflect the administrator DN, password and correct baseDN.**

   `serverconfig.xml` is located in `/FederationManager-base/web-src/WEB-INF/`.

8. **Regenerate and redeploy the WAR.**

9. **Restart the web container, if applicable.**

Default Authentication Services

Default services are configured during installation of Federation Manager. They include:

Access Control

Access Control properties define whether a particular user has read or write permission for the service configuration. The Access Control attributes are:

Users with Write Permission

Specifies a list of user IDs who have write permission on the service configuration. Write permission implies read permission. Typing * as a value means all users have write permission. `amadmin` has write permission, by default.

Users with Read Permission

Specifies a list of user IDs who have read permission on the service configuration. Typing * as a value means all users have write permission.

Core

This module is the general configuration base for the Federation Manager authentication services. It must be registered and configured to use any of the specific authentication module instances. It enables the administrator to define default values that will be picked up for the values that are not specifically set in the Federation Manager default authentication modules. The attributes are:

Pluggable Authentication Module Classes

Specifies the Java classes of the authentication modules available to the organization configured within the Federation Manager platform. You can write custom authentication modules by implementing the `AMLoginModule` SPI or the `JAAS LoginModule` SPI. To define new services, this field must take a text string specifying the full class name (including package name) of each new authentication service.

Supported Authentication Module for Clients

Specifies a list of supported authentication modules for a specific client. Use the format `clientType | module1,module2,module3`. This attribute is in effect when Client Detection is enabled.

Note – This attribute is not currently supported.

LDAP Connection Pool Size

Specifies the minimum and maximum connection pool to be used on a specific LDAP server and port. This attribute is for LDAP and Membership authentication services only. Use the format *host*:*port*:*min*:*max*.

Note – This connection pool is different from the SDK connection pool configured in `serverconfig.xml`.

Default LDAP Connection Pool Size

Sets the default minimum and maximum connection pool to be used with all LDAP authentication module configurations. If an entry for the host and port exists in the LDAP Connection Pool Size attribute, the minimum and maximum settings will not be used from LDAP Connection Default Pool Size.

Administrator Authentication Configuration

Defines the authentication service for administrators only. This attribute can be used if the authentication module for administrators needs to be different from the module for end users. The modules configured in this attribute are picked up when the Federation Manager console is accessed. For example, `http://`*host*`.`*port*`/`*console_deploy_uri*.

User Profile Dynamic Creation Default Roles

This field specifies the roles assigned to a new user whose profiles are created if Dynamic Creation is selected through User Profile. There is no default value. The administrator must specify the DNs of the roles that will be assigned to the new user. The role specified must be

under the organization for which authentication is being configured. It cannot be a filtered role. Also, if you wish to automatically assign specific services to a user, you have to configure a Required Services type attribute in the user's profile.

Note – This attribute is not currently supported.

Enable Persistent Cookie Mode

This option determines whether users can restart the browser and still return to their authenticated session. User sessions can be retained by enabling Enable Persistent Cookie Mode. When Enable Persistent Cookie Mode is enabled, a user session does not expire until its persistent cookie expires, or the user explicitly logs out. The expiration time is specified in Persistent Cookie Maximum Time. The default value is that Persistent Cookie Mode is not enabled and the authentication service uses only memory cookies.

Note – A persistent cookie must be explicitly requested by the client using the *iPSPCookie=yes* parameter in the login URL.

Persistent Cookie Maximum Time

Specifies the interval after which a persistent cookie expires. The interval begins when the user's session is successfully authenticated. The default value is 2147483 (time in seconds). The field will accept any integer value less than the default.

Alias Search Attribute Name

After successful authentication by a user, the user's profile is retrieved. This field specifies a second LDAP attribute to search from if a search on the first LDAP attribute fails to locate a matching user profile. Primarily, this attribute will be used when the user identification returned from an authentication module is not the same as that specified in User Naming Attribute. For example, a RADIUS server might return `abc1234` but the user name is `abc`. There is no default value for this attribute. The field will take any valid LDAP attribute (for example, `cn`).

Note – This attribute is not currently supported.

Default Authentication Locale

Specifies the default language subtype to be used by the authentication service. The default value is `en_US`. The following table contains a listing of the supported language locales.

TABLE 6–1 Supported Language Locales

Language Tag	Language Name
af	Afrikaans
be	Byelorussian
bg	Bulgarian
ca	Catalan
cs	Czechoslovakian
da	Danish
de	German
el	Greek
en	English
es	Spanish
eu	Basque
fl	Finnish
fo	Faroese
fr	French
ga	Irish
gl	Galician
hr	Croatian
hu	Hungarian
id	Indonesian
is	Icelandic
it	Italian
ja	Japanese
ko	Korean
nl	Dutch
no	Norwegian
pl	Polish
pt	Portuguese

TABLE 6–1 Supported Language Locales *(Continued)*

Language Tag	Language Name
ro	Romanian
ru	Russian
sk	Slovakian
sl	Slovenian
sq	Albanian
sr	Serbian
sv	Swedish
tr	Turkish
uk	Ukranian
zh	Chinese

In order to use a different locale, all authentication templates for that locale must first be created. A new directory must then be created for these templates.

Organization Authentication Configuration

Sets the authentication module for the organization. The default authentication module is LDAP.

Enable Login Failure Lockout Mode

Specifies whether a user can attempt a second authentication if the first attempt failed. Selecting this attribute enables a lockout and the user will have only one chance at authentication. By default, the lockout feature is not enabled. This attribute works in conjunction with Lockout-related and notification attributes.

Login Failure Lockout Count

Defines the number of attempts that a user may try to authenticate, within the time interval defined in Login Failure Lockout Interval, before being locked out.

Login Failure Lockout Interval

Defines (in minutes) the time between two failed login attempts. If a login fails and is followed by another failed login that occurs within the lockout interval, then the lockout count is incremented. Otherwise, the lockout count is reset.

Email Address to Send Lockout Notification

Specifies an email address that will receive notification if a user lockout occurs. To send email notification to multiple addresses, separate each email address with a space. For non-English locales, the format is *email_address*|*locale*|*charset*.

Warn User After N Failures

Specifies the number of authentication failures that can occur before a warning message is sent that the user will be locked out.

Login Failure Lockout Duration

Enables memory locking. By default, the lockout mechanism will inactivate the User Profile (after a login failure) defined in *Lockout Attribute Name*. If the value of Login Failure Lockout Duration is greater than 0, then its memory locking and the user account will be locked for the number of minutes specified.

Lockout Attribute Name

Designates any LDAP attribute that is to be set for lockout. The value in Lockout Attribute Value must also be changed to enable lockout for this attribute name. By default, Lockout Attribute Name is empty. The default implementation values are *inetuserstatus* (LDAP attribute) and *inactive* when the user is locked out and Login Failure Lockout Duration is set to 0.

Lockout Attribute Value

This attribute specifies whether lockout is enabled or disabled for the attribute defined in Lockout Attribute Name. By default, the value is set to inactive for *inetuserstatus*.

Default Success URL

This field accepts a list of multiple values that specify the URL to which users are redirected after successful authentication. The format of this attribute is `clientType|URL`, although you can specify only the value of the URL which assumes a default type of HTML. The default value is */amserver/console*.

Default Failure Login URL

This field accepts a list of multiple values that specify the URL to which users are redirected after an unsuccessful authentication. The format of this attribute is `clientType|URL`, although you can specify only the value of the URL which assumes a default type of HTML.

Authentication Post Processing Class

Specifies the name of the Java class used to customize post authentication processes for successful or unsuccessful logins. The Java class must implement the `com.sun.identity.authentication.spi.AMPostAuthProcessInterface` interface.

Note – Additionally, you must add the path to the class in your web server's Java classpath attribute.

Enable Generate UserID Mode

This attribute is used by the Membership authentication module. If this attribute field is enabled, the Membership module is able to generate user IDs, during the Self Registration process, for a specific user if the user ID already exists. The user IDs are generated from the Java class specified in Pluggable User Name Generator Class.

Pluggable User Name Generator Class

Specifies the name of the Java class is used to generate User IDs when Enable Generate UserID Mode is used.

Default Authentication Level

The authentication level value indicates how much to trust authentications. Once a user has authenticated, this value is stored in the SSO token for the session. When the SSO token is presented to an application the user wants to access, the application can use the stored value to determine whether the level is sufficient to grant the user access. If the authentication level stored in an SSO token does not meet the minimum value required, the application can prompt the user to authenticate again through a service with a higher authentication level.

The authentication level should be set within the organization's specific authentication template. The Default Authentication Level value described here will apply only when no authentication level has been specified in the Authentication Level field for a specific organization's authentication template. The Default Authentication Level default value is 0. (The value in this attribute is not used by Federation Manager but by any external application that may chose to use it.)

Flat File

The Flat File authentication module enables authentication against a flat file. The default flat file repository stores user profile attributes as a properties file with using the format *attributename*=*attributevalue*. The attributes are:

Caution – The comma (,) is used as the delimiter for multiple values of the same attribute. When used for another purpose, commas must be encoded as `%2C` to avoid the flat file implementation to interpret the value as two.

Directory Location

Specifies the absolute path to the directory where all flat file users are located. The directory is used as a database of user IDs and passwords against which users can authenticate.

Authentication Level

The authentication level is set separately for each method of authentication. The value indicates how much to trust an authentication mechanism. Once a user has authenticated, this value is stored in the SSO token for the session. When the SSO token is presented to an application the user wants to access, the application uses the stored value to determine whether the level is sufficient to grant the user access. If the authentication level stored in an SSO token does not meet the minimum value required, the application can prompt the user to authenticate again through a service with a higher authentication level. The default value is 0.

Note – If no authentication level is specified, the SSO token stores the value specified in the Core Authentication attribute Default Authentication Level.

SAML

The Security Assertion Markup Language (SAML) authentication module receives and validates SAML Assertions on a target server.

Authentication Level

The authentication level is set separately for each method of authentication. The value indicates how much to trust an authentication mechanism. Once a user has authenticated, this value is stored in the SSO token for the session. When the SSO token is presented to an application the user wants to access, the application uses the stored value to determine whether the level is sufficient to grant the user access. If the authentication level stored in an SSO token does not meet the minimum value required, the application can prompt the user to authenticate again through a service with a higher authentication level. The default value is 0.

Note – If no authentication level is specified, the SSO token stores the value specified in the Core Authentication attribute Default Authentication Level.

Additional Authentication Services

Additional services can be configured after installation of Federation Manager. They include:

Active Directory

Active Directory authentication relies on the use of an LDAPv3–compliant server. This module type works similarly to the LDAP authentication module type, but uses the Microsoft Active Directory. Using this module type makes it possible for LDAP and Active Directory to coexist. The attributes are:

Primary Active Directory Server

Specifies the host name and port number of the primary Active Directory server specified during Federation Manager installation. This is the first server contacted for Active Directory authentication. The format is *hostname*:*port*. If there is no port number, assume 389. Multiple entries must be prefixed by the local server name.

If you have Federation Manager deployed with multiple domains, you can specify the communication link between specific instances of Federation Manager and Active Directory in the following format:

local-servername|*server*:*port local-servername2*|*server2*:*port2* ...

For example, if you have two instances deployed in different locations (L1-machine1-IS and L2-machine2-IS) communicating with different instances of Active Directory (L1-machine1-DS and L2-machine2-DS), it would look like the following:

```
L1-machine1-IS.example.com|L1-machine1-DS.example.com:389

L2-machine2-IS.example.com|L2-machine2-DS.example.com:389
```

Secondary Active Directory Server

Specifies the host name and port number of a secondary Active Directory server available to the Federation Manager platform. If the primary Active Directory server does not respond to a request for authentication, this server would then be contacted. If the primary server is up, Federation Manager will switch back to the primary server. The format is also *hostname*:*port*. Multiple entries must be prefixed by the local server name.

Caution – When authenticating users from a data store that is remote, it is important that both the Primary and Secondary LDAP Server Ports have values. The value for one data store location can be used for both fields.

DN To Start User Search

Specifies the DN of the node where the search for a user would start. (For performance reasons, this DN should be as specific as possible.) The default value is the root of the directory tree. Any valid DN will be recognized. If OBJECT is selected in the Search Scope attribute, the DN should specify one level above the level in which the profile exists. Multiple entries must be prefixed by the local server name. The format is *servername*|*search-dn*. For multiple entries:

servername1|*search-dn servername2*|*search-dn servername3*|*search-dn*

If multiple users are found for the same search, authentication will fail.

DN for Root User Bind

Specifies the DN of the user that will be used to bind to the Directory Server specified in the Primary LDAP Server and Port field as administrator. The authentication service needs to bind as this DN in order to search for a matching user DN based on the user login ID. The default is `amldapuser`. Any valid DN will be recognized.

Make sure that password is correct before you logout. If it is incorrect, you will be locked out. If this should occur, you can login with the super user DN in the *com.iplanet.authentication.super.user* property in the `AMConfig.Properties` file. By default, this is the `amAdmin` account with which you would normally log in, although you will use the full DN. For example:

```
uid_amAdmin,ou=People,dc=example,dc=com
```

Password for Root User Bind

Carries the password for the administrator profile specified in the DN for Root User Bind field. There is no default value. Only the administrator's valid Active Directory password is recognized.

Password for Root User Bind (confirm)

Confirm the password.

Attribute Used to Retrieve User Profile

Specifies the attribute used for the naming convention of user entries. By default, Federation Manager assumes that user entries are identified by the `uid` attribute. If your data store uses a different attribute (such as `givenname`) specify the attribute name in this field.

Attributes Used to Search for a User to be Authenticated

Lists the attributes to be used to form the search filter for a user that is to be authenticated, and allows the user to authenticate with more than one attribute in the user's entry. For example, if this field is set to *uid*, *employeenumber*, and *mail*, the user could authenticate with any of these names.

User Search Filter

Specifies an attribute to be used to find the user under the DN to Start User Search field. It works with the User Naming Attribute. There is no default value. Any valid user entry attribute will be recognized.

Search Scope

Indicates the number of levels in the Directory Server tree that will be searched for a matching user profile. The search begins from the node specified in DN to Start User Search. The default value is SUBTREE. One of the following choices can be selected:

OBJECT	Searches only the specified node.
ONELEVEL	Searches at the level of the specified node and one level down.
SUBTREE	Search all entries at and below the specified node.

Enable SSL Access To Active Directory Server

Enables SSL access to the Directory Server specified in the Primary and Secondary Server and Port field. By default, the box is not checked and the SSL protocol will not be used to access the Directory Server.

If the Active Directory server is running with SSL enabled (LDAPS), you must make sure that Federation Manager is configured with proper SSL trusted certificates so that it may connect using LDAPS protocol.

Return User DN to Authenticate

If enabled, this option allows the Active Directory authentication module instance to return the DN instead of the User ID, and no search is necessary. Normally, an authentication module instance returns only the User ID, and the authentication service searches for the user in the local instance. If an external Active Directory is used, this option is typically not enabled.

Active Directory Server Check Interval

This attribute is used for Active Directory Server failback. It defines the number of minutes in which a thread will *sleep* before verifying that the primary Active Directory server is running.

User Creation Attributes

This attribute is used by the Active Directory authentication module instance when the Active Directory server is configured as an external Active Directory server. It contains a mapping of attributes between a local and an external Directory Server. This attribute has the following format:

attr1|*externalattr1*

attr2|*externalattr2*

When this attribute is populated, the values of the external attributes are read from the external Directory Server and are set for the internal Directory Server attributes. The values of the

external attributes are set in the internal attributes only when the `User Profile`attribute (in the Core Authentication module type) is set to Dynamically Created and the user does not exist in local Directory Server instance. The newly created user will contain the values for internal attributes, as specified in User Creation Attributes List, with the external attribute values to which they map.

Authentication Level

The authentication level is set separately for each method of authentication. The value indicates how much to trust an authentication mechanism. Once a user has authenticated, this value is stored in the SSO token for the session. When the SSO token is presented to an application the user wants to access, the application uses the stored value to determine whether the level is sufficient to grant the user access. If the authentication level stored in an SSO token does not meet the minimum value required, the application can prompt the user to authenticate again through a service with a higher authentication level. The default value is 0.

Note – If no authentication level is specified, the SSO token stores the value specified in the Core Authentication attribute Default Authentication Level.

Anonymous

This module type allows a user to log in without specifying credentials. You can create an Anonymous user so that anyone can log in as Anonymous without having to provide a password. Anonymous connections are usually customized by the administrator so that Anonymous users have limited access to the server. The attributes are:

Valid Anonymous Users

Contains a list of user IDs that have permission to login without providing credentials. If a user's login name matches a user ID in this list, access is granted and the session is assigned to the specified user ID. If this list is empty, a user will be authenticated as the user defined in the Default Anonymous User Name attribute when accessing the following default module instance login URL:

protocol`://`*server-host.server-domain*`:`*server-port*`/`*deploy-uri*`/UI/Login?module=Anonymous&org=`*org_name*

If this list is not empty, Federation Manager will prompt the user to enter any valid Anonymous user name when accessing default module instance login URL (as above).

Note – If the Valid Anonymous Users list is not empty, the user can login without the login page by accessing the following URL:

protocol://*server-host.server-domain*:*server-port*/*deploy-uri*/UI/Login?module=Anonymous&org=*org_name*&IDToken1=*valid-Anonymous-username*

Default Anonymous User Name

Defines the user ID to which a session is assigned if the Valid Anonymous User List is empty and the following default module instance login URL is accessed:

protocol://*server-host.server-domain*:*server-port*/*deploy-uri*/UI/Login?module=Anonymous&org=*org_name*

The default value is anonymous.

Note – If the Valid Anonymous Users list is not empty, the user can login without the login page by using the user defined in Default Anonymous User Name. This can be done by accessing the following URL:

protocol://*server-host.server-domain*:*server-port*/*deploy-uri*/UI/Login?module=Anonymous&org=*org_name*&IDToken1=*DefaultAnonymousUserName*

Enable Case Sensitive User IDs

If enabled, this option allows for case-sensitivity for user IDs. By default, this attribute is not enabled.

Authentication Level

The authentication level is set separately for each method of authentication. The value indicates how much to trust an authentication mechanism. Once a user has authenticated, this value is stored in the SSO token for the session. When the SSO token is presented to an application the user wants to access, the application uses the stored value to determine whether the level is sufficient to grant the user access. If the authentication level stored in an SSO token does not meet the minimum value required, the application can prompt the user to authenticate again through a service with a higher authentication level. The default value is 0.

Note – If no authentication level is specified, the SSO token stores the value specified in the Core Authentication attribute Default Authentication Level.

Certificate

This module enables a user to log in through a personal digital certificate (PDC). The module instance can require the use of the Online Certificate Status Protocol (OCSP) to determine the state of a certificate. Use of the OCSP is optional. The user is granted or denied access to a resource based on whether or not the certificate is valid. The attributes are:

Match Certificate in LDAP

Specifies whether to check if the user certificate presented at login is stored in the LDAP Server. If no match is found, the user is denied access. If a match is found and no other validation is required, the user is granted access. The default is that the Certificate Authentication service does not check for the user certificate.

Note – A certificate stored in the Directory Server is not necessarily valid; it may be on the certificate revocation list. See Match Certificate to CRL. However, the web container may check the validity of the user certificate presented at login.

Subject DN Attribute Used to Search LDAP for Certificates

Specifies the attribute of the certificate's *SubjectDN* value that will be used to search LDAP for certificates. This attribute must uniquely identify a user entry. The actual value will be used for the search. The default is `cn`.

Match Certificate to CRL

Specifies whether to compare the user certificate against the Certificate Revocation List (CRL) in the LDAP Server. The CRL is located by one of the attribute names in the issuer's *SubjectDN*. If the certificate is on the CRL, the user is denied access; if not, the user is allowed to proceed. This attribute is, by default, not enabled.

Note – Certificates should be revoked when the owner of the certificate has changed status and no longer has the right to use the certificate or when the private key of a certificate owner has been compromised.

Issuer DN Attribute Used to Search LDAP for CRLs

Specifies the attribute of the received certificate's issuer *subjectDN* value that will be used to search LDAP for CRLs. This field is used only when the Match Certificate to CRL attribute is enabled. The actual value will be used for the search. The default is `cn`.

HTTP Parameters for CRL Update

Specifies the HTTP parameters for obtaining a CRL from a servlet for a CRL update. Contact the administrator of your CA for these parameters.

Enable OCSP Validation

Enables OCSP validation to be performed by contacting the corresponding OCSP responder. The OCSP responder is decided as follows during runtime:

- If `com.sun.identity.authentication.ocspCheck` is true and the OCSP responder is set in the *com.sun.identity.authentication.ocsp.repsonder.url* attribute, the value of the attribute will be used as the OCSP responder.
- If *com.sun.identity.authentication.ocspCheck* is set to true and If the value of the attribute is not set in the `AMConfig.properties` file, the OCSP responder presented in your client certificate is used as the OCSP responder.
- If *com.sun.identity.authentication.ocspCheck* is set to false or if *com.sum.identity.authentication.ocspCheck* is set to true and if an OCSP responder can not be found, no OCSP validation will be performed.

Before enabling OCSP Validation, make sure that the time of the Federation Manager machine and the OCSP responder machine are in sync as close as possible. Also, the time on the Federation Manager machine must not be behind the time on the OCSP responder. For example:

```
OCSP responder machine - 12:00:00 pm

Federation Manager machine - 12:00:30 pm
```

LDAP Server Where Certificates are Stored

Specifies the name and port number of the LDAP server where the certificates are stored. The default value is the host name and port specified when Federation Manager was installed. The host name and port of any LDAP Server where the certificates are stored can be used. The format is `hostname:port`.

LDAP Start Search DN

Specifies the DN of the node where the search for the user's certificate should start. There is no default value. The field will recognize any valid DN.

Multiple entries must be prefixed by the local server name. The format is as follows:

```
servername|search dn
```

For multiple entries:

```
servername1|search dn servername2|search dn servername3|search dn...
```

If multiple users are found for the same search, authentication will fail.

LDAP Server Principal User

This field accepts the DN of the principal user for the LDAP server where the certificates are stored. There is no default value for this field which will recognize any valid DN. The principal user must be authorized to read, and search certificate information stored in the Directory Server.

LDAP Server Principal Password

This field carries the LDAP password associated with the user specified in the LDAP Server Principal User field. There is no default value for this field which will recognize the valid LDAP password for the specified principal user. This value is stored as readable text in the directory.

LDAP Server Principal Password (confirm)

Confirm the password.

LDAP Attribute for Profile ID

Specifies the attribute in the Directory Server entry that matches the certificate whose value should be used to identify the correct user profile. There is no default value for this field which will recognize any valid attribute in a user entry (*cn*, *sn*, and so forth) that can be used as the UserID.

Use SSL for LDAP Access

Specifies whether to use SSL to access the LDAP server. The default is that the Certificate Authentication service does not use SSL for LDAP access.

Certificate Field Used to Access User Profile

Specifies which field in the certificate's Subject DN should be used to search for a matching user profile. For example, if you choose email address, the certificate authentication service will search for the user profile that matches the attribute *emailAddr* in the user certificate. The user logging in then uses the matched profile. The default field is *subject CN*. The list contains:

- email address
- subject CN
- subject DN
- subject UID
- other

Other Certificate Field Used to Access User Profile

If the value of the Certificate Field Used to Access User Profile attribute is set to `other`, then this field specifies the attribute that will be selected from the received certificate's *subjectDN* value. The authentication service will then search the user profile that matches the value of that attribute.

Trusted Remote Hosts

Defines a list of trusted hosts that can be trusted to send certificates to Federation Manager. Federation Manager must verify whether the certificate emanated from one of these hosts. This configuration only used with Sun Java System Portal Server. This attribute accepts the following values:

None
: Disables the attribute. This is set by default.

Any
: Accepts Portal Server Gateway-style certificate authentication from any client IP address.

IP ADDR
: Lists the IP addresses from which to accept Portal Server Gateway-style certificate authentication requests (the IP Address of the Gateway(s)).

SSL Port Number

Specifies the port number for the secure socket layer. Currently, this attribute is only used by the Gateway servlet. Before you add or change an SSL Port Number, see the *Sun Java System Access Manager 7 2005Q4 Administration Guide*.

Authentication Level

The authentication level is set separately for each method of authentication. The value indicates how much to trust an authentication mechanism. Once a user has authenticated, this value is stored in the SSO token for the session. When the SSO token is presented to an application the user wants to access, the application uses the stored value to determine whether the level is sufficient to grant the user access. If the authentication level stored in an SSO token does not meet the minimum value required, the application can prompt the user to authenticate again through a service with a higher authentication level. The default value is 0.

Note – If no authentication level is specified, the SSO token stores the value specified in the Core authentication attribute Default Authentication Level.

HTTP Basic

The HTTP authentication module allows login using the HTTP basic authentication with no data encryption. A user name and password are requested through the use of a web browser. Credentials are validated internally using the LDAP authentication module.

Note – HTTP Basic authentication relies on the use of an LDAPv3–compliant server.

The HTTP Basic attribute is:

- "Authentication Level" on page 115

Authentication Level

The authentication level is set separately for each method of authentication. The value indicates how much to trust an authentication mechanism. Once a user has authenticated, this value is stored in the SSO token for the session. When the SSO token is presented to an application the

user wants to access, the application uses the stored value to determine whether the level is sufficient to grant the user access. If the authentication level stored in an SSO token does not meet the minimum value required, the application can prompt the user to authenticate again through a service with a higher authentication level. The default value is 0.

Note – If no authentication level is specified, the SSO token stores the value specified in the Core Authentication attribute Default Authentication Level.

JDBC

The Java Database Connectivity (JDBC) authentication module allows Federation Manager to authenticate users through any Structured Query Language (SQL) databases that provide JDBC-enabled drivers. The connection to the SQL database can be either directly through a JDBC driver or through a JNDI connection pool. The attributes are:

Connection Type

Specifies the connection type to the SQL database, using either a JNDI (Java Naming and Directory Interface) connection pool or JDBC driver. The options are:

- Connection pool is retrieved via JDNI
- Non-persistent JDBC connection

The JNDI connection pool utilizes the configuration from the underlying web container.

Connection Pool JNDI Name

If JNDI is selected in Connection Type, this field specifies the connection pool name. Because JDBC authentication uses the JNDI connection pool provided by the web container, the setup of JNDI connection pool may not be consistent among other web containers. See the *Sun Java System Access Manager 7 2005Q4 Administration Guide* for examples.

JDBC Driver

If JDBC is selected in Connection Type, this field specifies the JDBC driver provided by the SQL database. For example, `com.mysql.jdbc.Driver`.

JDBC URL

Specifies the database URL if JDBC is select in Connection Type. For example, the URL for mySQL is `jdbc.mysql://`*hostname*`:`*port*`/`*databaseName*.

Connect This User to Database

Specifies the user name from whom the database connection is made for the JDBC connection.

Password for Connecting to Database

Defines the password for the user specified in User to Connect to Database.

Password for Connecting to Database Confirm

Confirm the password.

Password Column String

Specifies the password column name in the SQL database.

Prepared Statement

Specifies the SQL statement that retrieves the password of the user that is logging in. For example:

```
select Password from Employees where USERNAME = ?
```

Class to Transform Password Syntax

Specifies the class name that transforms the password retrieved from the database, to the format of the user input, for password comparison. This class must implement the `JDBCPasswordSyntaxTransform` interface.

Authentication Level

The authentication level is set separately for each method of authentication. The value indicates how much to trust an authentication mechanism. Once a user has authenticated, this value is stored in the SSO token for the session. When the SSO token is presented to an application the user wants to access, the application uses the stored value to determine whether the level is sufficient to grant the user access. If the authentication level stored in an SSO token does not meet the minimum value required, the application can prompt the user to authenticate again through a service with a higher authentication level. The default value is 0.

Note – If no authentication level is specified, the SSO token stores the value specified in the Core Authentication attribute Default Authentication Level.

LDAP

LDAP authentication relies on the use of an LDAPv3–compliant server. This module enables authentication using LDAP bind, a Directory Server operation which associates a user ID password with a particular LDAP entry. You can define multiple LDAP authentication configurations for an organization. The attributes are:

Primary LDAP Server

Specifies the host name and port number of the primary LDAP server specified during Federation Manager installation. This is the first server contacted for authentication. The format is*hostname*:*port*. If there is no port number, assume 389. Multiple entries must be prefixed by the local server name.

If you have Federation Manager deployed with multiple domains, you can specify the communication link between specific instances of Federation Manager and Directory Server in the following format:

local-servername|*server*:*port local-servername2*|*server2*:*port2* ...

For example, if you have two Federation Manager instances deployed in different locations (L1-machine1-IS and L2- machine2-IS) communicating with different instances of Directory Server (L1-machine1-DS and L2-machine2-DS), it would look like the following:

```
L1-machine1-IS.example.com|L1-machine1-DS.example.com:389
```

```
L2-machine2-IS.example.com|L2-machine2-DS.example.com:389
```

Secondary LDAP Server

Specifies the host name and port number of a secondary LDAP server available to the Federation Manager platform. If the primary LDAP server does not respond to a request for authentication, this server would then be contacted. If the primary server is up, Federation Manager will switch back to the primary server. The format is also *hostname:port*. Multiple entries must be prefixed by the local server name.

Caution – When authenticating users from a Directory Server that is remote from the Federation Manager enterprise, it is important that both the Primary and Secondary LDAP Server Ports have values. The value for one Directory Server location can be used for both fields.

DN to Start User Search

Specifies the DN of the node where the search for a user would start. (For performance reasons, this DN should be as specific as possible.) The default value is the root of the directory tree. Any valid DN will be recognized. If OBJECT is selected in the Search Scope attribute, the DN should specify one level above the level in which the profile exists. Multiple entries must be prefixed by the local server name. The format is *servername*|*search-dn*. For multiple entries:

servername1|*search-dn servername2*|*search-dn servername3*|*search-dn*

If multiple users are found for the same search, authentication will fail.

DN for Root User Bind

Specifies the DN of the user that will be used to bind to the Directory Server specified in the Primary LDAP Server and Port field as administrator. The authentication service needs to bind as this DN in order to search for a matching user DN based on the user login ID. The default is `amldapuser`. Any valid DN will be recognized.

Password for Root User Bind

Carries the password for the administrator profile specified in the DN for Root User Bind field. There is no default value. Only the administrator's valid LDAP password will be recognized.

Password for Root User Bind (confirm)

Confirm the password.

Attribute Used to Retrieve User Profile

Specifies the attribute used for the naming convention of user entries. By default, Federation Manager assumes that user entries are identified by the `uid` attribute. If your Directory Server uses a different attribute (such as *givenname*) specify the attribute name in this field.

Attributes Used to Search for a User to be Authenticated

Lists the attributes to be used to form the search filter for a user that is to be authenticated, and allows the user to authenticate with more than one attribute in the user's entry. For example, if this field is set to *uid*, *employeenumber*, and *mail*, the user could authenticate with any of these names.

User Search Filter

Specifies an attribute to be used to find the user under the DN to Start User Search field. It works with the User Naming Attribute. There is no default value. Any valid user entry attribute will be recognized.

Search Scope

Indicates the number of levels in the Directory Server that will be searched for a matching user profile. The search begins from the node specified in the DN to Start User Search attribute. The default value is SUBTREE. One of the following choices can be selected from the list:

OBJECT	Searches only the specified node.
ONELEVEL	Searches at the level of the specified node and one level down.
SUBTREE	Search all entries at and below the specified node.

Enable SSL to Access LDAP Server

Enables SSL access to the Directory Server specified in the Primary and Secondary LDAP Server and Port field. By default, the box is not checked and the SSL protocol will not be used to access the Directory Server.

If the LDAP Server is running with SSL enabled (LDAPS), you must make sure that Federation Manager is configured with proper SSL trusted certificates so that it can connect to DirectoryServer over LDAPS protocol

Return User DN to Authenticate

When the Federation Manager directory is the same as the directory configured for LDAP, this option may be enabled. If enabled, this option allows the LDAP authentication module to return the DN instead of the User ID, and no search is necessary. Normally, an authentication module returns only the User ID, and the authentication service searches for the user in the local Federation Manager LDAP. If an external LDAP directory is used, this option is typically not enabled.

LDAP Server Check Interval

This attribute is used for LDAP Server failback. It defines the number of minutes in which a thread will "sleep" before verifying that the LDAP primary server is running.

User Creation Attribute List

This attribute is used by the LDAP authentication module when the LDAP server is configured as an external LDAP server. It contains a mapping of attributes between a local and an external Directory Server. This attribute has the following format:

attr1|*externalattr1*

attr2|*externalattr2*

When this attribute is populated, the values of the external attributes are read from the external Directory Server and are set for the internal Directory Server attributes. The values of the external attributes are set in the internal attributes only when the `User Profile`attribute (in the Core Authentication module) is set to Dynamically Created and the user does not exist in local Directory Server instance. The newly created user will contain the values for internal attributes, as specified in User Creation Attributes List, with the external attribute values to which they map.

Authentication Level

The authentication level is set separately for each method of authentication. The value indicates how much to trust an authentication mechanism. Once a user has authenticated, this value is stored in the SSO token for the session. When the SSO token is presented to an application the user wants to access, the application uses the stored value to determine whether the level is sufficient to grant the user access. If the authentication level stored in an SSO token does not meet the minimum value required, the application can prompt the user to authenticate again through a service with a higher authentication level. The default value is 0.

Note – If no authentication level is specified, the SSO token stores the value specified in the Core Authentication attribute Default Authentication Level.

Membership

The Membership Authentication module is implemented for personalized sites. When membership authentication is enabled, a user can self-register. This means the user can create an account, personalize it, and access it as a registered user without the help of an administrator. The attributes are:

- "Minimum Password Length" on page 122

Minimum Password Length

Specifies the minimum number of characters required for a password set during self-registration. The default value is 8.

Note – If this value is changed, it should also be changed in the `PasswdMinChars` property in the `amAuthMembership.properties` file. `amAuthMembership.properties` is located in the */FederationManager-base/*`locale/` directory.

Default User Roles

Specifies the roles assigned to new users whose profiles are created through self-registration. There is no default value. The administrator must specify the DNs of the roles that will be assigned to the new user.

Note – The role specified must be under the organization for which authentication is being configured. Only the roles that can be assigned to the user will be added during self-registration. All other DNs will be ignored. Filtered roles are not accepted.

User Status After Registration

Specifies whether services are immediately made available to a user who has self-registered. The default value is Active and services are available to the new user. By selecting Inactive, the administrator chooses to make no services available to a new user.

Primary LDAP Server

Specifies the host name and port number of the primary LDAP server specified during Federation Manager installation. This is the first server contacted for authentication. The format is *hostname*:*port*. If there is no port number, assume 389. Multiple entries must be prefixed by the local server name.

If you have Federation Manager deployed with multiple domains, you can specify the communication link between specific instances of Federation Manager and your LDAP server in the following format:

local-servername|*server*:*port local-servername2*|*server2*:*port2* . . .

For example, if you have two Federation Manager instances deployed in different locations (L1-machine1-IS and L2- machine2-IS) communicating with different instances of your LDAP server (L1-machine1-DS and L2-machine2-DS), it would look like the following:

```
L1-machine1-IS.example.com|L1-machine1-DS.example.com:389

L2-machine2-IS.example.com|L2-machine2-DS.example.com:389
```

Secondary LDAP Server

Specifies the host name and port number of a secondary LDAP server available to the Federation Manager platform. If the primary LDAP server does not respond to a request for authentication, this server would then be contacted. If the primary server is up, Federation Manager will switch back to the primary server. The format is also *hostname*:*port*. Multiple entries must be prefixed by the local server name.

Caution – When authenticating users from an LDAP server that is remote from the Federation Manager enterprise, it is important that both the Primary and Secondary LDAP Server Ports have values. The value for one server location can be used for both fields.

DN to Start User Search

Specifies the DN of the node where the search for a user would start. (For performance reasons, this DN should be as specific as possible.) The default value is the root of the directory tree. Any valid DN will be recognized. If OBJECT is selected in the Search Scope attribute, the DN should specify one level above the level in which the profile exists. Multiple entries must be prefixed by the local server name. The format is *servername*|*search dn*.

For multiple entries:

servername1|*search dn servername2*|*search dn servername3*|*search dn* . . .

If multiple users are found for the same search, authentication will fail.

DN for Root User Bind

Specifies the DN of the user that will be used to bind to the Directory Server specified in the Primary LDAP Server and Port field as administrator. The authentication service needs to bind as this DN in order to search for a matching user DN based on the user login ID. The default is `amldapuser`. Any valid DN will be recognized.

Password for Root User Bind

Carries the password for the administrator profile specified in the DN for Root User Bind field. There is no default value. Only the administrator's valid LDAP password will be recognized.

Password for Root User Bind (confirm)

Confirmation of the password.

Attribute Used to Retrieve User Profile

Specifies the attribute used for the naming convention of user entries. By default, Federation Manager assumes that user entries are identified by the `uid` attribute. If your Directory Server uses a different attribute (such as *givenname*) specify the attribute name in this field.

Attributes Used to Search for a User to be Authenticated

Lists the attributes to be used to form the search filter for a user that is to be authenticated, and allows the user to authenticate with more than one attribute in the user's entry. For example, if this field is set to *uid*, *employeenumber*, and *mail*, the user could authenticate with any of these names.

User Search Filter

Specifies an attribute to be used to find the user under the DN to Start User Search field. It works with the User Naming Attribute. There is no default value. Any valid user entry attribute will be recognized.

Search Scope

Indicates the number of levels in the Directory Server that will be searched for a matching user profile. The search begins from the node specified in the attribute "DN to Start User Search" attribute. The default value is SUBTREE. One of the following choices can be selected from the list:

OBJECT Searches only the specified node.

ONELEVEL	Searches at the level of the specified node and one level down.
SUBTREE	Search all entries at and below the specified node.

Enable SSL to Access LDAP Server

Enables SSL access to the Directory Server specified in the Primary and Secondary LDAP Server and Port field. By default, the box is not checked and the SSL protocol will not be used to access the Directory Server.

If the LDAP Server is running with SSL enabled (LDAPS), you must make sure that Federation Manager is configured with proper SSL trusted certificates so that it can connect to Directory Server over LDAPS protocol

Return User DN to Authenticate

When the Federation Manager directory is the same as the directory configured for LDAP, this option may be enabled. If enabled, this option allows the LDAP authentication module to return the DN instead of the User ID, and no search is necessary. Normally, an authentication module returns only the User ID, and the authentication service searches for the user in the local Federation Manager LDAP. If an external LDAP directory is used, this option is typically not enabled.

Authentication Level

The authentication level is set separately for each method of authentication. The value indicates how much to trust an authentication mechanism. Once a user has authenticated, this value is stored in the SSO token for the session. When the SSO token is presented to an application the user wants to access, the application uses the stored value to determine whether the level is sufficient to grant the user access. If the authentication level stored in an SSO token does not meet the minimum value required, the application can prompt the user to authenticate again through a service with a higher authentication level. The default value is 0.

Note – If no authentication level is specified, the SSO token stores the value specified in the Core Authentication attribute Default Authentication Level.

MSISDN

The Mobile Station Integrated Services Digital Network (MSISDN) authentication module enables authentication using a mobile subscriber ISDN associated with a device such as a cellular telephone. It is a non-interactive module. The module retrieves the subscriber ISDN and validates it against the Directory Server to find a user that matches the number. The MSISDN Authentication attributes are:

Trusted Gateway IP Address

Specifies a list of IP addresses of trusted clients that can access MSIDSN modules. You can set the IP addresses of all clients allows to access the MSISDN module by entering the address (for example, `123.456.123.111`) in the entry field and clicking Add. By default, the list is empty. If the attribute is left empty, then all clients are allowed. If you specify none, no clients are allowed.

MSISDN Number Argument

Specifies a list of parameter names that identify which parameters to search in the request header or cookie header for the MSISDN number. For example, if you define *x-Cookie-Param*, *AM_NUMBER*, and *COOKIE-ID*, the MSISDN authentication services will search those parameters for the MSISDN number.

LDAP Server and Port

Specifies the host name and port number of the Directory Server in which the search will occur for the users with MSISDN numbers. The format is*hostname:port*. If there is no port number, assume `389`.

If you have Federation Manager deployed with multiple domains, you can specify the communication link between specific instances of Federation Manager and Directory Server in the following format (multiple entries must be prefixed by the local server name):

```
local_servername|server:port local_servername2|server2:port2 ...
```

For example, if you have two Federation Manager instances deployed in different locations (L1-machine1-IS and L2- machine2-IS) communicating with different instances of Directory Server (L1-machine1-DS and L2-machine2-DS), it would look the following:

```
L1-machine1-IS.example.com|L1-machine1-DS.example.com:389
```

```
L2-machine2-IS.example.com|L2-machine2-DS.example.com:389
```

LDAP Start Search DN

Specifies the DN of the node where the search for the user's MSISDN number should start. There is no default value. The field will recognize any valid DN. Multiple entries must be prefixed by the local server name. The format is *servername*|*search dn*. For multiple entries:

servername1|*search-dn servername2*|*search-dn servername3*|*search-dn*

If multiple users are found for the same search, authentication will fail.

Attribute To Use To Search LDAP

Specifies the name of the attribute in the user's profile that contains the MSISDN number to search for a particular user. The default value is *sunIdentityMSISDNNumber*. This value should not be changed, unless you are certain that another attribute in the user's profile contains the same MSISDN number.

LDAP Server Principal User

Specifies the LDAP bind DN to allow MSISDN searches in the LDAP server. The default bind DN is `cn=amldapuser,ou=DSAME Users,dc=sun,dc=com`.

LDAP Server Principal Password

Specifies the LDAP bind password for the bind DN, as defined in LDAP Server Principal User.

LDAP Server Principal Password (confirm)

Confirm the password.

Enable SSL for LDAP Access

Enables SSL access to the Directory Server specified in the LDAP Server and Port attribute. By default, this is not enabled and the SSL protocol will not be used to access the Directory Server. However, if this attribute is enabled, you can bind to a non-SSL server.

LDAP Attribute Used to Retrieve User Profile

Specifies the headers to use for searching the request for the MSISDN number. The supported values are as follows:

Cookie Header — Performs the search in the cookie. RequestHeader - performs the search in the request header.

RequestHeader	Performs the search in the request header.
RequestParameter	Performs the search in the request parameter. By default, all options are selected.

Return User DN on Authentication

When the Federation Manager directory is the same as the directory configured for MSDISN, this option may be enabled. If enabled, this option allows the authentication module to return the DN instead of the User ID, and no search is necessary. Normally, an authentication module returns only the User ID, and the authentication service searches for the user in the local Federation Manager. If an external directory is used, this option is typically not enabled.

Authentication Level

The authentication level is set separately for each method of authentication. The value indicates how much to trust an authentication mechanism. Once a user has authenticated, this value is stored in the SSO token for the session. When the SSO token is presented to an application the user wants to access, the application uses the stored value to determine whether the level is sufficient to grant the user access. If the authentication level stored in an SSO token does not meet the minimum value required, the application can prompt the user to authenticate again through a service with a higher authentication level. The default value is 0.

Note – If no authentication level is specified, the SSO token stores the value specified in the Core Authentication attribute Default Authentication Level.

RADIUS

This module allows for authentication using an external Remote Authentication Dial-In User Service (RADIUS) server. The attributes are:

- "Server 1" on page 129
- "Server 2" on page 129
- "Shared Secret" on page 129
- "Shared Secret Confirm" on page 129
- "Port Number" on page 129
- "Timeout" on page 129
- "Authentication Level" on page 129

Server 1

Displays the IP address or fully qualified host name of the primary RADIUS server. The default IP address is 127.0.0.1. The field will recognize any valid IP address or host name. Multiple entries must be prefixed by the local server name as in the following syntax:

local-servername|*ip-address local-servername2*|*ip-address2* . . .

Server 2

Displays the IP address or fully qualified domain name (FQDN) of the secondary RADIUS server. It is a failover server which will be contacted if the primary server could not be contacted. The default IP address is 127.0.0.1. Multiple entries must be prefixed by the local server name as in the following syntax:

local-servername|*ip-address local-servername2*|*ip-address2* . . .

Shared Secret

Carries the shared secret for RADIUS authentication. The shared secret should have the same qualifications as a well-chosen password. There is no default value for this field.

Shared Secret Confirm

Confirmation of the shared secret for RADIUS authentication.

Port Number

Specifies the port on which the RADIUS server is listening. The default value is 1645.

Timeout

Specifies the time interval in seconds to wait for the RADIUS server to respond before a timeout. The default value is 3 seconds. It will recognize any number specifying the timeout in seconds.

Authentication Level

The authentication level is set separately for each method of authentication. The value indicates how much to trust an authentication mechanism. Once a user has authenticated, this value is stored in the SSO token for the session. When the SSO token is presented to an application the user wants to access, the application uses the stored value to determine whether the level is sufficient to grant the user access. If the authentication level stored in an SSO token does not meet the minimum value required, the application can prompt the user to authenticate again through a service with a higher authentication level. The default value is 0.

Note – If no authentication level is specified, the SSO token stores the value specified in the Core Authentication attribute Default Authentication Level.

SafeWord

This module allows for users to authenticate using Secure Computing's SafeWord or SafeWord PremierAccess authentication servers. The attributes are:

Server

Specifies the SafeWord or SafeWord PremiereAccess server name and port. Port 7482 is set as the default for a SafeWord server. The default port number for a SafeWord PremierAccess server is 5030.

Server Verification Files Directory

Specifies the directory into which the SafeWord client library places its verification files. The default is as follows:

```
/var/opt/SUNWam/auth/safeword/serverVerification
```

If a different directory is specified in this field, the directory must exist before attempting SafeWord authentication.

Logging Enable

Enables SafeWord logging. By default, SafeWord logging is enabled.

Logging Level

Select the SafeWord logging level from the drop-down menu. The levels are:

- DEBUG
- ERROR
- INFO
- NONE

Log File

Specifies the directory path and log file name for SafeWord client logging. If a path or filename different from the default is specified, it must exist before attempting SafeWord authentication. If more than one organization is configured for SafeWord authentication, and different SafeWord servers are used, different paths must be specified or only the first organization where SafeWord authentication occurs will work.

Authentication Connection Timeout

Defines the timeout period (in seconds) between the SafeWord client and the SafeWord server. The default is 120 seconds.

Client Type

Defines the Client Type that the SafeWord server uses to communicate with different clients, such as Mobile Client, VPN, Fixed Password, Challenge/Response, and so forth.

EASSP Version

This attribute specifies the Extended Authentication and Single Sign-on Protocol (EASSP) version. This field accepts either the standard (101) or premier access (201) protocol versions.

Minimum Authenticator Strength

Defines the minimum authenticator strength for the client/SafeWord server authentication. Each client type has a different authenticator value, and the higher the value, the higher the authenticator strength. 20 is the highest value possible. 0 is the lowest value possible.

Authentication Level

The authentication level is set separately for each method of authentication. The value indicates how much to trust an authentication mechanism. Once a user has authenticated, this value is stored in the SSO token for the session. When the SSO token is presented to an application the user wants to access, the application uses the stored value to determine whether the level is sufficient to grant the user access. If the authentication level stored in an SSO token does not

meet the minimum value required, the application can prompt the user to authenticate again through a service with a higher authentication level. The default value is 0.

Note – If no authentication level is specified, the SSO token stores the value specified in the Core Authentication attribute Default Authentication Level.

SecurID

This module allows for authentication using RSA ACE/Server software and RSA SecurID authenticators. the SecurID authentication module is not available for the Linux or Solaris x86 platforms and this should not be registered, configured, or enabled on these two platforms. It is only available for Solaris. The attributes are:

ACE/Server Configuration Path

Specifies the directory in which the SecurID ACE/Server `sdconf.rec` file is located, by default in `/opt/ace/data` If you specify a different directory in this field, the directory must exist before attempting SecurID authentication.

Helper Configuration Port

Specifies the port on which the SecurID helper *listens* upon startup for the configuration information contained in the SecurID Helper Authentication Port attribute. The default is 58943.

If this attribute is changed, you must also change the `securidHelper.ports` property in the `AMConfig.properties` file, and restart Federation Manager. The entry in `AMConfig.properties` is a space-separated list of the ports for the instances of SecurID helpers. For each organization that communicates with a different ACE/Server (which has a different `sdconf.rec` file), there must be a separate SecurID helper.

Helper Authentication Port

Specifies the port that the organization's SecurID authentication module will configure its SecurID helper instance to 'listen' for authentication requests. This port number must be unique across all organizations using SecurID or UNIX authentication. The default port is 57943.

Authentication Level

The authentication level is set separately for each method of authentication. The value indicates how much to trust an authentication mechanism. Once a user has authenticated, this value is stored in the SSO token for the session. When the SSO token is presented to an application the user wants to access, the application uses the stored value to determine whether the level is sufficient to grant the user access. If the authentication level stored in an SSO token does not meet the minimum value required, the application can prompt the user to authenticate again through a service with a higher authentication level. The default value is 0.

Note – If no authentication level is specified, the SSO token stores the value specified in the Core Authentication attribute Default Authentication Level.

Windows Desktop SSO

This module is specific to Windows and is also known as Kerberos authentication. The user presents a Kerberos token to Federation Manager through the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) protocol. The Windows Desktop SSO authentication plug-in module provides a client (user) with desktop single sign-on. This means that a user who has already authenticated with a key distribution center can be authenticated with Federation Manager without having to provide the login information again. The Windows Desktop SSO attributes are global attributes. The attributes are:

Service Principal

Specifies the Kerberos principal that is used for authentication. Use the following format:

`HTTP/`*hostname*`.`*domainname*`@`*dc-domain-name*

hostname and *domainname* represent the host name and domain name of the Federation Manager instance. *dc-domain-name* is the Kerberos domain in which the Windows 2000 Kerberos server (domain controller) resides. It is possibly different from the domain name of Federation Manager.

Keytab File Name

This attribute specifies the Kerberos keytab file that is used for authentication. Use the following format, although the format is not required:

hostname`.HTTP.keytab`

hostname is the host name of the Federation Manager instance.

Kerberos Realm

This attribute specifies the Kerberos Distribution Center (domain controller) domain name. Depending up on your configuration, the domain name of the domain controller may be different than the Federation Manager domain name.

Kerberos Server Name

This attribute specifies the Kerberos Distribution Center (the domain controller) hostname. You must enter the fully qualified domain name (FQDN) of the domain controller.

Return Principal with Domain Name

If enabled, this attributes allows Federation Manager to automatically return the Kerberos principal with the domain controller's domain name during authentication.

Authentication Level

The authentication level is set separately for each method of authentication. The value indicates how much to trust an authentication mechanism. Once a user has authenticated, this value is stored in the SSO token for the session. When the SSO token is presented to an application the user wants to access, the application uses the stored value to determine whether the level is sufficient to grant the user access. If the authentication level stored in an SSO token does not meet the minimum value required, the application can prompt the user to authenticate again through a service with a higher authentication level. The default value is 0.

Note – If no authentication level is specified, the SSO token stores the value specified in the Core Authentication attribute Default Authentication Level.

Windows NT

The Windows NT Authentication module allows for authentication against a Microsoft Windows NT server. The values applied to them under Service Configuration become the

default values for the Windows NT Authentication template. The service template needs to be created after registering the service for the organization. The default values can be changed after registration by the administrator.

Note – In order to activate the Windows NT Authentication module, Samba Client 2.2.2 must be downloaded and installed to the following directory:

AcessManager-base`/SUNWam/bin`

The Samba Client is a file and print server for blending Windows and UNIX machines without requiring a separate Windows NT/2000 Server. Red Hat Linux ships with a Samba client, located in the`/usr/bin` directory. In order to authenticate using the Windows NT Authentication service for Linux, copy the client binary to*FederationManager-base*`/identity/bin`.

The Windows NT attributes are:

- "Authentication Domain" on page 135
- "Authentication Host" on page 135
- "Samba Configuration File Name" on page 135
- "Authentication Level" on page 136

Authentication Domain

Defines the name of the domain to which the user belongs.

Authentication Host

Defines the name of the Windows NT authentication host. Name resolution will be performed based on the netBIOS name as opposed to the fully qualified domain name (FQDN). If you do not have a server on your subnet supplying netBIOS name resolution, the mappings should be hardcoded. By default, the first part of the FQDN is the netBIOS name. For example, the host name should be `example1` not `example1.company1.com`.

Note – If the DHCP (Dynamic Host Configuration Protocol) is used, put a suitable entry in the `HOSTS` file on the Windows 2000 machine.

Samba Configuration File Name

Defines the Samba configuration filename and supports the `-s` option in the `smbclient` command. The value must be the full directory path where the Samba configuration file is located. For example, `/etc/opt/SUNWam/config/smb.conf`.

Authentication Level

The authentication level is set separately for each method of authentication. The value indicates how much to trust an authentication mechanism. Once a user has authenticated, this value is stored in the SSO token for the session. When the SSO token is presented to an application the user wants to access, the application uses the stored value to determine whether the level is sufficient to grant the user access. If the authentication level stored in an SSO token does not meet the minimum value required, the application can prompt the user to authenticate again through a service with a higher authentication level. The default value is 0.

Note – If no authentication level is specified, the SSO token stores the value specified in the Core Authentication attribute Default Authentication Level.

◆ ◆ ◆ CHAPTER 7

Federation, Authentication Domains and Entities

Federation Manager implements the Liberty Alliance Project *Liberty Identity Federation Framework* (Liberty ID-FF) specification. The Liberty ID-FF defines a set of protocols, bindings and profiles that provide a solution for identity federation, cross-domain authentication and session management.

This chapter covers the following topics:

Entities: Provider and Affiliate

In Federation Manager an *entity* can contain configuration information for an individual identity provider, an individual service provider, or one of each. An entity can also contain configuration information for an *affiliation*, a group of providers of either type. Both provider and affiliation entities can be configured using the Federation Manager Console by selecting Entities under Federation.

Note – An entity can be created but it will not be available for assignment to an authentication domain until it has been populated with provider(s).

There are two types of entities:

Provider Entity A *provider entity* holds the metadata for individual service or identity providers. All identity providers and service providers must first be configured within a provider entity. After they are configured in a provider entity, they can be associated with an authentication domain, or chosen to be included in an *affiliate entity*. Using the entity descriptor attributes, one

individual identity provider, one individual service provider, or one provider of each type can be defined within a provider entity. A provider entity profile contains the following groups of attributes:

- General Attributes
- Identity Provider
- Service Provider

Providers in a provider entity must also be defined as *remote* or *hosted*.

Affiliate Entity An *affiliate entity* holds the metadata that defines a group of one or more providers that was formed without regard to the boundaries of an authentication domain. This *affiliation* (referenced by an affiliationID) is formed and maintained by an *affiliation owner* (referenced by the providerID of the entity that defined it) who chooses the trusted providers from already configured provider entities. Members of the affiliation may invoke services either as a member of the affiliation (using the affiliationID), or individually (using their providerID). For example, when a service provider issues an authentication request on behalf of an affiliation, the AffiliationID will be used to achieve single sign-on and the identity provider will resolve federations based on the same AffiliationID. The affiliate entity itself does not contain configuration information for any providers, only configuration information for the affiliation. An affiliate entity profile contains the following groupings of attributes:

- General Attributes
- Affiliate

Using these attributes, a group of providers are collectively identified and maintained by an affiliation owner.

Typically, providers in an authentication domain exchange XML metadata as specified in the Liberty Alliance Project Metadata specification. Federation Manager provides command line utilities to import these XML files as *remote* providers. *Hosted* providers (those deployed within a specific instance of Federation Manager) already have configured metadata in the directory where Federation Manager was installed. Federation Manager also provides facilities to export XML metadata representing a given hosted provider. This exported data can then be used as input to configure a provider in a compliant product (Sun Java System Access Manager or another instance of Federation Manager).

Depending on the type of provider the Federation Manager instance adopts, XML metadata can be loaded using the command line tools to create hosted providers. All providers created this way can be managed as entities using the Federation Manager Console.

All configured entities are listed under Entity Descriptors. The following tasks are associated with entities:

▼ To Create a New Provider Entity or Affiliate Entity

Establishing an entity is a two-step process. First, you create the entity. (This is when you define whether it is a *provider entity* or an *affiliate entity*.) After creating the entity, you configure it with provider information or affiliation information. The following procedure is used to create the new entity only. The starting point is the Entity Descriptors screen under Federation.

Note – An entity can be created but it will not be available for assignment to an authentication domain until it has been populated with provider(s).

---

1 **Click New to display the entity attributes.**

The New Entity Descriptor page is displayed.

2 **Type a value for the Entity Name.**

This field specifies the Uniform Resource Identifier (URI) of the entity and must be unique. For example, `http://shivalik.sun.com` or `http://provider2.com:875`.

3 **(Optional) Enter a description of the entity in the Description field.**

4 **Select one of the following options to define the entity's type.**

- **Select Provider and click Create.**

  The new entity is now displayed as a provider entity in the list of configured Entities.

- **Select Affiliate, enter a value for both Affiliate ID and Affiliate Owner ID and click Create.**

  The Affiliate ID specifies a URI defined by the Affiliate Owner that uniquely represents the affiliate entity. For example, `http://shivalik.sun.com` or `http://provider2.com:875`. The Affiliate Owner ID is the provider ID of the service provider (defined in a provider entity) that is forming the affiliation. After entering these values and clicking OK, the new entity is displayed as an affiliate entity in the list of configured Entities.

  Note – Defining a service provider as the Affiliate Owner does not automatically include it as a member of the affiliate. If an owner is also a member, the provider ID must be defined in both attributes.

The new entity is displayed on the Entities screen.

▼ To Modify the General Attributes of a Provider Entity or an Affiliate Entity

After creating an entity, you can edit its profile. This might include modifying attributes, adding identity providers and service providers (as entities), or configuring affiliates. Editing the General attributes of an entity profile might entail modifying the already-defined Common attributes, adding a contact person, or providing optional organizational information. The following procedure is for editing the General attributes. The starting point is the Entities screen under Federation.

1. **Select the name of a configured entity to modify its profile.**

   The entity's profile page is displayed.

2. **Select General from the View menu.**

3. **Edit the values of any of the Entity Common Attributes.**

   Entity Name
   : The static value of this attribute is the name provided when you created the entity.

   Type
   : The static value of this attribute is the type of entity, Provider or Affiliate.

   Description
   : The value of this attribute is the description provided when you created the entity. You may modify the description originally entered.

   Valid Until
   : Enter the expiration date for the metadata pertaining to the provider. Use Coordinated Universal Time (UTC) in the format *yyyy-mm-dd*T*hh:mm:ss.S*Z. For example, `2004-12-31T12:30:00.0Z`.

Cache Duration
Enter the maximum amount of time the entity can be cached. The value is defined in the format P*n*Y*n*M*n*DT*n*H*n*M*n*S, where *n* is an integer variable. For example, `P1Y2M4DT9H8M20S` defines the cache duration as 1 year, 2 months, 4 days, 9 hours, 8 minutes, and 20 seconds.

4 Provide values for the Entity Contact Person Profile attributes.

First Name
Type the given name of the entity's contact person.

Last Name
Type the surname of the entity's contact person.

Type
Choose the type of contact from the drop-down menu:

- Administrative
- Billing
- Technical
- Other

Company
Type the name of the company that employs this person.

Liberty Principal ID
Type a URI that points to an online instance of the contact person's personal information profile.

Emails
Type one or more email addresses for the contact person.

Telephone Numbers
Type one or more telephone numbers for the contact person.

5 Provide values for the Organizations Profile attributes.

The Organizations Profile attributes provide basic information that may be required during interactions. These attributes are optional.

Names
Type the complete legal name of the entity's organization. Use the format *locale*|*organization-name*. For example, `en|`*organization-name*`.com`.

Note – If the Names attribute contains a value, it is required to add values to the Display Names and URL attributes.

Display Names
Type a name that is suitable for display. Use the format *locale*|*organization-display-name*. For example, `en|`*organization-display-name*`.com`.

URL
Type a URL that can be used to direct a principal to additional information on the entity's organization. Use the format *locale*|*organization-URL*. For example, `en|http://www.`*organization-name*`.com`.

6 Click Save to complete the configuration, or continue defining values for Identity Provider or Service Provider.

Procedures for defining values for Identity Provider or Service Provider attributes are detailed in the following sections:

- **To define values for Identity Provider attributes, see "To Add an Identity Provider to a Provider Entity" on page 142.**
- **To define values for Service Provider attributes, see "To Add a Service Provider to a Provider Entity" on page 153.**

▼ To Add an Identity Provider to a Provider Entity

After editing the General attributes of a provider entity, you can add identity providers to it. The first step in this process is to define the type of identity provider you are adding.

- A *hosted provider* is hosted on the same server as Federation Manager.
- A *remote provider* is not hosted on the same server as Federation Manager.

Follow this procedure to begin the process of adding an identity provider to a provider entity. The starting point is the Entities screen under Federation.

1 Select the name of a configured provider entity to modify its profile.

The entity's profile page is displayed.

2 Select Identity Provider from the View menu.

3 Select the type of provider you are adding.

- Choose New Hosted Provider if the provider is hosted on the same server as Federation Manager.
- Choose New Remote Provider if the provider is not hosted on the same server as Federation Manager.

The appropriate Identity Provider profile page is displayed.

4 Based on your selection in the previous step, choose one of the following:

- **"To Add a New Hosted Identity Provider to a Provider Entity" on page 143**

- "To Add a New Remote Identity Provider to a Provider Entity" on page 149

To Add a New Hosted Identity Provider to a Provider Entity

A *hosted provider* is hosted on the same server as Federation Manager. Editing the New Hosted Identity Provider attributes entails adding metadata concerning the identity provider to the provider's entity profile. The starting point for this procedure is "To Add an Identity Provider to a Provider Entity" on page 142.

Caution – Federation Manager does not support hosted identity providers. These attributes and this procedure are included for testing purposes only. For real deployment scenarios, use an identity provider focused product such as Sun Java System Access Manager.

1 Provide information for the Common Attributes.

Common Attributes contain values that generally define the identity provider itself.

Description
: This attribute contains the description provided when you created the entity. You can modify the description originally entered.

Valid Until
: Type the expiration date for the metadata pertaining to the provider. Use Coordinated Universal Time (UTC) in the format *yyyy-mm-dd*T*hh:mm:ss.*SZ. For example, `2004-12-31T12:30:00.0Z`.

Cache Duration
: Type the maximum amount of time the entity can be cached. The value is defined in the format P*n*Y*n*M*n*DT*n*H*n*M*n*S, where *n* is an integer. For example, `P1Y2M4DT9H8M20S` defines the cache duration as 1 year, 2 months, 4 days, 9 hours, 8 minutes, and 20 seconds.

Protocol Support Enumeration
: Choose the protocol release supported by this entity.

  - `urn:liberty:iff:2003-08` refers to the *Liberty Identity Federation Framework version 1.2*.
  - `urn:liberty:iff:2002-12` refers to the *Liberty Identity Federation Framework version 1.1*.

Server Name Identifier Mapping Binding
: Name identifier mapping allows a service provider to obtain a name identifier for a principal that has federated in the namespace of a different service provider. Implementing this protocol allows the requesting service provider to communicate with the second service provider without an identity federation having been enabled. Type a URI that identifies the communication specifications and click Add.

Note – Currently, the Name Identifier Mapping profile only supports SOAP. If this attribute is used, its value must be `http://projectliberty.org/profiles/nim-sp-http`.

Additional Meta Locations
: Type a URL that points to other relevant metadata concerning the provider.

Signing Key: Key Alias
: Type the key alias used to sign requests and responses.

Encryption Key: Key Alias
: Type the security certificate alias. Certificates are stored in a Java keystore file. Each specific certificate is mapped to an alias that is used to fetch the certificate.

Encryption Key: Key Size
: Type the length for keys used by the web service consumer when interacting with another entity.

Encryption Key: Encryption Method
: Choose the method of encryption. The choices include:

  - None
  - AES
  - DES
  - 3DES

Name Identifier Encryption
: Select the check box to enable encryption of the name identifier.

2 Provide information for the Communication URLs attributes.

Communication URLs attributes contain locations for sending transmissions to the identity provider being configured.

SOAP Endpoint
: Type a URL to the identity provider's SOAP message receiver. This value communicates the location of the SOAP receiver in non browser communications.

Single Sign-On Service URL
: Type a URL to which service providers can send single sign-on and federation requests.

Single Logout Service
: Type a URL to which service providers can send logout requests. Single logout synchronizes the logout functionality across all sessions authenticated by the identity provider.

Single Logout Return
: Type a URL to which the identity provider will redirect the principal after completing a logout.

Federation Termination Service
: Type a URL to which a service provider will send federation termination requests.

Federation Termination Return
: Type a URL to which the identity provider will redirect the principal after completing federation termination.

Name Registration Service
: Type a URL to which a service provider will send requests to specify the name identifier that will be used when communicating with the identity provider about a principal. This service can only be used after a federation session is established.

Name Registration Return
: Type a URL to which the identity provider will redirect the principal after HTTP name registration has been completed.

3 Provide information for the Communication Profiles attributes.

Communication Profiles attributes define the transmission methods used by the identity provider.

Federation Termination
: Select a profile to notify other providers of a principal's federation termination:

- HTTP Redirect
- SOAP

Single Logout
: Select a profile to notify other providers of a principal's logout:

- HTTP Redirect
- HTTP Get
- SOAP

Name Registration
: Select a profile to notify other providers of a principal's name registration:

- HTTP Redirect
- SOAP

Single Sign-on/Federation
: Select a profile for sending authentication requests:

- Browser Post (specifies a browser-based HTTP POST protocol)
- Browser Artifact (specifies a non-browser SOAP-based protocol)
- LECP (specifies a Liberty-enabled Client Proxy)

Note – Federation Manager can handle requests that come from a Liberty-enabled client proxy profile, but it requires additional configuration that is beyond the scope of this manual.

4 Select any of the available authentication domains to assign to the provider.

A provider can belong to one or more authentication domains. However, a provider without a specified authentication domain can not participate in Liberty-based communications. If no authentication domains have been created, you can define this attribute later.

5 Provide information for the Hosted Configuration attributes.

Hosted Configuration attributes define general information regarding the provider hosted on the same machine as Federation Manager.

Provider URL
: Type the URL of the provider hosted locally.

Provider Alias
: Type an alias name for the provider hosted locally.

Authentication Type
: Select what type of provider should be used for authentication requests from a provider hosted locally.

  - *Remote* specifies that the provider hosted locally would contact a remote identity provider upon receiving an authentication request.
  - *Local* specifies that the provider hosted locally should contact a local identity provider upon receiving an authentication request (essentially, itself).

Default Authentication Context
: Select the authentication context class (method of authentication) to use if the identity provider does not receive this information as part of a service provider request. This value also specifies the authentication context used by the service provider when an unknown user tries to access a protected resource. The options are:

  - Password
  - Mobile Digital ID
  - Smartcard
  - Smartcard-PKI
  - MobileUnregistered
  - Software-PKI
  - Previous-Session
  - Mobile Contract
  - Time-Sync-Token
  - Password-ProtectedTransport

Identity Provider Forced Authentication
: Select the check box to indicate that the identity provider must reauthenticate the principal (even if the principal has an existing session from a prior authentication) when an authentication request is received from a remote service provider. This attribute is enabled by default.

Request Identity Provider to be Passive
: Select the check box to specify that the identity provider must not prompt a user for authentication credentials upon receiving an authentication request from a remote service provider. The default (unchecked) is to authenticate the user upon receiving an authentication request.

Realm
: Type a value which points to the realm in which this provider is configured. For example, `/sp`.

Liberty Version URI
: Type the URI of the version of the Liberty Alliance Project specification being used. The default value is `http://projectliberty.org/specs/v1`.

Name Identifier Implementation
: This field defines the class used by a service provider to participate in name registration. Name registration is a profile by which service providers specify a principal's name identifier that an identity provider will use when communicating to the service provider. The value is `com.sun.identity.federation.services.util.FSNameIdentifierImpl`.

Home Page URL
: Type the URL of the home page for the provider hosted locally.

Single Sign-on Failure Redirect URL
: Type the URL to which a principal will be redirected if single sign-on has failed.

Assertion Issuer
: Type the name of the host that issues the assertion. This value might be the load balancer's host name if Federation Manager is behind one.

Generate Discovery Bootstrapping Resource Offering
: Select the check box if you want a Discovery Service Resource Offering to be generated during the Liberty-based single sign on process for bootstrapping purposes.

Auto Federation
: Select the check box to enable auto federation.

Auto Federation Common Attribute Name
: When creating an Auto Federation Attribute Statement, the value of this attribute will be used. The statement will contain `AutoFedAttribute` as the attribute name and this common attribute as the value.

Attribute Statement Plugin
: Specify a pluggable class used for adding attribute statements to an assertion that is generated during the Liberty-based single sign-on process.

6 Provide information for the SAML Attributes.

SAML Attributes define general values regarding how the identity provider will send SAML assertions.

Assertion Interval
: Type the interval of time (in seconds) that an assertion issued by the identity provider will remain valid. A principal will remain authenticated until the assertion interval expires.

Cleanup Interval
: Type the interval of time (in seconds) before assertions stored in the identity provider will be cleared.

Artifact Timeout
: Type the interval of time (in seconds) to specify the timeout for assertion artifacts.

Assertion Limit
: Type a number to define how many assertions an identity provider can issue, or how many assertions that can be stored.

7 Provide values for the Organization attributes.

Note – The Organization attributes provide basic information that may be required when interacting with a principal. These attributes are optional.

Names
: Type the complete legal name of the entity's organization. Use the format *locale*|*organization-name*. For example, `en|`*organization-name*`.com`.

  Note – If the Names attribute contains a value, it is required to add values to the Display Names and URL attributes.

Display Names
: Type a name that is suitable for display. Use the format *locale*|*organization-display-name*. For example, `en|`*organization-display-name*`.com`.

URL
: Type a URL that can be used to direct a principal to additional information on the entity's organization. Use the format *locale*|*organization-URL*. For example, `en|http://www.`*organization-name*`.com`.

8 **(Optional) To configure Contact Persons for the provider, click New Contact Person.**

For more information, see "To Add a Contact Person to a Provider in a Provider Entity" on page 189.

9 **Click OK to add the provider to the entity.**

10 **Continue configuring the entity by selecting another option from the View menu or click Save to complete the configuration.**

▼ To Add a New Remote Identity Provider to a Provider Entity

A *remote provider* is not hosted on the same server as Federation Manager. Editing the New Remote Identity Provider attributes entails adding metadata concerning the identity provider to the provider entity profile. The starting point is "To Add an Identity Provider to a Provider Entity" on page 142.

1 **Provide information for the Common Attributes.**

Common Attributes contain values that generally define the identity provider itself.

Description
: This attribute contains the description provided when you created the entity. You can modify the description originally entered.

Valid Until
: Type the expiration date for the metadata pertaining to the provider. Use Coordinated Universal Time (UTC) in the format *yyyy-mm-dd*T*hh:mm:ss.*SZ. For example, `2004-12-31T12:30:00.0Z`.

Cache Duration
: Type the maximum amount of time the entity can be cached. The value is defined in the format P*n*Y*n*M*n*DT*n*H*n*M*n*S, where *n* is an integer. For example, `P1Y2M4DT9H8M20S` defines the cache duration as 1 year, 2 months, 4 days, 9 hours, 8 minutes, and 20 seconds.

Protocol Support Enumeration
: Choose the protocol release supported by this entity.

- `urn:liberty:iff:2003-08` refers to the *Liberty Identity Federation Framework version 1.2*.
- `urn:liberty:iff:2002-12` refers to the *Liberty Identity Federation Framework version 1.1*.

Server Name Identifier Mapping Binding
: Name identifier mapping allows a service provider to obtain a name identifier for a principal that has federated in the namespace of a different service provider. Implementing this

protocol allows the requesting service provider to communicate with the second service provider without an identity federation having been enabled. Type a URI that identifies the communication specifications.

Note – Currently, the Name Identifier Mapping profile only supports SOAP. If this attribute is used, its value must be `http://projectliberty.org/profiles/nim-sp-http`.

Additional Meta Locations
Type a URL that points to other relevant metadata concerning the provider.

Signing Key: Key Alias
Type the key alias used to sign requests and responses.

Encryption Key: Key Alias
Type the security certificate alias. Certificates are stored in a Java keystore file. Each specific certificate is mapped to an alias that is used to fetch the certificate.

Encryption Key: Key Size
Type the length for keys used by the web service consumer when interacting with another entity.

Encryption Key: Encryption Method
Choose the method of encryption. The choices include:

- None
- AES
- DES
- 3DES

Name Identifier Encryption
Select the check box to enable encryption of the name identifier.

2 Provide information for the Communication URLs attributes.

Communication URLs attributes contain locations for sending transmissions to the remote identity provider.

SOAP Endpoint
Type a URL to the identity provider's SOAP message receiver. This value communicates the location of the SOAP receiver in non browser communications.

Single Sign-On Service URL
Type a URL to which service providers can send single sign-on and federation requests.

Single Logout Service
Type a URL to which service providers can send logout requests. Single logout synchronizes the logout functionality across all sessions authenticated by the identity provider.

Single Logout Return
: Type a URL to which the identity provider will redirect the principal after completing a logout.

Federation Termination Service
: Type a URL to which a service provider will send federation termination requests.

Federation Termination Return
: Type a URL to which the identity provider will redirect the principal after completing federation termination.

Name Registration Service
: Type a URL to which a service provider will send requests to specify the name identifier that will be used when communicating with the identity provider about a principal. This service can only be used after a federation session is established.

Name Registration Return
: Type a URL to which the identity provider will redirect the principal after HTTP name registration has been completed.

3 Provide information for the Communication Profiles attributes.

Communication Profiles attributes define the transmission methods used by the identity provider.

Federation Termination
: Select a profile to notify other providers of a principal's federation termination:

- HTTP Redirect
- SOAP

Single Logout
: Select a profile to notify other providers of a principal's logout:

- HTTP Redirect
- HTTP Get
- SOAP

Name Registration
: Select a profile to notify other providers of a principal's name registration:

- HTTP Redirect
- SOAP

Single Sign-on/Federation
: Select a profile for sending authentication requests:

- Browser Post (specifies a browser-based HTTP POST protocol)
- Browser Artifact (specifies a non-browser SOAP-based protocol)
- LECP (specifies a Liberty-enabled Client Proxy)

Note – Federation Manager can handle requests that come from a Liberty-enabled client proxy profile, but it requires additional configuration that is beyond the scope of this manual.

4 Select any of the available authentication domains to assign to the provider.

A provider can belong to one or more authentication domains, however a provider without a specified authentication domain can not participate in Liberty-based communications. If no authentication domains have been created, this attribute can be defined later.

5 Provide information for the Proxy Authentication Configuration attributes.

Proxy Authentication Configuration attributes define values for dynamic identity provider proxying.

Enable Proxy Authentication
: Select the check box to enable proxy authentication for a service provider.

Proxy Identity Providers List
: Add a list of identity providers that can be used for proxy authentication. The value is a URI defined as the provider's identifier.

Maximum Number of Proxies
: Type the maximum number of identity providers that can be proxied.

Use Introduction Cookie for Proxying
: Select the check box if you want introductions to be used to find the proxying identity provider.

6 Provide values for the Organization attributes.

The Organization attributes provide basic information that may be required when interacting with a principal. These attributes are optional.

Names
: Type the complete legal name of the entity's organization. Use the format *locale*|*organization-name*. For example, `en|`*organization-name*`.com`.

Note – If the Names attribute contains a value, it is required to add values to the Display Names and URL attributes.

Display Names
: Type a name that is suitable for display. Use the format *locale*|*organization-display-name*. For example, `en|`*organization-display-name*`.com`.

URL
: Type a URL that can be used to direct a principal to additional information on the entity's organization. Use the format *locale*|*organization-URL*. For example,

`en|http://www.`*organization-name*`.com`.

7 **(Optional) To configure Contact Persons for the provider, click New Contact Person.**

For more information, see "To Add a Contact Person to a Provider in a Provider Entity" on page 189.

8 **Click OK to add the provider to the entity.**

9 **Continue configuring the entity by selecting another option from the View menu or click Save to complete the configuration.**

▼ To Add a Service Provider to a Provider Entity

After editing the General attributes of a provider entity, you can add service providers to it. The first step in this process is to define the type of service provider you are adding.

- A *hosted provider* is hosted on the same server as Federation Manager.
- A *remote provider* is not hosted on the same server as Federation Manager.

Follow this procedure to begin the process of adding a service provider to a provider entity. The starting point is the Entity Descriptors screen under Federation.

1 **Select the name of a configured provider entity to modify its profile.**

The entity's profile page is displayed.

2 **Select Service Provider from the View menu.**

3 **Select the type of provider you are adding:**

- Choose New Hosted Provider if the provider is hosted on the same server as Federation Manager.
- Choose New Remote Provider if the provider is not hosted on the same server as Federation Manager.

The appropriate Service Provider page is displayed.

4 **Based on your selection in the previous step, choose one of the following:**

- "To Add a New Hosted Service Provider to a Provider Entity" on page 154
- "To Add a New Remote Service Provider to a Provider Entity" on page 161

▼ To Add a New Hosted Service Provider to a Provider Entity

A *hosted provider* is hosted on the same server as Federation Manager. Editing the New Hosted Service Provider attributes entails adding metadata concerning the service provider to the provider entity profile. The starting point is "To Add a Service Provider to a Provider Entity" on page 153.

1 Provide information for the Common Attributes.

Common Attributes contain values that generally define the identity provider itself.

Description
: This attribute contains the description provided when you created the entity. You can modify the description originally entered.

Valid Until
: Type the expiration date for the metadata pertaining to the provider. Use Coordinated Universal Time (UTC) in the format *yyyy-mm-dd*T*hh:mm:ss.*SZ. For example, `2004-12-31T12:30:00.0Z`.

Cache Duration
: Type the maximum amount of time the entity can be cached. The value is defined in the format P*n*Y*n*M*n*DT*n*H*n*M*n*S, where *n* is an integer. For example, `P1Y2M4DT9H8M20S` defines the cache duration as 1 year, 2 months, 4 days, 9 hours, 8 minutes, and 20 seconds.

Protocol Support Enumeration
: Choose the protocol release supported by this entity.

  - `urn:liberty:iff:2003-08` refers to the *Liberty Identity Federation Framework version 1.2*.
  - `urn:liberty:iff:2002-12` refers to the *Liberty Identity Federation Framework version 1.1*.

Server Name Identifier Mapping Binding
: Name identifier mapping allows a service provider to obtain a name identifier for a principal that has federated in the namespace of a different service provider. Implementing this protocol allows the requesting service provider to communicate with the second service provider without an identity federation having been enabled. Type a URI that identifies the communication specifications.

Note – Currently, the Name Identifier Mapping profile only supports SOAP. If this attribute is used, its value must be `http://projectliberty.org/profiles/nim-sp-http`.

Additional Meta Locations
: Type a URL that points to other relevant metadata concerning the provider.

Signing Key: Key Alias
Type the key alias used to sign requests and responses.

Encryption Key: Key Alias
Type the security certificate alias. Certificates are stored in a Java keystore file. Each specific certificate is mapped to an alias that is used to fetch the certificate.

Encryption Key: Key Size
Type the length for keys used by the web service consumer when interacting with another entity.

Encryption Key: Encryption Method
Choose the method of encryption. The choices include:

- None
- AES
- DES
- 3DES

Name Identifier Encryption
Select the check box to enable encryption of the name identifier.

2 Provide information for the Communication URLs attributes.

Communication URLs attributes contain locations for sending transmissions to the service provider being configured.

SOAP Endpoint
Type a URL to the service provider's SOAP message receiver. This value communicates the location of the SOAP receiver in non browser communications.

Single Logout Service
Type a URL to which service providers can send logout requests. Single logout synchronizes the logout functionality across all sessions authenticated by the identity provider.

Single Logout Return
Type a URL to which the service provider will redirect the principal after completing a logout.

Federation Termination Service
Type a URL to which another provider will send federation termination requests.

Federation Termination Return
Type a URL to which the service provider will redirect the principal after completing federation termination.

Name Registration Service
Type a URL to which a service provider will send requests to specify the name identifier that will be used when communicating with the identity provider about a principal. This service can only be used after a federation session is established.

Name Registration Return
Type a URL to which the identity provider will redirect the principal after HTTP name registration has been completed.

3 Provide information for the Communication Profiles attributes.

Communication Profiles attributes define the transmission methods used by the service provider.

Federation Termination
Select a profile to notify other providers of a principal's federation termination:

- HTTP Redirect
- SOAP

Single Logout
Select a profile to notify other providers of a principal's logout:

- HTTP Redirect
- HTTP Get
- SOAP

Name Registration
Select a profile to notify other providers of a principal's name registration:

- HTTP Redirect
- SOAP

Single Sign-on/Federation
Select a profile for sending authentication requests:

- Browser Post (specifies a browser-based HTTP POST protocol)
- Browser Artifact (specifies a non-browser SOAP-based protocol)
- LECP (specifies a Liberty-enabled Client Proxy)

Note – Federation Manager can handle requests that come from a Liberty-enabled client proxy profile, but it requires additional configuration that is beyond the scope of this manual.

---

4 Select any of the available authentication domains to assign to the provider.

A provider can belong to one or more authentication domains. However, a provider without a specified authentication domain can not participate in Liberty-based communications. If no authentication domains have been created, this attribute can be defined later.

5 Provide information for the Service Provider attributes.

Service Provider attributes define general information regarding the service provider.

Assertion Consumer URL
Type the URL to the end point which will receive all SAML assertions.

Assertion Consumer Service URL ID
: If the value of the Protocol Support Enumeration common attribute is `urn:liberty:iff:2003-08`, type the required ID.

Set Assertion Consumer Service URL as Default
: Select the check box to use the Assertion Consumer Service URL as the default value when no identifier is provided in the request.

Sign Authentication Request
: Select the check box to make the service provider always signs authentication requests.

Name Registration after Federation
: Select the check box to enable the service provider to participate in name registration after a principal has been federated.

Name ID Policy
: Select the option permitting requester influence over name identifier policy at the identity provider. The options include:

None	The identity provider will return the name identifier(s) corresponding to the federation that exists between the identity provider and the requesting service provider or affiliation group for the principal. If no such federation exists, an error will be returned.
One-time	The identity provider will issue a temporary, one-time-use identifier for the principal after federation.
Federation	The identity provider may start a new identity federation if one does not already exist for the principal.

Enable Affiliation Federation
: Select the check box to enable affiliation federation.

6 Provide information for the Hosted Configuration attributes.

Hosted Configuration attributes define general information regarding the provider hosted on the same machine as Federation Manager.

Provider URL
: Type the URL of the local identity provider.

Provider Alias
: Type an alias name for the local identity provider.

Authentication Type
: Select the provider that should be used for authentication requests from a provider hosted locally.

- *Remote* specifies that the provider hosted locally would contact a remote identity provider upon receiving an authentication request.

- *Local* specifies that the provider hosted locally should contact a local identity provider upon receiving an authentication request (essentially, itself).

Default Authentication Context
: Select the authentication context class (method of authentication) to use if the identity provider does not receive this information as part of a service provider request. This value also specifies the authentication context used by the service provider when an unknown user tries to access a protected resource. The options are as follows:

- Password
- Mobile Digital ID
- Smartcard
- Smartcard-PKI
- MobileUnregistered
- Software-PKI
- Previous-Session
- Mobile Contract
- Time-Sync-Token
- Password-ProtectedTransport

Identity Provider Forced Authentication
: Select the check box to indicate that the identity provider must reauthenticate the principal (even if the principal has an existing session from a prior authentication) when an authentication request is received from a remote service provider. This attribute is enabled by default.

Request Identity Provider to be Passive
: Select the check box to specify that the identity provider must not prompt a user for authentication credentials upon receiving an authentication request from a remote service provider. The default (unchecked) is to authenticate the user upon receiving an authentication request.

Organization DN
: Type the value of the organization's distinguished name.

Liberty Version URI
: Type the URI of the version of the Liberty Alliance Project specification being used. The default value is `http://projectliberty.org/specs/v1`.

Name Identifier Implementation
: This field defines the class used by a service provider to participate in name registration. Name registration is a profile by which service providers specify a principal's name identifier that an identity provider will use when communicating to the service provider. The value is `com.sun.identity.federation.services.util.FSNameIdentifierImpl`.

Home Page URL
: Type the URL of the home page of the identity provider.

Single Sign-on Failure Redirect URL
: Type the URL to which a principal will be redirected if single sign-on has failed.

Auto Federation
: Select the check box to enable auto federation.

Auto Federation Common Attribute Name
: When creating an Auto Federation Attribute Statement, the value of this attribute will be used. The statement will contain `AutoFedAttribute` as the attribute name and this common attribute as the value.

Attribute Statement Plugin
: Specify a pluggable class used for adding attribute statements to an assertion that is generated during the Liberty-based single sign-on process.

User Provider Implementation Class Name
: Specifies a pluggable implementation to store and retrieve the user attribute information from the users data store. The default implementation of the `com.sun.identity.federation.accountmgmt.FSUserProvider` interface is the `com.sun.identity.federation.accountmgmt.DefaultFSUserProvider` class.

Service Provider Adapter Implementation Class Name
: Specifies a pluggable implementation of the `com.sun.identity.federation.plugins.FederationSPAdapter` interface. The implemented class allows applications to customize their actions before and after invoking the federation protocols. For example, a service provider may want to choose to redirect to a specific location after single sign-on. There is no default implementation but the `spi` sample included with Federation Manager makes use of the class.

Configuration for Service Provider Adapter Implementation
: Stores configuration information that may be used to initialize the Service Provider Adapter Implementation Class Name. The usage of this attribute is also demonstrated in the `spi` sample application.

7 Provide information for the Proxy Authentication Attributes.

Proxy Authentication Configuration attributes define values for dynamic identity provider proxying.

Enable Proxy Authentication
: Select the check box to enable proxy authentication for a service provider.

Proxy Identity Providers List
: Add a list of identity providers that can be used for proxy authentication. The value is a URI defined as the provider's identifier.

Maximum Number of Proxies
: Type the maximum number of identity providers that can be proxied.

Use Introduction Cookie for Proxying
: Select the check box if you want introductions to be used to find the proxying identity provider.

8 Provide information for the SAML Attributes.

SAML Attributes define general information regarding SAML assertions.

Assertion Interval
: Type the interval of time (in seconds) for which an assertion issued by the identity provider will remain valid. A principal will remain authenticated until the assertion interval expires.

Cleanup Interval
: Type the interval of time (in seconds) before assertions stored in the identity provider will be cleared.

Artifact Timeout
: Type the interval of time (in seconds) to specify the time out for assertion artifacts.

Assertion Limit
: Type a number to define the amount of assertions an identity provider can issue, or the number of assertions that can be stored.

9 Provide values for the Organizations Profile attributes.

The Organizations Profile attributes provide basic information that may be required when interacting with a principal. These attributes are optional.

Names
: Type the complete legal name of the entity's organization. Use the format *locale*|*organization-name*. For example, `en|`*organization-name*`.com`.

Note – If the Names attribute contains a value, it is required to add values to the Display Names and URL attributes.

Display Names
: Type a name that is suitable for display. Use the format *locale*|*organization-display-name*. For example, `en|`*organization-display-name*`.com`.

URL
: Type a URL that can be used to direct a principal to additional information on the entity's organization. Use the format *locale*|*organization-URL*. For example, `en|http://www.`*organization-name*`.com`.

10 (Optional) To configure Contact Persons for the provider, click New Contact Person.

See "To Add a Contact Person to a Provider in a Provider Entity" on page 189.

11 Continue configuring the entity by selecting another option from the View menu or click OK to complete the configuration.

▼ To Add a New Remote Service Provider to a Provider Entity

A *remote provider* is not hosted on the same server as Federation Manager. Editing the New Remote Service Provider attributes entails adding metadata concerning the service provider to the provider entity profile. The starting point is "To Add a Service Provider to a Provider Entity" on page 153.

1 Provide information for the Common Attributes.

Common Attributes contain values that generally define the identity provider itself.

Description
: This attribute contains the description provided when you created the entity. You can modify the description originally entered.

Valid Until
: Type the expiration date for the metadata pertaining to the provider. Use Coordinated Universal Time (UTC) in the format *yyyy-mm-dd*T*hh:mm:ss.S*Z. For example, `2004-12-31T12:30:00.0Z`.

Cache Duration
: Type the maximum amount of time the entity can be cached. The value is defined in the format P*n*Y*n*M*n*DT*n*H*n*M*n*S, where *n* is an integer. For example, `P1Y2M4DT9H8M20S` defines the cache duration as 1 year, 2 months, 4 days, 9 hours, 8 minutes, and 20 seconds.

Protocol Support Enumeration
: Choose the protocol release supported by this entity.

  - `urn:liberty:iff:2003-08` refers to the *Liberty Identity Federation Framework version 1.2*.
  - `urn:liberty:iff:2002-12` refers to the *Liberty Identity Federation Framework version 1.1*.

Server Name Identifier Mapping Binding
: Name identifier mapping allows a service provider to obtain a name identifier for a principal that has federated in the namespace of a different service provider. Implementing this protocol allows the requesting service provider to communicate with the second service provider without an identity federation having been enabled. Type a URI that identifies the communication specifications.

Note – Currently, the Name Identifier Mapping profile only supports SOAP. If this attribute is used, its value must be `http://projectliberty.org/profiles/nim-sp-http`.

Additional Meta Locations
Type a URL that points to other relevant metadata concerning the provider.

Signing Key: Key Alias
Type the key alias used to sign requests and responses.

Encryption Key: Key Alias
Type the security certificate alias. Certificates are stored in a Java keystore file. Each specific certificate is mapped to an alias that is used to fetch the certificate.

Encryption Key: Key Size
Type the length for keys used by the web service consumer when interacting with another entity.

Encryption Key: Encryption Method
Choose the method of encryption. The choices include:

- None
- AES
- DES
- 3DES

Name Identifier Encryption
Select the check box to enable encryption of the name identifier.

2 Provide information for the Communication URLs attributes.

Communication URLs attributes contain locations for sending transmissions to the remote service provider being configured.

SOAP Endpoint
Type a URL to the service provider's SOAP message receiver. This value communicates the location of the SOAP receiver in non browser communications.

Single Logout Service
Type a URL to which providers can send logout requests. Single logout synchronizes the logout functionality across all sessions authenticated by the identity provider.

Single Logout Return
Type a URL to which the provider will redirect the principal after completing a logout.

Federation Termination Service
Type a URL to which a provider will send federation termination requests.

Federation Termination Return
Type a URL to which the provider will redirect the principal after completing federation termination.

Name Registration Service
Type a URL to which a provider will send requests to specify the name identifier that will be used when communicating with the identity provider about a principal. This service can only be used after a federation session is established.

Name Registration Return
Type a URL to which the provider will redirect the principal after HTTP name registration has been completed.

3 Provide information for the Communication Profiles attributes.

Communication Profiles attributes define the transmission methods used by the service provider.

Federation Termination
Select a profile to notify other providers of a principal's federation termination:

- HTTP Redirect
- SOAP

Single Logout
Select a profile to notify other providers of a principal's logout:

- HTTP Redirect
- HTTP Get
- SOAP

Name Registration
Select a profile to notify other providers of a principal's name registration:

- HTTP Redirect
- SOAP

Single Sign-on/Federation
Select a profile for sending authentication requests:

- Browser Post (specifies a browser-based HTTP POST protocol)
- Browser Artifact (specifies a non-browser SOAP-based protocol)
- LECP (specifies a Liberty-enabled Client Proxy)

Note – Federation Manager can handle requests that come from a Liberty-enabled client proxy profile, but it requires additional configuration that is beyond the scope of this manual.

---

4 Select any of the available authentication domains to assign to the provider.

A provider can belong to one or more authentication domains. However, a provider without a specified authentication domain can not participate in Liberty-based communications. If no authentication domains have been created, this attribute can be defined later.

5 Provide information for the Service Provider attributes.

Service Provider attributes define general information regarding the service provider.

Assertion Consumer URL
: Type the URL to the end point which will receive all SAML assertions.

Assertion Consumer Service URL ID
: If the value of the Protocol Support Enumeration common attribute is `urn:liberty:iff:2003-08`, type the required ID.

Set Assertion Consumer Service URL as Default
: Select the check box to use the Assertion Consumer Service URL as the default value when no identifier is provided in the request.

Sign Authentication Request
: Select the check box to make the service provider always signs authentication requests.

Name Registration after Federation
: Select the check box to enable the service provider to participate in name registration after a principal has been federated.

Name ID Policy
: Select the option permitting requester influence over name identifier policy at the identity provider. The options include:

None	The identity provider will return the name identifier(s) corresponding to the federation that exists between the identity provider and the requesting service provider or affiliation group for the principal. If no such federation exists, an error will be returned.
One-time	The identity provider will issue a temporary, one-time-use identifier for the principal after federation.
Federation	The identity provider may start a new identity federation if one does not already exist for the principal.

Enable Affiliation Federation
: Select the check box to enable affiliation federation.

6 Provide information for the Proxy Authentication Configuration attributes.

Proxy Authentication Configuration attributes define values for dynamic identity provider proxying.

Enable Proxy Authentication
: Select the check box to enable proxy authentication for a service provider.

Proxy Identity Providers List
: Add a list of identity providers that can be used for proxy authentication. The value is a URI defined as the provider's identifier.

Maximum Number of Proxies
: Type the maximum number of identity providers that can be proxied.

Use Introduction Cookie for Proxying
: Select the check box if you want introductions to be used to find the proxying identity provider.

7 Provide values for the Organizations Profile attributes.

The Organizations Profile attributes provide basic information that may be required when interacting with a principal. These attributes are optional.

Names
: Type the complete legal name of the entity's organization. Use the format *locale*|*organization-name*. For example, `en|`*organization-name*`.com`.

Note – If the Names attribute contains a value, it is required to add values to the Display Names and URL attributes.

Display Names
: Type a name that is suitable for display. Use the format *locale*|*organization-display-name*. For example, `en|`*organization-display-name*`.com`.

URL
: Type a URL that can be used to direct a principal to additional information on the entity's organization. Use the format *locale*|*organization-URL*. For example, `en|http://www.`*organization-name*`.com`.

8 (Optional) To configure Contact Persons for the provider, click New Contact Person.

See "To Add a Contact Person to a Provider in a Provider Entity" on page 189.

9 Continue configuring the entity by selecting another option from the View menu or click OK to complete the configuration.

▼ To Modify Hosted Identity Provider Attributes in a Provider Entity

After creating an entity and adding an identity provider, you can edit the identity provider profile. In a provider entity, this might entail adding metadata that was not available to configure when originally adding the identity provider. The starting point is the Entity Descriptors screen under Federation.

1 Click on the name of a configured provider entity to modify its profile.

The entity's profile page is displayed.

2 Select Identity Provider from the View menu.

3 Modify values for the Common Attributes.

Common Attributes contain values that generally define the identity provider itself.

Provider Type
: The static value of this attribute defines whether this is a hosted or remote provider.

Description
: This attribute contains the description provided when you created the entity. You can modify the description originally entered.

Valid Until
: Type the expiration date for the metadata pertaining to the provider. Use Coordinated Universal Time (UTC) in the format *yyyy-mm-dd*T*hh:mm:ss.S*Z. For example, `2004-12-31T12:30:00.0Z`.

Cache Duration
: Type the maximum amount of time the entity can be cached. The value is defined in the format P*n*Y*n*M*n*DT*n*H*n*M*n*S, where *n* is an integer. For example, `P1Y2M4DT9H8M20S` defines the cache duration as 1 year, 2 months, 4 days, 9 hours, 8 minutes, and 20 seconds.

Protocol Support Enumeration
: Choose the protocol release supported by this entity.

  - `urn:liberty:iff:2003-08` refers to the *Liberty Identity Federation Framework version 1.2*.
  - `urn:liberty:iff:2002-12` refers to the *Liberty Identity Federation Framework version 1.1*.

Server Name Identifier Mapping Binding
: Name identifier mapping allows a service provider to obtain a name identifier for a principal that has federated in the namespace of a different service provider. Implementing this protocol allows the requesting service provider to communicate with the second service provider without an identity federation having been enabled. Type a URI that identifies the communication specifications.

Note – Currently, the Name Identifier Mapping profile only supports SOAP. If this attribute is used, its value must be `http://projectliberty.org/profiles/nim-sp-http`.

Additional Meta Locations
Type a URL that points to other relevant metadata concerning the provider.

Signing Key: Key Alias
Type the key alias used to sign requests and responses.

Encryption Key: Key Alias
Type the security certificate alias. Certificates are stored in a Java keystore file. Each specific certificate is mapped to an alias that is used to fetch the certificate.

Encryption Key: Key Size
Type the length for keys used by the web service consumer when interacting with another entity.

Encryption Key: Encryption Method
Choose the method of encryption. The choices include:

- None
- AES
- DES
- 3DES

Name Identifier Encryption
Select the check box to enable encryption of the name identifier.

4 Modify values for the Communication URLs attributes.

Communication URLs attributes contain locations for sending transmissions to the service provider being configured.

SOAP Endpoint
Type a URI to the identity provider's SOAP message receiver. This value communicates the location of the SOAP receiver in non browser communications.

Single Sign-On Service URL
Type a URL to which service providers can send single sign-on and federation requests.

Single Logout Service
Type a URL to which service providers can send logout requests. Single logout synchronizes the logout functionality across all sessions authenticated by the identity provider.

Single Logout Return
Type a URL to which the identity provider will redirect the principal after completing a logout.

Federation Termination Service
Type a URL to which a service provider will send federation termination requests.

Federation Termination Return
Type a URL to which the identity provider will redirect the principal after completing federation termination.

Name Registration Service
Type a URL to which a service provider will send requests to specify the name identifier that will be used when communicating with the identity provider about a principal. This service can only be used after a federation session is established.

Name Registration Return
Type a URL to which the identity provider will redirect the principal after HTTP name registration has been completed.

5 Modify values for the Communication Profiles attributes.

Communication Profiles attributes define the transmission methods used by the identity provider.

Federation Termination
Select a profile to notify other providers of a principal's federation termination:

- HTTP Redirect
- SOAP

Single Logout
Select a profile to notify other providers of a principal's logout:

- HTTP Redirect
- HTTP Get
- SOAP

Name Registration
Select a profile to notify other providers of a principal's name registration:

- HTTP Redirect
- SOAP

Single Sign-on/Federation
Select a profile for sending authentication requests:

- Browser Post (specifies a browser-based HTTP POST protocol)
- Browser Artifact (specifies a non-browser SOAP-based protocol)
- LECP (specifies a Liberty-enabled Client Proxy)

Note – Federation Manager can handle requests that come from a Liberty-enabled client proxy profile, but it requires additional configuration that is beyond the scope of this manual.

6 Select any of the available authentication domains to assign to the provider.

A provider can belong to one or more authentication domains. However, a provider without a specified authentication domain can not participate in Liberty-based communications. If no authentication domains have been created, this attribute can be defined later.

7 Select the authentication context to be used if the identity provider does not receive the information as part of a service provider request.

This attribute maps the Liberty-defined authentication context classes to authentication methods available at the identity provider. This value also specifies the authentication context used by the service provider when an unknown user tries to access a protected resource.

Supported
: Select the check box next to the authentication context class if the identity provider supports it.

Context Reference
: The Liberty-defined authentication context classes are:

- Mobile Contract
- Mobile Digital ID
- MobileUnregistered
- Password
- Password-ProtectedTransport
- Previous-Session
- Smartcard
- Smartcard-PKI
- Software-PKI
- Time-Sync-Token

Key
: Choose the authentication type to which the context is mapped.

Value
: Type the authentication option.

Priority
: Choose a priority level for cases where there are multiple contexts.

8 Choose from the available Trusted Providers and add to the entity.

The list contains configured entities that have been populated with service providers.

9 Provide information for the Hosted Configuration attributes.

Hosted Configuration attributes define general information regarding the provider hosted on the same machine as Federation Manager.

Provider Alias
Type an alias name for the local identity provider.

Authentication Type
Select the provider that should be used for authentication requests from a provider hosted locally.

- *Remote* specifies that the provider hosted locally would contact a remote identity provider upon receiving an authentication request.
- *Local* specifies that the provider hosted locally should contact a local identity provider upon receiving an authentication request (essentially, itself).

Default Authentication Context
Select the authentication context class (method of authentication) to use if the identity provider does not receive this information as part of a service provider request. This value also specifies the authentication context used by the service provider when an unknown user tries to access a protected resource. The options are as follows:

- Password
- Mobile Digital ID
- Smartcard
- Smartcard-PKI
- MobileUnregistered
- Software-PKI
- Previous-Session
- Mobile Contract
- Time-Sync-Token
- Password-ProtectedTransport

Identity Provider Forced Authentication
Select the check box to indicate that the identity provider must reauthenticate the principal (even if the principal has an existing session from a prior authentication) when an authentication request is received from a remote service provider. This attribute is enabled by default.

Request Identity Provider to be Passive
Select the check box to specify that the identity provider must not prompt a user for authentication credentials upon receiving an authentication request from a remote service provider. The default (unchecked) is to authenticate the user upon receiving an authentication request.

Organization DN
Type a value which points to the organization in which this provider is configured. For example, `/sp`.

Liberty Version URI
: Type the URI of the version of the Liberty Alliance Project specification being used. The default value is `http://projectliberty.org/specs/v1`.

Name Identifier Implementation
: This field defines the class used by a service provider to participate in name registration. Name registration is a profile by which service providers specify a principal's name identifier that an identity provider will use when communicating to the service provider. The value is `com.sun.identity.federation.services.util.FSNameIdentifierImpl`.

Home Page URL
: Type the URL of the home page of the identity provider.

Single Sign-on Failure Redirect URL
: Type the URL to which a principal will be redirected if single sign-on has failed.

Assertion Issuer
: Type the name of the host that issues the assertion. This value might be the load balancer's host name if Federation Manager is behind one.

Generate Discovery Bootstrapping Resource Offering
: Select the check box if you want a Discovery Service Resource Offering to be generated during the Liberty-based single sign on process for bootstrapping purposes.

Auto Federation
: Select the check box to enable auto federation.

Auto Federation Common Attribute Name
: When creating an Auto Federation Attribute Statement, the value of this attribute will be used. The statement will contain `AutoFedAttribute` as the attribute name and this common attribute as the value.

Attribute Statement Plugin
: Specify a pluggable class used for adding attribute statements to an assertion that is generated during the Liberty-based single sign-on process.

User Provider Implementation Class Name
: Specifies a pluggable implementation to store and retrieve the user attribute information from the users data store. The default implementation of the `com.sun.identity.federation.accountmgmt.FSUserProvider` interface is the `com.sun.identity.federation.accountmgmt.DefaultFSUserProvider` class.

10 Provide information for the SAML Attributes.

SAML Attributes define general information regarding SAML assertions that are sent by the identity provider.

Assertion Interval
: Type the interval of time (in seconds) that an assertion issued by the identity provider will remain valid. A principal will remain authenticated until the assertion interval expires.

Cleanup Interval
: Type the interval of time (in seconds) before assertions stored in the identity provider will be cleared.

Artifact Timeout
: Type the interval of time (in seconds) to specify the timeout for assertion artifacts.

Assertion Limit
: Type a number to define how many assertions an identity provider can issue, or how many assertions that can be stored.

11 Provide values for the Organizations Profile attributes.

The Organizations Profile attributes provide basic information that may be required when interacting with a principal. These attributes are optional.

Names
: Type the complete legal name of the entity's organization. Use the format *locale*|*organization-name*. For example, `en|`*organization-name*`.com`.

Note – If the Names attribute contains a value, it is required to add values to the Display Names and URL attributes.

Display Names
: Type a name that is suitable for display. Use the format *locale*|*organization-display-name*. For example, `en|`*organization-display-name*`.com`.

URL
: Type a URL that can be used to direct a principal to additional information on the entity's organization. Use the format *locale*|*organization-URL*. For example, `en|http://www.`*organization-name*`.com`.

12 (Optional) To configure Contact Persons for the provider, click New Contact Person.

See "To Add a Contact Person to a Provider in a Provider Entity" on page 189.

13 Continue modifying the entity by selecting another option from the View menu or click Save to complete the configuration.

▼ To Modify Remote Identity Provider Attributes in a Provider Entity

After creating an entity and adding an identity provider, you can edit the identity provider profile. In a provider entity, this might entail adding metadata that was not available to configure when originally adding the identity provider. The starting point is the Entities configuration screen of the Federation module.

1 Click on the name of a configured provider entity to modify its profile.

The entity's profile page is displayed.

2 Select Identity Provider from the View menu.

3 Modify values for the Common Attributes.

Common Attributes contain values that generally define the identity provider itself.

Provider Type
: The static value of this attribute defines whether this is a hosted or remote provider.

Description
: This attribute contains the description provided when you created the entity. You can modify the description originally entered.

Valid Until
: Type the expiration date for the metadata pertaining to the provider. Use Coordinated Universal Time (UTC) in the format *yyyy-mm-dd*T*hh:mm:ss.S*Z. For example, `2004-12-31T12:30:00.0Z`.

Cache Duration
: Type the maximum amount of time the entity can be cached. The value is defined in the format P*n*Y*n*M*n*DT*n*H*n*M*n*S, where *n* is an integer. For example, `P1Y2M4DT9H8M20S` defines the cache duration as 1 year, 2 months, 4 days, 9 hours, 8 minutes, and 20 seconds.

Protocol Support Enumeration
: Choose the protocol release supported by this entity.

  - `urn:liberty:iff:2003-08` refers to the *Liberty Identity Federation Framework version 1.2*.
  - `urn:liberty:iff:2002-12` refers to the *Liberty Identity Federation Framework version 1.1*.

Server Name Identifier Mapping Binding
: Name identifier mapping allows a service provider to obtain a name identifier for a principal that has federated in the namespace of a different service provider. Implementing this protocol allows the requesting service provider to communicate with the second service provider without an identity federation having been enabled. Type a URI that identifies the communication specifications.

Note – Currently, the Name Identifier Mapping profile only supports SOAP. If this attribute is used, its value must be `http://projectliberty.org/profiles/nim-sp-http`.

Additional Meta Locations
: Type a URL that points to other relevant metadata concerning the provider.

Signing Key: Key Alias
: Type the key alias used to sign requests and responses.

Encryption Key: Key Alias
: Type the security certificate alias. Certificates are stored in a Java keystore file. Each specific certificate is mapped to an alias that is used to fetch the certificate.

Encryption Key: Key Size
: Type the length for keys used by the web service consumer when interacting with another entity.

Encryption Key: Encryption Method
: Choose the method of encryption. The choices include:

  - None
  - AES
  - DES
  - 3DES

Name Identifier Encryption
: Select the check box to enable encryption of the name identifier.

4 Modify values for the Communication URLs attributes.

Communication URLs attributes contain locations for sending transmissions to the service provider being configured.

SOAP Endpoint
: Type a URI to the identity provider's SOAP message receiver. This value communicates the location of the SOAP receiver in non browser communications.

Single Sign-On Service URL
: Type a URL to which service providers can send single sign-on and federation requests.

Single Logout Service
: Type a URL to which service providers can send logout requests. Single logout synchronizes the logout functionality across all sessions authenticated by the identity provider.

Single Logout Return
: Type a URL to which the identity provider will redirect the principal after completing a logout.

Federation Termination Service
: Type a URL to which a service provider will send federation termination requests.

Federation Termination Return
: Type a URL to which the identity provider will redirect the principal after completing federation termination.

Name Registration Service
: Type a URL to which a service provider will send requests to specify the name identifier that will be used when communicating with the identity provider about a principal. This service can only be used after a federation session is established.

Name Registration Return
: Type a URL to which the identity provider will redirect the principal after HTTP name registration has been completed.

5 Modify values in the Communication Profiles attributes.

Communication Profiles attributes define the transmission methods used by the identity provider.

Federation Termination
: Select a profile to notify other providers of a principal's federation termination:

- HTTP Redirect
- SOAP

Single Logout
: Select a profile to notify other providers of a principal's logout:

- HTTP Redirect
- HTTP Get
- SOAP

Name Registration
: Select a profile to notify other providers of a principal's name registration:

- HTTP Redirect
- SOAP

Single Sign-on/Federation
: Select a profile for sending authentication requests:

- Browser Post (specifies a browser-based HTTP POST protocol)
- Browser Artifact (specifies a non-browser SOAP-based protocol)
- LECP (specifies a Liberty-enabled Client Proxy)

Note – Federation Manager can handle requests that come from a Liberty-enabled client proxy profile, but it requires additional configuration that is beyond the scope of this manual.

6 Select any of the available authentication domains to assign to the provider.

A provider can belong to one or more authentication domains. However, a provider without a specified authentication domain can not participate in Liberty-based communications. If no authentication domains have been created, this attribute can be defined later.

7 Provide information for the Proxy Authentication Configuration attributes.

Proxy Authentication Configuration attributes define values for dynamic identity provider proxying.

Enable Proxy Authentication
: Select the check box to enable proxy authentication for a service provider.

Proxy Identity Providers List
: Add a list of identity providers that can be used for proxy authentication. The value is a URI defined as the provider's identifier.

Maximum Number of Proxies
: Type the maximum number of identity providers that can be proxied.

Use Introduction Cookie for Proxying
: Select the check box if you want introductions to be used to find the proxying identity provider.

8 Provide values for the Organizations Profile attributes.

The Organizations Profile attributes provide basic information that may be required when interacting with a principal. These attributes are optional.

Names
: Type the complete legal name of the entity's organization. Use the format *locale*|*organization-name*. For example, `en|`*organization-name*`.com`.

Note – If the Names attribute contains a value, it is required to add values to the Display Names and URL attributes.

Display Names
: Type a name that is suitable for display. Use the format *locale*|*organization-display-name*. For example, `en|`*organization-display-name*`.com`.

URL
: Type a URL that can be used to direct a principal to additional information on the entity's organization. Use the format *locale*|*organization-URL*. For example,

`en|http://www.`*organization-name*`.com`.

9 **(Optional) To configure Contact Persons for the provider, click New Contact Person.**

See "To Add a Contact Person to a Provider in a Provider Entity" on page 189.

10 **Click Save to complete the configuration.**

▼ To Modify Hosted Service Provider Attributes in a Provider Entity

After creating an entity and adding a service provider, you can edit the service provider profile. In a provider entity, this might entail adding metadata that was not available to configure when originally adding the service provider. The starting point is the Entities configuration screen of the Federation module.

1 **Click on the name of a configured provider entity to modify its profile.**

2 **Select Service Provider from the View menu.**

3 **Modify values for the Common Attributes.**

Common Attributes contain values that generally define the identity provider itself.

Provider Type
: The static value of this attribute defines whether this is a hosted or remote provider.

Description
: This attribute contains the description provided when you created the entity. You can modify the description originally entered.

Valid Until
: Type the expiration date for the metadata pertaining to the provider. Use Coordinated Universal Time (UTC) in the format *yyyy-mm-dd*T*hh:mm:ss.S*Z. For example, `2004-12-31T12:30:00.0Z`.

Cache Duration
: Type the maximum amount of time the entity can be cached. The value is defined in the format P*n*Y*n*M*n*DT*n*H*n*M*n*S, where *n* is an integer. For example, `P1Y2M4DT9H8M20S` defines the cache duration as 1 year, 2 months, 4 days, 9 hours, 8 minutes, and 20 seconds.

Protocol Support Enumeration
: Choose the protocol release supported by this entity.

- `urn:liberty:iff:2003-08` refers to the *Liberty Identity Federation Framework version 1.2*.
- `urn:liberty:iff:2002-12` refers to the *Liberty Identity Federation Framework version 1.1*.

Server Name Identifier Mapping Binding
Name identifier mapping allows a service provider to obtain a name identifier for a principal that has federated in the namespace of a different service provider. Implementing this protocol allows the requesting service provider to communicate with the second service provider without an identity federation having been enabled. Type a URI that identifies the communication specifications.

Note – Currently, the Name Identifier Mapping profile only supports SOAP. If this attribute is used, its value must be `http://projectliberty.org/profiles/nim-sp-http`.

Additional Meta Locations
Type a URL that points to other relevant metadata concerning the provider.

Signing Key: Key Alias
Type the key alias used to sign requests and responses.

Encryption Key: Key Alias
Type the security certificate alias. Certificates are stored in a Java keystore file. Each specific certificate is mapped to an alias that is used to fetch the certificate.

Encryption Key: Key Size
Type the length for keys used by the web service consumer when interacting with another entity.

Encryption Key: Encryption Method
Choose the method of encryption. The choices include:

- None
- AES
- DES
- 3DES

Name Identifier Encryption
Select the check box to enable encryption of the name identifier.

4 Modify values for the Communication URLs attributes.

Communication URLs attributes contain locations for sending transmissions to the service provider being configured.

SOAP Endpoint
Type a URI to the identity provider's SOAP message receiver. This value communicates the location of the SOAP receiver in non browser communications.

Single Logout Service
Type a URL to which service providers can send logout requests. Single logout synchronizes the logout functionality across all sessions authenticated by the identity provider.

Single Logout Return
: Type a URL to which the identity provider will redirect the principal after completing a logout.

Federation Termination Service
: Type a URL to which a service provider will send federation termination requests.

Federation Termination Return
: Type a URL to which the identity provider will redirect the principal after completing federation termination.

Name Registration Service
: Type a URL to which a service provider will send requests to specify the name identifier that will be used when communicating with the identity provider about a principal. This service can only be used after a federation session is established.

Name Registration Return
: Type a URL to which the identity provider will redirect the principal after HTTP name registration has been completed.

5 Modify values for the Communication Profiles attributes.

Communication Profiles attributes define the transmission methods used by the service provider.

Federation Termination
: Select a profile to notify other providers of a principal's federation termination:

- HTTP Redirect
- SOAP

Single Logout
: Select a profile to notify other providers of a principal's logout:

- HTTP Redirect
- HTTP Get
- SOAP

Name Registration
: Select a profile to notify other providers of a principal's name registration:

- HTTP Redirect
- SOAP

Single Sign-on/Federation
: Select a profile for sending authentication requests:

- Browser Post (specifies a browser-based HTTP POST protocol)
- Browser Artifact (specifies a non-browser SOAP-based protocol)
- LECP (specifies a Liberty-enabled Client Proxy)

Note – Federation Manager can handle requests that come from a Liberty-enabled client proxy profile, but it requires additional configuration that is beyond the scope of this manual.

6 Select any of the available authentication domains to assign to the provider.

A provider can belong to one or more authentication domains. However, a provider without a specified authentication domain can not participate in Liberty-based communications. If no authentication domains have been created, this attribute can be defined later.

7 Type a positive number in the Level box for the challenge/response protocol.

The number corresponds to the authentication level defined for the particular authentication module. The user will be redirected to a module with the defined authentication level.

- Mobile Contract
- Mobile Digital ID
- Mobile Unregistered
- Password
- Password-ProtectedTransport
- Previous-Session
- Smartcard
- Smartcard-PKI
- Software-PKI
- Time-Sync-Token

8 Choose from the available Trusted Providers and add to the entity.

The list contains configured entities that have been populated with service providers.

9 Modify values for the Service Provider attributes.

Service Provider attributes define general information regarding the service provider.

Assertion Consumer URL
: Type the URL to the end point which will receive all SAML assertions.

Assertion Consumer Service URL ID
: If the value of the Protocol Support Enumeration common attribute is `urn:liberty:iff:2003-08`, type the required ID.

Set Assertion Consumer Service URL as Default
: Select the check box to use the Assertion Consumer Service URL as the default value when no identifier is provided in the request.

Sign Authentication Request
: Select the check box to make the service provider always signs authentication requests.

Name Registration after Federation
Select the check box to enable the service provider to participate in name registration after a principal has been federated.

Name ID Policy
Select the option permitting requester influence over name identifier policy at the identity provider. The options include:

None	The identity provider will return the name identifier(s) corresponding to the federation that exists between the identity provider and the requesting service provider or affiliation group for the principal. If no such federation exists, an error will be returned.
One-time	The identity provider will issue a temporary, one-time-use identifier for the principal after federation.
Federation	The identity provider may start a new identity federation if one does not already exist for the principal.

Enable Affiliation Federation
Select the check box to enable affiliation federation.

10 Modify values for the Hosted Configuration attributes.

Hosted Configuration attributes define general information regarding the provider hosted on the same machine as Federation Manager.

Provider Alias
Type an alias name for the local identity provider.

Authentication Type
Select the provider that should be used for authentication requests from a provider hosted locally.

- *Remote* specifies that the provider hosted locally would contact a remote identity provider upon receiving an authentication request.
- *Local* specifies that the provider hosted locally should contact a local identity provider upon receiving an authentication request (essentially, itself).

Default Authentication Context
Select the authentication context class (method of authentication) to use if the identity provider does not receive this information as part of a service provider request. This value also specifies the authentication context used by the service provider when an unknown user tries to access a protected resource. The options are as follows:

- Password
- Mobile Digital ID
- Smartcard
- Smartcard-PKI
- MobileUnregistered

- Software-PKI
- Previous-Session
- Mobile Contract
- Time-Sync-Token
- Password-ProtectedTransport

Identity Provider Forced Authentication
: Select the check box to indicate that the identity provider must reauthenticate the principal (even if the principal has an existing session from a prior authentication) when an authentication request is received from a remote service provider. This attribute is enabled by default.

Request Identity Provider to be Passive
: Select the check box to specify that the identity provider must not prompt a user for authentication credentials upon receiving an authentication request from a remote service provider. The default (unchecked) is to authenticate the user upon receiving an authentication request.

Organization DN
: Type the value of the organization's distinguished name.

Liberty Version URI
: Type the URI of the version of the Liberty Alliance Project specification being used. The default value is `http://projectliberty.org/specs/v1`.

Name Identifier Implementation
: This field defines the class used by a service provider to participate in name registration. Name registration is a profile by which service providers specify a principal's name identifier that an identity provider will use when communicating to the service provider. The value is `com.sun.identity.federation.services.util.FSNameIdentifierImpl`.

Home Page URL
: Type the URL of the home page of the identity provider.

Single Sign-on Failure Redirect URL
: Type the URL to which a principal will be redirected if single sign-on has failed.

Auto Federation
: Select the check box to enable auto federation.

Auto Federation Common Attribute Name
: When creating an Auto Federation Attribute Statement, the value of this attribute will be used. The statement will contain `AutoFedAttribute` as the attribute name and this common attribute as the value.

Attribute Statement Plugin
: Specify a pluggable class used for adding attribute statements to an assertion that is generated during the Liberty-based single sign-on process.

User Provider Implementation Class Name
: Specifies a pluggable implementation to store and retrieve the user attribute information from the users data store. The default implementation of the `com.sun.identity.federation.accountmgmt.FSUserProvider` interface is the `com.sun.identity.federation.accountmgmt.DefaultFSUserProvider` class.

Service Provider Adapter Implementation Class Name
: Specifies a pluggable implementation of the `com.sun.identity.federation.plugins.FederationSPAdapter` interface. The implemented class allows applications to customize their actions before and after invoking the federation protocols. For example, a service provider may want to choose to redirect to a specific location after single sign-on. There is no default implementation but the `spi` sample included with Federation Manager makes use of the class.

Configuration for Service Provider Adapter Implementation
: Stores configuration information that may be used to initialize the Service Provider Adapter Implementation Class Name. The usage of this attribute is also demonstrated in the `spi` sample application.

11 Provide information for the Proxy Authentication Attributes.

Proxy Authentication Configuration attributes define values for dynamic identity provider proxying.

Enable Proxy Authentication
: Select the check box to enable proxy authentication for a service provider.

Proxy Identity Providers List
: Add a list of identity providers that can be used for proxy authentication. The value is a URI defined as the provider's identifier.

Maximum Number of Proxies
: Type the maximum number of identity providers that can be proxied.

Use Introduction Cookie for Proxying
: Select the check box if you want introductions to be used to find the proxying identity provider.

12 Provide information for the SAML Attributes.

SAML Attributes define general information regarding SAML assertions sent by the identity provider.

Assertion Interval
: Type the interval of time (in seconds) for which an assertion issued by the identity provider will remain valid. A principal will remain authenticated until the assertion interval expires.

Cleanup Interval
: Type the interval of time (in seconds) before assertions stored in the identity provider will be cleared.

Artifact Timeout
Type the interval of time (in seconds) to specify the time out for assertion artifacts.

Assertion Limit
Type a number to define the amount of assertions an identity provider can issue, or the number of assertions that can be stored.

13 Provide values for the Organizations Profile attributes.

The Organizations Profile attributes provide basic information that may be required when interacting with a principal. These attributes are optional.

Names
Type the complete legal name of the entity's organization. Use the format *locale*|*organization-name*. For example, `en|`*organization-name*`.com`.

Note – If the Names attribute contains a value, it is required to add values to the Display Names and URL attributes.

Display Names
Type a name that is suitable for display. Use the format *locale*|*organization-display-name*. For example, `en|`*organization-display-name*`.com`.

URL
Type a URL that can be used to direct a principal to additional information on the entity's organization. Use the format *locale*|*organization-URL*. For example, `en|http://www.`*organization-name*`.com`.

14 (Optional) To configure Contact Persons for the provider, click New Contact Person.

See "To Add a Contact Person to a Provider in a Provider Entity" on page 189.

15 Click Save to complete the configuration.

▼ To Modify Remote Service Provider Attributes in a Provider Entity

After creating an entity and adding a service provider, you can edit the service provider profile. In a provider entity, this might entail adding metadata that was not available to configure when originally adding the service provider. The starting point is the Entity Descriptors screen under Federation.

1 Click on the name of a configured provider entity to modify its profile.

The entity's profile page is displayed.

2 Select Service Provider from the View menu.

3 Modify values for the Common Attributes.

Common Attributes contain values that generally define the identity provider itself.

Provider Type
: The static value of this attribute defines whether this is a hosted or remote provider.

Description
: This attribute contains the description provided when you created the entity. You can modify the description originally entered.

Valid Until
: Type the expiration date for the metadata pertaining to the provider. Use Coordinated Universal Time (UTC) in the format *yyyy-mm-dd*T*hh:mm:ss.S*Z. For example, `2004-12-31T12:30:00.0Z`.

Cache Duration
: Type the maximum amount of time the entity can be cached. The value is defined in the format P*n*Y*n*M*n*DT*n*H*n*M*n*S, where *n* is an integer. For example, `P1Y2M4DT9H8M20S` defines the cache duration as 1 year, 2 months, 4 days, 9 hours, 8 minutes, and 20 seconds.

Protocol Support Enumeration
: Choose the protocol release supported by this entity.

  - `urn:liberty:iff:2003-08` refers to the *Liberty Identity Federation Framework version 1.2*.
  - `urn:liberty:iff:2002-12` refers to the *Liberty Identity Federation Framework version 1.1*.

Server Name Identifier Mapping Binding
: Name identifier mapping allows a service provider to obtain a name identifier for a principal that has federated in the namespace of a different service provider. Implementing this protocol allows the requesting service provider to communicate with the second service provider without an identity federation having been enabled. Type a URI that identifies the communication specifications.

Note – Currently, the Name Identifier Mapping profile only supports SOAP. If this attribute is used, its value must be `http://projectliberty.org/profiles/nim-sp-http`.

Additional Meta Locations
: Type a URL that points to other relevant metadata concerning the provider.

Signing Key: Key Alias
: Type the key alias used to sign requests and responses.

Encryption Key: Key Alias
Type the security certificate alias. Certificates are stored in a Java keystore file. Each specific certificate is mapped to an alias that is used to fetch the certificate.

Encryption Key: Key Size
Type the length for keys used by the web service consumer when interacting with another entity.

Encryption Key: Encryption Method
Choose the method of encryption. The choices include:

- None
- AES
- DES
- 3DES

Name Identifier Encryption
Select the check box to enable encryption of the name identifier.

4 Modify values for the Communication URLs attributes.

Communication URLs attributes contain locations for sending transmissions to the service provider being configured.

SOAP Endpoint
Type a URI to the identity provider's SOAP message receiver. This value communicates the location of the SOAP receiver in non browser communications.

Single Logout Service
Type a URL to which service providers can send logout requests. Single logout synchronizes the logout functionality across all sessions authenticated by the identity provider.

Single Logout Return
Type a URL to which the identity provider will redirect the principal after completing a logout.

Federation Termination Service
Type a URL to which a service provider will send federation termination requests.

Federation Termination Return
Type a URL to which the identity provider will redirect the principal after completing federation termination.

Name Registration Service
Type a URL to which a service provider will send requests to specify the name identifier that will be used when communicating with the identity provider about a principal. This service can only be used after a federation session is established.

Name Registration Return
Type a URL to which the identity provider will redirect the principal after HTTP name registration has been completed.

5 Modify values for the Communication Profiles attributes.

Communication Profiles attributes define the transmission methods used by the service provider.

Federation Termination
Select a profile to notify other providers of a principal's federation termination:

- HTTP Redirect
- SOAP

Single Logout
Select a profile to notify other providers of a principal's logout:

- HTTP Redirect
- HTTP Get
- SOAP

Name Registration
Select a profile to notify other providers of a principal's name registration:

- HTTP Redirect
- SOAP

Single Sign-on/Federation
Select a profile for sending authentication requests:

- Browser Post (specifies a browser-based HTTP POST protocol)
- Browser Artifact (specifies a non-browser SOAP-based protocol)
- LECP (specifies a Liberty-enabled Client Proxy)

Note – Federation Manager can handle requests that come from a Liberty-enabled client proxy profile, but it requires additional configuration that is beyond the scope of this manual.

---

6 Select any of the available authentication domains to assign to the provider.

A provider can belong to one or more authentication domains. However, a provider without a specified authentication domain can not participate in Liberty-based communications. If no authentication domains have been created, this attribute can be defined later.

7 Modify values for the Service Provider attributes.

Service Provider attributes define general information regarding the service provider.

Assertion Consumer URL
Type the URL to the end point which will receive all SAML assertions.

Assertion Consumer Service URL ID
If the value of the Protocol Support Enumeration common attribute is `urn:liberty:iff:2003-08`, type the required ID.

Set Assertion Consumer Service URL as Default
Select the check box to use the Assertion Consumer Service URL as the default value when no identifier is provided in the request.

Sign Authentication Request
Select the check box to make the service provider always signs authentication requests.

Name Registration after Federation
Select the check box to enable the service provider to participate in name registration after a principal has been federated.

Name ID Policy
Select the option permitting requester influence over name identifier policy at the identity provider. The options include:

None	The identity provider will return the name identifier(s) corresponding to the federation that exists between the identity provider and the requesting service provider or affiliation group for the principal. If no such federation exists, an error will be returned.
One-time	The identity provider will issue a temporary, one-time-use identifier for the principal after federation.
Federation	The identity provider may start a new identity federation if one does not already exist for the principal.

Enable Affiliation Federation
Select the check box to enable affiliation federation.

8 Provide information for the Proxy Authentication Attributes.

Proxy Authentication Configuration attributes define values for dynamic identity provider proxying.

Enable Proxy Authentication
Select the check box to enable proxy authentication for a service provider.

Proxy Identity Providers List
Add a list of identity providers that can be used for proxy authentication. The value is a URI defined as the provider's identifier.

Maximum Number of Proxies
Type the maximum number of identity providers that can be proxied.

Use Introduction Cookie for Proxying
Select the check box if you want introductions to be used to find the proxying identity provider.

9 Provide values for the Organizations Profile attributes.

The Organizations Profile attributes provide basic information that may be required when interacting with a principal. These attributes are optional.

Names
: Type the complete legal name of the entity's organization. Use the format *locale*|*organization-name*. For example, `en|`*organization-name*`.com`.

Note – If the Names attribute contains a value, it is required to add values to the Display Names and URL attributes.

Display Names
: Type a name that is suitable for display. Use the format *locale*|*organization-display-name*. For example, `en|`*organization-display-name*`.com`.

URL
: Type a URL that can be used to direct a principal to additional information on the entity's organization. Use the format *locale*|*organization-URL*. For example, `en|http://www.`*organization-name*`.com`.

10 (Optional) To configure Contact Persons for the provider, click New Contact Person.

See "To Add a Contact Person to a Provider in a Provider Entity" on page 189.

11 Click Save to complete the configuration.

▼ To Add a Contact Person to a Provider in a Provider Entity

Information regarding a contact person for a particular service provider or identity provider can be added to the provider's profile in a provider entity. The starting point is the Entities configuration screen in the Federation module.

1 Click on the name of a configured entity to modify its profile.

The entity's profile page is displayed.

2 Select either Identity Provider or Service Provider from the View menu.

3 Click New Contact Person under the Contact Persons attribute group.

4 Provide values for the New Contact Person attributes.

First Name
: Type the given name of the entity's contact person.

Last Name
: Type the surname of the entity's contact person.

Type
Choose the type of contact from the drop-down menu:

- Administrative
- Billing
- Technical
- Other

Company
Type the name of the company that employs this person.

Liberty Principal ID
Type a URI that points to an online instance of the contact person's personal information profile.

Emails
Type one or more email addresses for the contact person.

Telephone Numbers
Type one or more telephone numbers for the contact person.

5 Click OK to complete the Contact Person configuration.

6 Click Save on the Entity Profile page to complete the entity configuration.

▼ To Modify a Contact Person Profile in a Provider Entity

Information regarding a contact person for a particular service provider or identity provider can be added to the provider's profile in a provider entity. The starting point is the Entities configuration screen in the Federation module.

1 Click on the name of a configured entity to modify its profile.

2 Select either Identity Provider or Service Provider from the View menu.

3 Click on the name of a configured Contact Person to modify its profile.

4 Provide new or modified values for the Contact Person profile.

First Name
Type the given name of the entity's contact person.

Last Name
Type the surname of the entity's contact person.

Type
Choose the type of contact from the drop-down menu:

- Administrative

- Billing
- Technical
- Other

Company
: Type the name of the company that employs this person.

Liberty Principal ID
: Type a URI that points to an online instance of the contact person's personal information profile.

Emails
: Type one or more email addresses for the contact person.

Telephone Numbers
: Type one or more telephone numbers for the contact person.

5 Click Save to complete the Contact Person configuration.

6 Click Save on the Entity Profile page to complete the entity configuration.

▼ To Create the Affiliation in an Affiliate Entity

After editing the General attributes of an affiliate entity, you can create the affiliation. This might entail modifying the defined Common attributes, or adding members to the affiliation. The starting point is the Entity Descriptors screen under Federation.

1 Click on the name of a configured affiliate entity to modify its profile.

The entity's profile page is displayed.

2 Select Affiliate from the View menu.

3 Modify values for the Common Attributes.

Affiliate ID
: Type a new value for the URL identifier of the affiliation, if applicable. The value of this attribute is the name provided when you created the entity.

Affiliate Owner ID
: Type a new value for the Provider ID of the owner or parent operator of the affiliation. This value points to additional affiliation metadata.

Valid Until
: Type the expiration date for the metadata pertaining to the provider. Use Coordinated Universal Time (UTC) in the format *yyyy-mm-dd*T*hh:mm:ss.S*Z. For example, `2004-12-31T12:30:00.0Z`.

Cache Duration
Type the maximum amount of time the entity can be cached. The value is defined in the format P*n*Y*n*M*n*DT*n*H*n*M*n*S, where *n* is an integer. For example, `P1Y2M4DT9H8M20S` defines the cache duration as 1 year, 2 months, 4 days, 9 hours, 8 minutes, and 20 seconds.

Signing Key: Key Alias
Type the key alias used to sign requests and responses.

Encryption Key: Key Alias
Type the security certificate alias. Certificates are stored in a Java keystore file. Each specific certificate is mapped to an alias that is used to fetch the certificate.

Encryption Key: Key Size
Type the length for keys used by the web service consumer when interacting with another entity.

Encryption Key: Encryption Method
Choose the method of encryption. The choices include:

- None
- AES
- DES
- 3DES

4 **Add members to the affiliation from those available.**

5 **Click Save.**

Authentication Domains

An *authentication domain* is a federation of any number of service providers (and at least one identity provider) with whom principals can transact business in a secure and apparently seamless environment. (An authentication domain is NOT a domain in the domain name system sense of the word.) In Federation Manager, before creating and populating an authentication domain, you create a grouping of providers called an *entity*. Then, you configure and save the authentication domain itself. Finally, to add an entity, you edit the configured authentication domain.

Note – For more information on entities, see "Entities: Provider and Affiliate" on page 137.

Authentication domains are configured using the Federation Manager Console by selecting Authentication Domains under Federation. The following tasks are associated with authentication domains:

- "To Create a New Authentication Domain" on page 193

- "To Modify an Authentication Domain Profile" on page 193
- "To Add Providers to an Authentication Domain" on page 194
- "To Delete an Authentication Domain Profile" on page 195

Note – The members of the domain must have previously established a *circle of trust* based on the Liberty Alliance Project architecture and operational agreements.

▼ To Create a New Authentication Domain

Follow this procedure to create a new authentication domain. The starting point is the Authentication Domains screen under Federation.

1 **Click New to display the authentication domain attributes.**

The New Authentication Domain profile page is displayed.

2 **Type a name for the authentication domain.**

3 **(Optional) Type a description of the authentication domain in the Description field.**

4 **(Optional) Type a value for the Writer Service URL.**

The Writer Service URL specifies the location of the servlet that writes the common domain cookie. Use the format `http://`*common-domain-host*`:`*port*`/common/writer`.

5 **(Optional) Type a value for the Reader Service URL.**

The Reader Service URL specifies the location of the servlet that reads the common domain cookie. Use the format `http://`*common-domain-host*`:`*port*`/common/transfer`.

6 **Choose Active or Inactive.**

The default status is Active. Choosing Inactive disables communication within the authentication domain.

7 **Click OK to complete the configuration.**

The new authentication domain is displayed on the Authentication Domains screen.

▼ To Modify an Authentication Domain Profile

Follow this procedure to edit the General attributes of an existing authentication domain, or to add providers to it. (See "To Add Providers to an Authentication Domain" on page 194.) The starting point is the Authentication Domains screen under Federation.

1. **Select the name of a configured authentication domain to modify its profile, or to add providers to it.**

   The Authentication Domain Profile page is displayed.

2. **Edit the values of the authentication domain's General attributes.**

   Name
   : Contains the name for the authentication domain. This value is static.

   Description
   : Contains a description of the authentication domain.

   Writer Service URL
   : Specifies the location of the service that writes the common domain cookie. Use the format `http://`*common-domain-host*`:`*port*`/common/writer`.

   Reader Service URL
   : Specifies the location of the service that reads the common domain cookie. Use the format `http://`*common-domain-host*`:`*port*`/common/transfer`.

   Status
   : The default status is Active. Selecting Inactive disables communication within the authentication domain.

3. **Click Add to add providers to the authentication domain.**

   The Trusted Partner page is displayed with a list of available provider entities.

4. **Choose one or more available providers and click the Add arrow to select them.**

   The list provided contains the names of entities that have been created. These entities contain providers. For more information, see "To Add Providers to an Authentication Domain" on page 194.

5. **Click OK to save the providers to the authentication domain.**

   This will return you to the previous Authentication Domain Profile screen.

6. **Click Save to complete the operation.**

▼ To Add Providers to an Authentication Domain

Identity providers and service providers must first be configured within entities before they are available to add to an authentication domain. Once created and populated with providers, an entity (and thus the providers) can be assigned to an authentication domain.

Note – An entity cannot be assigned to an authentication domain until it has been populated with provider(s).

For more information, see "Entities: Provider and Affiliate" on page 137.

1. **Choose one or more available providers and click the Add arrow to select them.**

   The list provided contains the names of entities that have been created. These entities contain providers.

2. **Click OK to save the providers to the authentication domain.**

   This will return you to the previous Authentication Domain Profile screen.

3. **Finish your configurations and click Save to complete the operation.**

▼ To Delete an Authentication Domain Profile

Follow this procedure to delete an existing authentication domain. The starting point is the Authentication Domains screen under Federation.

1. **Check the box next to the name of the authentication domain you want to delete.**

2. **Click Delete.**

   Deleting an authentication domain does not delete the providers that belong to it.

Auto-Federation

The auto-federation feature in Federation Manager will automatically federate a user's disparate provider accounts based on a common attribute. This common attribute will be exchanged in a single sign-on assertion so that the consuming service provider can identify the user and create account federations. If auto-federation is enabled and it is deemed that a user at provider A and a user at provider B have the same value for the defined common attribute (for example, `emailaddress`), the two accounts will be federated automatically without principal interaction.

Note – Auto-federating a principal's two distinct accounts at two different providers requires each provider to have agreed to implement support for this functionality beforehand.

▼ To Enable Auto-Federation

Ensure that each local service and identity provider participating in auto federation is configured for it . Remote providers would not be configured in your deployment.

1. **In the Federation Manager Console, click the Federation tab.**
2. **Under Federation, select the Entities tab.**
3. **Select the name of a hosted provider entity to edit its profile.**
   Whether an entity is configured to hold hosted or remote providers is not information that is disclosed on this screen.
4. **Select Identity Provider or Service Provider from the View menu.**
5. **Select Hosted Configuration.**
6. **Enable Auto Federation by checking the box.**
7. **Type a value for the Auto Federation Common Attribute Name attribute.**
   For example, enter `emailaddress` or `userID`. You should be sure that each participating user profile (at both providers) has a value for this attribute.
8. **Click Save to complete the configuration.**

Bulk Federation

Federation Manager provides a script for federating user accounts in bulk. It is called `ambulkfed` and is located in `/FederationManager-base/SUNWam/bin`. The script assumes that the user database is LDAPv3–compliant.

Note – The `ambulkfed` script is the primary script for bulk federation. It uses two other Perl scripts, `amGenerateLDIF.pl` and `amGenerateNI.pl`, behind the scenes.

As input, `ambulkfed` takes a file that maps the user distinguished name (DN) of the identity provider to the user DN of the service provider. Each line of the file must place the mappings in the following order and separated by a pipe ("|"): `uid=spuser,dc=iplanet,dc=com | uid=idpuser,dc=iplanet,dc=com`. The script generates unique random identifiers for each mapping and creates four files:

- `spnameidentifiers.txt`
- `idpnameidentifiers.txt`

- `spuserdata.ldif`
- `idpuserdata.ldif`

These files contain the data for bulk federation. The LDIFs are used for instances of LDAPv3–compliant data stores. `ambulkfed` generates and loads the LDIF data based on its given provider role. For example, it will load `spuserdata.ldif` because Federation Manager acts as a service provider. The LDIFs will also be stored locally and can be used with `ldapmodify` to load the data into a remote provider. If the remote provider is not an instance of Federation Manager, the generated text files `spnameidentifiers.txt` and `idpnameidentifiers.txt` can be used to generate federation data based on the input needs of the provider.

◆ ◆ ◆ CHAPTER 8

SAML Administration

Sun Java System Federation Manager uses the Security Assertion Markup Language (SAML) as a means for exchanging security information.

This chapter covers the following topics:

SAML Overview

Security Assertion Markup Language (SAML) is an open-standard protocol that uses an XML framework to exchange security information between an *authority* and a *trusted* partner site. The security information concerns itself with a subject's authentication status, access authorization and attribute information. (A person identified by an email address is a subject as might be a printer.) A SAML authority (in some instances, referred to as the asserting party) is a platform or application that has been integrated with SAML API, allowing it to relay security information. For example, asserting that a subject has been authenticated into its system by passing the required attributes. Trusted partner sites receive the security information and rely on its authenticity.

Note – For more information about the SAML implementation (including definitions of the profile types and assertions), see Chapter 9, "SAML Administration," in *Sun Java System Access Manager 7 2005Q4 Federation and SAML Administration Guide*.

The following SAML attributes can be configured for your implementation:

The following tasks are associated with configuring SAML:

SAML Service Attributes

The following sections describe the attributes for the SAML service. The attributes are:

Target Specifier

This attribute assigns a name to the destination site URL used in SAML redirects. The default is TARGET. Only sites configured in the Trusted Partners attribute can be specified as a TARGET.

Site Identifiers

This attribute defines any site hosted by the server on which Federation Manager is installed. A default value and an automatically generated Site ID are defined for the host itself during

installation (with values retrieved from `AMConfig.properties`). Multiple entries are possible. For example, load balancing or multiple instances of Federation Manager sharing the same Directory Server would all need to be defined. To configure a new Site Identifier, click New and see "To Configure a Site Identifier" on page 201.

To Configure a Site Identifier

This attribute defines any site hosted by the server on which Federation Manager is installed. A default value and an automatically generated Site ID are defined for the host during installation (with values retrieved from `AMConfig.properties`). Multiple entries are possible. For example, load balancing or multiple instances of Federation Manager sharing the same Directory Server would all need to be defined. The starting point is the Site Identifiers attribute on the SAML screen under Federation.

1. **In the Federation Manager Console, click the SAML tab.**

2. **Under Site Identifiers, click New to add a new site identifier or click on the name of a configured site identifier to modify its profile.**

   The Site Identifier attributes are displayed.

3. **Provide values for the Site Identifier attributes based on the following information:**

   Name
   : The value of this property is *protocol*`://`*host*`:`*port*.

     If configuring SAML for SSL (in both the source and destination site), ensure that the protocol defined here is `https//`.

   Site ID
   : The site ID is an identifier generated for each site (although the value will be the same for multiple servers behind a load balancer). There is a class in the `com.sun.identity.saml.common` package that can be used to generate this identifier manually, if needed. Type the following at the command line:

     ```
     % java -classpath FederationManager-base/SUNWam/fm/web-src/WEB-INF/lib/am_services.jar
     com.sun.identity.saml.common.SAMLSiteID protocol://host:port
     ```

   Issuer Name
   : The default value of this property is *host*`:`*port*, but it could be a URI.

4. **Click OK to complete the Site Identifier configuration.**

5. **Click Save on the SAML Profile page to complete the SAML configuration.**

Trusted Partners

This attribute defines any trusted partner (remote to the server on which Federation Manager is installed) that will be communicating with Federation Manager.

Note – The trusted partner site must have a prearranged trust relationship with one or more of the sites configured in the Site Identifiers attribute.

To configure a new Trusted Partner, click New and see "Trusted Partners: Selecting Partner Type and Profile" on page 202.

▼ Trusted Partners: Selecting Partner Type and Profile

This attribute defines any trusted partner (remote to the server on which Federation Manager is installed) that will be communicating with Federation Manager.

Note – The trusted partner site must have a prearranged trust relationship with one or more of the sites configured in the Site Identifiers property.

The first step in configuring a trusted partner is to determine the partner's role in the trust relationship. A trusted partner can be a source site (one that generates a single sign-on assertion) or a destination site (one that consumes a single sign-on assertion). For example, if the partner is the source site, this attribute is configured based on how it will send assertions. If the partner is the destination site, this attribute is configured based on the profile in which it will be receiving assertions. Following is the first part of the procedure for configuring a trusted partner. The starting point is the SAML configuration screen of the Federation module.

Note – To edit or duplicate the attributes of a trusted partner profile, click the appropriate button in the Actions column next to the configured trusted partner name.

---

1. **In the Federation Manager Console, click the SAML tab.**
2. **Under Trusted Partners, click New.**

   The Trusted Partner profile page is displayed.

Note – Click Edit to change the role of the partner site if you are modifying an existing trusted partner.

---

3 **Select the role (Destination or Source) of the partner site you are configuring by checking the appropriate profile(s) used to communicate with it.**

You may choose *Web Browser Artifact Profile* or *Web Browser Post Profile* for either Destination, Source or both, or *SOAP Query* for Destination only. The choices made dictate which of the attributes in the following steps need to be configured.

4 **Click Next.**

5 **To complete the configuration, see "Trusted Partners: Configuring Trusted Partner Attributes" on page 203.**

Trusted Partners: Configuring Trusted Partner Attributes

Following is the second part of the procedure for configuring a trusted partner. The starting point is "Trusted Partners: Selecting Partner Type and Profile" on page 202. Based on the role(s) selected in the first part, any of the sub-attributes listed in the following sections may need to be defined for the Trusted Partner.

Note – If you reached this page by clicking Edit or Duplicate on the SAML configuration screen under Federation, modify the trusted partner profile based on the steps below and click Save to change the values. Click Save on the SAML Profile page to complete the modification.

---

1 **Type in values for the Common Settings subattributes.**

Source ID
: This is a 20 byte sequence (encoded using the Base64 format) that comes from the partner site. It is generally the same value as that used for the Site ID attribute when configuring the Site Identifiers attribute.

Target
: This is the domain of the partner site (with or without a port number). If you want to contact a web page that is hosted in this domain, the redirect URL is picked up from the values defined in .

Note – If there are two defined entries for the same domain (one containing a port number and one without a port number), the entry with the port number takes precedence. For example, assume the following two trusted partner definitions: `target=sun.com` and `target=sun.com:8080`. If the principal is seeking `http://machine.sun.com:8080/index.html`, the second definition will be chosen.

Site Attribute Mapper
: The class is used to return a list of attribute values defined as `AttributeStatements` elements in an Authentication Assertion. A site attribute mapper needs to be implemented from the `com.sun.identity.saml.plugins.PartnerSiteAttributeMapper` interface.

If no class is defined, no attributes will be included in the assertion.

Version
: The SAML version used (`1.0` or `1.1`) to send SAML requests. If this parameter has no value, the following default values (defined in `AMConfig.properties`) are used:

- `com.example.identity.saml.assertion.version=1.1`
- `com.example.identity.saml.protocol.version=1.1`

Account Mapper
: The class that defines how the subject of an assertion is related to an identity at the destination site. An account mapper needs to be implemented from the `com.sun.identity.saml.plugins.PartnerAccountMapper` interface. If no class is specified, a default account mapper implementation will be used. This default implementation assumes that two sites have the same directory structure. For example, the root suffix, and user IDs are the same.

Certificate
: A certificate alias that is used to verify the signature in an assertion when it is signed by the partner and the certificate cannot be found in the `KeyInfo` portion of the signed assertion.

Host List
: A list of the IP addresses, the DNS host name, or the alias of the client authentication certificate used by the partner. This is configured for all hosts within the partner site that can send requests to this authority. This list helps to ensure that the requestor is indeed the intended receiver of the artifact. If the requester is defined in this list, the interaction will continue. If the requester's information does not match any hosts defined in the host list, the request will be rejected.

Issuer
: The creator of a generated assertion. The default syntax is *hostname*:*port*.

2 Type in values for the Destination subattributes.

Artifact: SAML URL
: The URL that points to the servlet that implements the Web Browser Artifact Profile.

Post: Post URL
: The URL that points to the servlet that implements the Web Browser POST Profile.

SOAP Query: Attribute Mapper
: The class that is used to obtain single sign-on information from a query. You need to implement an attribute mapper from the included interface. If no class is specified, the `DefaultAttributeMapper` will be used.

SOAP Query: Action Mapper
: The class that is used to get single sign-on information and map partner actions to authorization decisions. You need to implement an action mapper from the included interface. If no class is specified, the `DefaultActionMapper` will be used.

3 Type in values for the Source subattributes.

Artifact: SOAP URL
The URL to the SAML SOAP Receiver.

Authentication Type
Authentication types that can be used with SAML:

- `NOAUTH`
- `BASICAUTH`
- `SSL`
- `SSLWITHBASICAUTH`

This attribute is optional. If not specified, the default is `NOAUTH`. If `BASICAUTH` or `SSLWITHBASICAUTH` is specified, the Trusted Partners attribute is required and should be `HTTPS`.

User
When `BASICAUTH` is chosen as the Authentication Type, the value of this attribute defines the user identifier of the partner being used to protect the partner's SOAP receiver.

User's Password
When `BASICAUTH` is chosen as the Authentication Type, the value of this attribute defines the password for the user identifier of the partner being used to protect the partner's SOAP receiver.

User's Password (reenter)
Reenter the password defined previously.

4 Click Finish.

5 Click Save on the SAML Profile page to complete the configuration.

Target URLs

If the TARGET URL received via either profile is listed as a value of this attribute, the received assertion(s) will be sent to the TARGET URL via an HTTP FORM POST.

Note – To edit or duplicate the attributes of a trusted partner profile, click the appropriate button in the Actions column next to the configured trusted partner name.

▼ To Configure a Target URL

If the TARGET URL received via either SAML profile is listed as a value of this attribute, the assertion(s) received will be sent to the TARGET URL via an HTTP FORM POST. The following subattributes can be defined (or modified) for each URL which will receive assertions via POST.

1. **In the Federation Manager Console, click the SAML tab.**
2. **Under Target URLs, click New to add a new target URL or Edit to modify an existing one.**
   The Add New Post to Target URL page is displayed.
3. **Type values for the attributes.**
   Protocol
   Choose either `http` or `https`.
   Server Name
   The name of the server on which the TARGET URL resides, such as `www.sun.com`.
   Port
   This attribute contains the port number such as `58080`.
   Path
   This attribute contains the URI such as `/amserver/console`.
4. **Click OK to complete the Target URL configuration.**
5. **Click Save on the SAML Profile page to complete the SAML configuration.**

Assertion Timeout

This attribute specifies the number of seconds before a timeout occurs on an assertion. The default is 420.

Assertion Skew Factor for notBefore Time

This attribute is used to calculate the `notBefore` time of an assertion. For example, if `IssueInstant` is `2002-09024T21:39:49Z`, and `Assertion Skew Factor For notBefore Time` is set to 300 seconds (180 is the default value), the `notBefore` attribute of the conditions element for the assertion would be `2002-09-24T21:34:49Z`.

Note – The total valid duration of an assertion is defined by the values set in both the `Assertion Timeout` and `Assertion Skew Factor For notBefore Time` attributes.

Artifact Timeout

This attribute specifies the period of time an assertion that is created for an artifact will be valid. The default is 400.

SAML Artifact Name

This attribute assigns a variable name to a SAML artifact. The artifact is bounded-size data that identifies an assertion and a source site. It is carried as part of a URL query string and conveyed by redirection to the destination site. The default name is `SAMLart`. Using the default `SAMLart`, the redirect query string could be `http://`*host*:*port*/*deploy-URI*`/SamlAwareServlet?TARGET=`*target-URL*`/&SAMLart=artifact123`.

Sign SAML Assertion

This attribute specifies whether all SAML assertions will be digitally signed (`XML DSIG`) before being delivered. Selecting the check box enables this feature.

Sign SAML Request

This attribute specifies whether all SAML requests will be digitally signed (`XML DSIG`) before being delivered. Selecting the check box enables this feature.

Sign SAML Response

This attribute specifies whether all SAML responses will be digitally signed (`XML DSIG`) before being delivered. Selecting the check box enables this feature.

Note – All SAML responses used by the Web Browser POST Profile will be digitally signed whether this option is enabled or not enabled.

Enabling SSL Communication between SAML Entities

The Secure Sockets Layer (SSL) protocol can be enabled in Federation Manager to secure communications between SAML entities for SOAP binding. In SAML transactions, when one SAML entity talks to a second SAML entity deployed on an SSL-enabled web container, the former is referred to as the *SSL client*, and the latter is referred to as the *SSL server*. In Federation Manager, a SAML entity uses a Java keystore for its database of *key entries* and *trusted certificate entries*. A *key entry* is used for cryptographic purposes and consists of an entity's identity and its private key. It is stored in a keystore. A *trusted certificate entry* is used when making decisions about trust and contains a public key and an entity's identity. It is generally stored in a separate keystore called a *truststore* (although both types of entries may be stored in one file). To enable SSL in Federation Manager you must perform several modifications; for example, pointing to the specific `.keystore` and `.truststore` files.

The Java Virtual Machine (JVM) running the SSL client needs to have two system properties set. You set JVM system properties with a `-D` option on the Java command line. The properties that need to be set are:

- `-Djavax.net.ssl.trustStoreType=JKS`
- `-Djavax.net.ssl.trustStore=`*full-path-to-truststore-file* where *full-path-to-truststore-file* is the full path to a Java `.truststore` file that contains the server side Certificate Authority (CA) certificate(s).

Note – A *keystore* file can be manually created using the keytool utility which comes with the Java Development Kit (JDK). Following this, the CA certificate can be imported into the file, again using keytool.

If the client is going to present a certificate to the server, the following SSL-related configurations must also be performed. On the client side, the following three properties need to be set.

- `-Djavax.net.ssl.keyStore=`*full-path-to-keystore-file*

Note – The `keyStore` should contain the client certificate. This keystore could be the same as the `trustStore` previously defined.

- `-Djavax.net.ssl.keyStoreType=JKS`
- `-Djavax.net.ssl.keyStorePassword=`*password-for-keystore-file*

On the server side, the client certificate needs to be imported to the server side keystore. The server side keystore shares the same file as the XML signature keystore, and follows the same rule that both passwords should be encrypted using ampassword utility. The location and passwords of the server side keystore are specified by the following properties in the `AMConfig.properties` file:

- `com.sun.identity.saml.xmlsig.keystore=`
- `com.sun.identity.saml.xmlsig.storepass=`
- `com.sun.identity.saml.xmlsig.keypass=`

◆ ◆ ◆ CHAPTER 9

Web Services

Liberty-based web services are those based on specifications in the Liberty ID-WSF and the Liberty ID-SIS. They are accessible from the Federation Manager Console by clicking the Web Services tab. The implemented web services include:

Liberty Personal Profile Service

The *Liberty ID-SIS Personal Profile Service Specification* (Liberty ID-SIS—PP) describes a data service which provides an identity's basic profile information. It is intended to be the least common denominator for holding consumer-based information about a principal. Federation Manager has implemented this specification and developed the Liberty Personal Profile Service. The Liberty Personal Profile Service can be queried for identity data or this data can be updated.

Note – In order for access to occur, the hosting provider of the Liberty Personal Profile Service needs to be registered with the Discovery Service on behalf of each identity principal.

The following global attributes can be configured for your implementation of the Liberty Personal Profile Service.

The following tasks are associated with configuring the Liberty Personal Profile Service:

ResourceID Mapper

The value of this attribute specifies the implementation of the `com.sun.identity.liberty.ws.interfaces.ResourceIDMapper` interface. Although a new implementation can be developed, Federation Manager provides the default `com.sun.identity.liberty.ws.idpp.plugin.IDPPResourceIDMapper` which maps a discovery resource identifier to a user ID.

Authorizer

Note – This functionality is not supported.

Before processing a request, the Liberty Personal Profile Service will verify the authorization of the WSC making the request. There are two levels of authorization check:

1. Is the requesting entity authorized to access the requested resource profile information?
2. Is the requested resource published to the requestor?

Authorization occurs via a plug-in to the Liberty Personal Profile Service: an implementation of the `com.sun.identity.liberty.ws.interfaces.Authorizer` interface. Although a new implementation can be developed, Federation Manager provides the default class, `com.sun.identity.liberty.ws.idpp.plugin.IDPPAuthorizer`. This plug-in defines four policy action values for the query and modify operations:

- Allow

- Deny
- Interact For Consent
- Interact For Value

The resource values for the rules are similar to `x-path` expressions defined by the Liberty Personal Profile Service. For example, a rule can be defined like the example below.

EXAMPLE 9–1 Rules for Authorization

```
/PP/CommonName/AnalyzedName/FN    Query   Interact for consent
/PP/CommonName/*                  Modify  Interact for value
/PP/InformalName                  Query   Deny
```

Authorization can be turned off by deselecting one or both of the following attributes also defined in the Liberty Personal Profile Service:

- "Require Query PolicyEval" on page 214
- "Require Modify PolicyEval" on page 214

Attribute Mapper

This value of this attribute defines the class for mapping a Liberty Personal Profile Service attribute to an LDAP User attribute. By default, the class is `com.sun.identity.liberty.ws.idpp.plugin.IDPPAttributeMapper`.

Provider ID

The value of this attribute defines the unique identifier for this instance of the Liberty Personal Profile Service. The format is *protocol*`://`*host*`:`*port*/*deloy-uri*`/Liberty/idpp`.

Name Scheme

The value of this attribute defines the naming scheme for the Liberty Personal Profile Service common name. Choose First Last, or First Middle Last.

Namespace Prefix

The value of this attribute specifies the namespace prefix used for Liberty Personal Profile Service XML protocol messages. A *namespace* differentiates elements with the same name that come from different XML schemas. The Namespace Prefix is prepended to the element.

Supported Containers

The values of this attribute define a list of supported containers in the Liberty Personal Profile Service. A *container*, as used in this instance, is an attribute of the Liberty Personal Profile Service. For example, Emergency Contact and Common Name are two default containers for the Liberty Personal Profile Service.

To add a new container, click Add, and see "To Configure a Supported Container" on page 212.

Note – Currently, this functionality is not supported.

▼ To Configure a Supported Container

A *container* is an attribute that defines a holder for a piece of identity data. The following procedure is for adding new attributes to the Liberty Personal Profile Service. The starting point is the Liberty Personal Profile Service screen under Web Services.

1. **In the Federation Manager Console, click the Web Services tab.**
2. **Under Web Services, select the Personal Profile tab.**
3. **Under Supported Containers, click New or choose the name of a configured container to modify its profile.**

   The New Supported Container page is displayed.
4. **Provide values for the New Supported Container attributes.**

   Container Name
   : Enter a name for the container such as *CreditCard*.

   Plugin
   : Enter a class name to handle the whole container. This could be used to override the default implementation `com.sun.identity.liberty.ws.idpp.plugin.IDPPContainer`.
5. **Click OK to complete the Container configuration.**
6. **Click Save on the Liberty Personal Profile Service page to complete the service configuration.**

PPLDAP Attribute Map List

Each identity attribute defined in the Liberty Personal Profile Service maps one-to-one with an LDAP attribute. (For example, `JobTitle=sunIdentityServerPPEmploymentIdentityJobTitle` maps the Liberty `JobTitle` attribute to the `sunIdentityServerPPEmploymentIdentityJobTitle` attribute.) The value of

PPLDAP Attribute Map List is a list that specifies the mappings. The list is used by the attribute mapper defined in the "Attribute Mapper" on page 211 attribute which is, by default, `com.sun.identity.liberty.ws.idpp.plugin.IDPPAttributeMapper`.

In the following code sample, the Liberty Personal Profile Service `informalName` attribute mapping to the LDAP attribute `uid` is added to the mappings already present in the Liberty Personal Profile Service XML service file, `amLibertyPersonalProfile.xml`.

Note – Attribute mappings are defined as global attributes under the name `sunIdentityServerPPDSAttributeMapList` in `amLibertyPersonalProfile.xml`. This attribute corresponds to that `sunIdentityServerPPDSAttributeMapList` global attribute.

EXAMPLE 9–2 Attribute Mappings as Defined in XML Service File

```
<AttributeSchema name="sunIdentityServerPPDSAttributeMapList"
                      type="list"
                      syntax="string"
                      i18nKey="p108">
                      <DefaultValues>
                         <Value>CN=sunIdentityServerPPCommonNameCN</Value>
                         <Value>FN=sunIdentityServerPPCommonNameFN</Value>
                         <Value>MN=sunIdentityServerPPCommonNameMN</Value>
                         <Value>SN=sunIdentityServerPPCommonNameSN</Value>
                         <Value>InformalName=uid</Value>
              </AttributeSchema>
```

Note – When adding new attributes to the Liberty Personal Profile Service or the LDAP data store, ensure that the new attribute mappings are configured in the PPLDAP Attribute Map List attribute. See "To Configure an Attribute Mapping" on page 213.

▼ To Configure an Attribute Mapping

A *mapping* is an attribute that defines a holder for a piece of identity data. The following procedure is for adding new attributes to the Liberty Personal Profile Service. The starting point is the Liberty Personal Profile Service screen under Web Services.

1. **In the Federation Manager Console, click the Web Services tab.**
2. **Under Web Services, select the Personal Profile tab.**
3. **Under PPLDAP Attribute Map List, click Add or click on the name of a configured mapping to modify it.**

   The New LDAP Attribute Mapping page is displayed.

4 **Provide values for the container attributes.**

Name Prefix
Enter the name of the Liberty Personal Profile Service identity attribute to be mapped.

LDAP Attribute
Enter the name of the LDAP attribute to which the Name Prefix maps.

5 **Click OK to complete the Mapping configuration.**

6 **Click Save on the Liberty Personal Profile Service page to complete the service configuration.**

Require Query PolicyEval

If selected, this option requires a policy evaluation to be performed for Liberty Personal Profile Service queries.

Require Modify PolicyEval

If selected, this option requires a policy evaluation to be performed for Liberty Personal Profile Service modifications.

Extension Container Attributes

The Liberty Personal Profile Service allows you to specify extension attributes that are not defined in the Liberty Alliance Project specifications. The values of this attribute specify a list of extension container attributes. All extensions should be defined as:

```
/PP/Extension/PPISExtension [@name='extensionattribute']
```

The following sample illustrates an extension query expression for *creditcard*, an extension attribute.

EXAMPLE 9–3 Extension Query for creditcard

```
 /pp:PP/pp:Extension/ispp:PPISExtension[@name='creditcard']
Note: The prefix for the PPISExtension is different,
 and the schema for the PP extension is as follows:
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.sun.com/identity/liberty/pp"
  targetNamespace="http://www.sun.com/identity/liberty/pp">
  <xs:annotation>
      <xs:documentation>
```

EXAMPLE 9–3 Extension Query for creditcard *(Continued)*

```
      </xs:documentation>
  </xs:annotation>

  <xs:element name="PPISExtension">
     <xs:complexType>
        <xs:simpleContent>
           <xs:extension base="xs:string">
              <xs:attribute name="name" type="xs:string"
                use="required"/>
           </xs:extension>
        </xs:simpleContent>
     </xs:complexType>
   </xs:element>
</xs:schema>
```

Extension Attributes Namespace Prefix

The value of this attribute specifies the namespace prefix for the extensions defined in the Extension Container Attributes. This prefix is prepended to the element and is useful to distinguish metadata from different XML schema namespaces.

Is ServiceUpdate Enabled

The SOAP Binding Service allows a service to indicate that requesters should contact it on a different endpoint or use a different security mechanism and credentials to access the requested resource. If selected, this attribute affirms that there is an update to the service instance.

Service Instance Update Class

The value of this attribute specifies the default implementation class `com.sun.identity.liberty.ws.idpp.plugin.IDPPServiceInstanceUpdate`. This class is used to update the information for the service instance.

Alternate Endpoint

The value of this attribute specifies an alternate SOAP endpoint to which a SOAP request can be sent.

Alternate Security Mechanisms

This attribute allows you to choose a security mechanism. For more information on this functionality and the mechanisms, see the *Liberty ID-WSF Security Mechanisms* specification.

Discovery Service

The initial step in accessing identity data is to determine where the information is located. (For example, which identity service holds the principal's credit card information, or which server stores the principal's calendar service.) Typically, there are one or more services on a network that allow other entities to perform an action on identity data. Because clients are not expected to keep track of these services or to know which can be trusted, they require a discovery service.

A *discoverable web service* is assigned a service type URI in the specification that defines it. This URI points to the Web Services Description Language (WSDL) file that describes the service's data, the operations that can be performed on it, and a protocol detailing how to perform an action. The discoverable service specification itself adds the available ways the data can be exchanged. A *discovery service* is essentially a web service interface for discovery resources. A *discovery resource* is a registry of resource offerings. A *resource offering* defines an association between a piece of identity data and the service instance that provides access to that data. A *resource identifier* is a unique resource identifier (URI) registered with the Discovery Service that points to a particular discovery resource.

When a client sends a request for some type of data, it includes a resource identifier that the Discovery Service uses to locate the web services provider (WSP) for the requested attributes. The Discovery Service returns a resource offering that contains the information necessary to locate the data.

Note – In order for access to occur, the hosting provider of the Liberty Personal Profile Service needs to be registered with the Discovery Service on behalf of each identity principal.

The following Discovery Service global attributes can be configured for your implementation.

- "Encrypt NameIdentifier in Session Context for Bootstrapping" on page 220
- "Use Implied Resource; don't generate ResourceID for Bootstrapping" on page 220
- "Resource Offerings for Bootstrapping" on page 220

The following tasks are associated with configuring the Discovery Service:

- "To Configure a ResourceID Mapper" on page 219
- "To Configure a Resource Offering for Bootstrapping" on page 221
- "To Configure a Service Description" on page 222

Provider ID

This attribute takes a URI that points to the Discovery Service. Use the format *protocol*`://`*host*`:`*port*`/amserver/Liberty/disco`. This value can be changed as long as other relevant attributes values are changed to match the new location.

Supported Authentication Mechanisms

This attribute specifies the authentication methods supported by the Discovery Service. These security mechanisms refer to the way a web service consumer authenticates to the web service provider or provides message-level security. By default, all available methods that the service instance supports are selected. If an authentication method is not selected, and a web services consumer sends a request using that method, the request is rejected. See "To Configure a Service Description" on page 222.

Supported Directives

This attribute allows you to specify a policy-related directive for a resource. If a service provider wants to use an unsupported directive, the request will fail. The following table details the available options.

Directive	Purpose
`AuthenticateRequester`	The Discovery Service should include a SAML assertion (containing an `AuthenticationStatement`) in its responses to enable the client to authenticate to the service instance hosting the resource.
`AuthenticateSessionContext`	The Discovery Service should include a SAML assertion (containing a `SessionContextStatement`) in its responses that indicate the status of the session.

Directive	Purpose
`AuthorizeRequestor`	The Discovery Service should include a SAML assertion (containing a `ResourceAccessStatement`) in its responses that indicate whether the client is allowed to access the resource.
`EncryptResourceID`	The Discovery Service should encrypt the resource identifier in responses to all clients.
`GenerateBearerToken`	For use with Bearer Token Authentication, the Discovery Service should generate a token that grants the bearer permission to access the resource.

Enable Policy Evaluation for DiscoveryLookup

If enabled, the service will perform a policy evaluation for the `DiscoveryLookup` operation. By default, the option is not selected.

Enable Policy Evaluation for DiscoveryUpdate

If enabled, the service will perform a policy evaluation for the `DiscoveryUpdate` operation. By default, this option is not selected.

Authorizer Plugin Class

The value of this attribute is the name and path to the class that implements the `com.sun.identity.liberty.ws.interfaces.Authorizer` interface used for policy evaluation of a web services consumer. The default class is `com.sun.identity.liberty.ws.disco.plugins.DefaultDiscoAuthorizer`.

Entry Handler Plugin Class

The value of this attribute is the name and path to the class that implements the `com.sun.identity.liberty.ws.disco.plugins.DiscoEntryHandler` interface used to set or retrieve a principal's discovery entries. To handle this feature differently, you can implement the interface and set the implementing class as the value for this attribute. The default implementation for the Discovery Service is `com.sun.identity.liberty.ws.disco.plugins.UserDiscoEntryHandler`.

Classes for ResourceID Mapper Plugin

The value of this attribute is a list of classes that generate identifiers for a resource offering configured for an organization or role.
`com.sun.identity.liberty.ws.interfaces.ResourceIDMapper` is an interface used to map a user identifier to the resource identifier associated with it. The Discovery Service provides two implementations for this interface:

- `com.sun.identity.liberty.ws.disco.plugins.Default64ResourceIDMapper` (format: *providerID* + "/" + *the Base64 encoded userIDs*)
- `com.sun.identity.liberty.ws.disco.plugins.DefaultHexResourceIDMapper` (format: *providerID* + "/" + *the hex string of userID*)

Different implementations may be developed with the implementing class and added as a value of this attribute by clicking New and using the format `providerid=`*providerID*`|`*class_name_and_path*. See "To Configure a ResourceID Mapper" on page 219.

▼ To Configure a ResourceID Mapper

`com.sun.identity.liberty.ws.interfaces.ResourceIDMapper` is an interface used to map a user identifier to the resource identifier associated with it. Different implementations may be developed and added to the attribute. The following procedure is for adding a new resourceID mapper to the Discovery Service. The starting point is the Discovery Service screen under Web Services.

1. **In the Federation Manager Console, click the Web Services tab.**
2. **Under Web Services, select the Discovery Service tab.**
3. **Under Classes for ResourceID Mapper Plugin, click New or click on the name of a configured mapper to modify it.**

   The New Resource ID Mapping page is displayed.
4. **Provide values for the mapper attributes.**

   Provider ID
   : A URI that points to the Discovery Service. Use the format `http://`*host*`:`*port*`/amserver/Liberty/disco`.

   ID Mapper
   : The name of the implementing class.
5. **Click OK to complete the mapper configuration.**
6. **Click Save on the Discovery Service page to complete the configuration.**

Authenticate Response Message

If enabled, the service will authenticate the response message. By default, the function is not enabled.

Generate Session Context Statement for Bootstrapping

If enabled, this attribute specifies whether to generate a `SessionContextStatement` for bootstrapping. `SessionConxtext` in the `SessionContextStatement` is needed by the Discovery Service to support the `AuthenicateSessionContext` directive. By default, this option is not enabled.

Encrypt NameIdentifier in Session Context for Bootstrapping

If enabled, the service will encrypt the name identifier in a `SessionContextStatement`. By default, the option is not enabled.

Use Implied Resource; don't generate ResourceID for Bootstrapping

If enabled, the service will not generate a resource identifier for bootstrapping. By default, the option is not enabled.

Resource Offerings for Bootstrapping

This attribute defines a resource offering for bootstrapping a service. After single sign-on (SSO), this resource offering and its associated credentials will be sent to the client in the SSO assertion. Only one resource offering is allowed for bootstrapping. By default, this offering contains information regarding the Discovery Service. Tasks associated with this attribute include:

- "To Configure a Resource Offering for Bootstrapping" on page 221
- "To Configure a Service Description" on page 222

Note – The value of the Resource Offerings for Bootstrapping Resources attribute is a default value configured during installation. If you wish to define a new resource offering, you must first delete the existing resource offering. If you wish to modify the existing resource offering, click on the Edit link.

▼ To Configure a Resource Offering for Bootstrapping

Only one resource offering is allowed for bootstrapping. By default, this offering contains information regarding the Discovery Service. If a resource offering is already defined, you can modify the attributes by clicking the Edit link. You may also select the box next to the name of the Resource Offering to delete the existing resource offering. To configure a new resource offering, you would then click New.

1. **In the Federation Manager Console, click the Web Services tab.**
2. **Under Web Services, select the Discovery Service tab.**
3. **Under Resource Offerings for Bootstrapping, click New or click Edit to modify existing attributes.**

   The Resource Offering attributes are displayed.
4. **Provide or modify values for the resource offerings attributes.**

   Description
   : An optional description of the resource offering.

   Service Type
   : A URI that defines the type of service the resource offering implements. For example, `urn:liberty:disco:2003-08`.

   Note – It is recommended that this URI be the same as the `targetNamespace` URI of the abstract WSDL description for the service.

   Provider ID
   : A URI that points to the provider of the service instance. For example, `http://server.sun.com:80/amserver/Liberty/disco`.

   Security Mechanism ID
   : One or more URIs that identify the security mechanisms supported by the service instance defined in the previous attributes. These security mechanisms refer to the way a web service consumer authenticates to the web service provider. This attribute lists all of the security mechanisms that the service instance supports. The consumer picks the first mechanism (in the order listed) that it supports. They are listed in order of preference.

See "To Configure a Service Description" on page 222.

Options
: Check this box if the service has no options available for the resource offering. Options provide hints to a potential requester whether certain data or operations may be available with a particular resource offering. For example, an option may be provided stating that home contact information is available.

Option List
: This attribute contains a list of options for the service instance. The option is defined as a URI. The set of possible URIs are generally standardized by the service type.

Directives
: All supported directives (as described in "Supported Directives" on page 217) may contain a descriptive reference. If these Description ID References attributes are not defined for a directive, the directive is taken to apply to all authentication mechanisms provided in the resource offering. If a directive is enabled here, it MUST be defined with a list of Description ID References that refer to the authentication mechanism with which the directive is associated. The directive also MUST be taken to apply only to those descriptions referred to in the ID Refs list. This may be useful if certain directives are incompatible with certain security mechanisms. The supported directives for which Description ID References can be defined are:

  - GenerateBearerToken
  - AuthenticateRequestor
  - Encrypt ResourceID
  - AuthenticateSessionContext
  - AuthorizeRequester

5 Click OK to complete the mapper configuration.

6 Click Save on the Discovery Service page to complete the service configuration.

▼ To Configure a Service Description

The Service Description attribute defines a running web service at a distinct protocol endpoint. It is defined when you configure "Resource Offerings for Bootstrapping" on page 220. Information about service instances needs to be communicated in various contexts. For example, the Discovery Service defined is an identity service which provides an enumeration of resource offerings (each of which includes a service instance description).

1 In the Federation Manager Console, click the Web Services tab.

2 Under Web Services, select the Discovery Service tab.

3. **Under Resource Offerings for Bootstrapping, click New or click Edit to modify existing attributes.**

   The Resource Offering attributes are displayed.

4. **From the configuration screen of the Resource Offering for Bootstrapping attribute, click Add Mechanism ID to display the new security mechanism ID attributes or click Edit to modify an existing description.**

5. **Provide values for the attributes based on the following information:**

   Security Mechanism ID
   : This attribute is where authentication methods supported by the Discovery Service are added. These security mechanisms refer to the way a web service consumer authenticates to the web service provider or provides message-level security. By default, all available methods that the service instance supports are selected. If an authentication method is not selected, and a web services consumer sends a request using that method, the request is rejected. See "Supported Authentication Mechanisms" on page 217.

   End Point URL
   : Takes the URI for the SOAP-over-HTTP endpoint. For example, **`http://daiquiri.sun.com:80/amserver/Liberty/disco`**.

   SOAP Action
   : `SOAP Action` can be used to indicate the intent of the SOAP HTTP request. The SOAP processor on the receiving system can use this information to determine the ultimate destination for the service. The value is a URI. No defined value indicates no intent.

   Note – SOAP places no restrictions on the format or specificity of the URI or that it is resolvable.

6. **Click OK to complete the service configuration.**

7. **Click Save on the Discovery Service page to complete the service configuration.**

SOAP Binding Service

SOAP Binding Service is a transport layer for sending and receiving SOAP messages. In the SOAP Binding Service process, an identity service calls the client side application programming interface to construct a message and send it to the SOAP Receiver servlet (SOAP endpoint). The SOAP Receiver servlet receives the message, verifies the signature, and constructs a second message. The SOAP Receiver servlet then invokes the correct Request Handler to send this second message to the corresponding identity service for a response. The identity service processes the second message, generates a response, and sends that response back to the SOAP Receiver servlet. The servlet, in turn, sends the response back to the identity service for processing.

The following SOAP Binding Service global attributes can be configured for your implementation.

The following task is associated with configuring the SOAP Binding Service:

Request Handler List

The Request Handler List stores information about the classes implemented from the `com.sun.identity.liberty.ws.soapbinding.RequestHandler` interface. The SOAP Binding Service provides the interface to process requests and return responses. It must be implemented on the server side for each Liberty-based web service that uses the SOAP Binding Service.

Note – Currently, the Discovery Service, implemented Data Services Template (DST) services (including the Liberty Personal Profile Service and the sample Employee Profile Service, if deployed), and the Authentication Web Service use the SOAP Binding Service client API.

The Request Handler List displays entries that contain key/value pairs separated by a pipe as in `key=disco|class=com.example.identity.liberty.ws.disco.DiscoveryService`.

To modify the Request Handler list, see "To Configure a Request Handler" on page 224.

▼ To Configure a Request Handler

The Request Handler List stores information about the classes implemented from the `com.sun.identity.liberty.ws.soapbinding.RequestHandler` interface. `com.sun.identity.liberty.ws.soapbinding.RequestHandler` must be implemented on the server side by each Liberty-based identity service that accesses the SOAP Binding Service.

1. **In the Federation Manager Console, click the Web Services tab.**
2. **Under Web Services, select the SOAP Binding Service tab.**
3. **Under Request Handler List, click New or Edit to display the Request Handler List attributes.**
4. **Provide values for the attributes.**

Key
: The `Key` property is the last part of the URI path to a SOAP endpoint. The SOAP endpoint in Federation Manager is the SOAP Receiver servlet. The URI to the SOAP Receiver uses the format `protocol://`*host*`:`*port*`/`*deloy-uri*`/Liberty/`*key*. If you define `disco` as the `Key`, the URI path to the SOAP endpoint for the corresponding Discovery Service would be `protocol://`*host*`:`*port*`/amserver/Liberty/disco`. Different service clients use different keys when connecting to the SOAP Receiver.

Class
: The `Class` property specifies the name of the implemented Request Handler class for the particular web service. For example, `class=com.example.identity.liberty.ws.disco.DiscoveryService`.

SOAP Action
: `SOAP Action` can be used to indicate the intent of the SOAP HTTP request. The SOAP processor on the receiving system can use this information to determine the ultimate destination for the service. The value is a URI. No defined value indicates no intent.

Note – SOAP places no restrictions on the format or specificity of the URI or that it is resolvable.

---

5 **Click OK to complete the Request Handler configuration.**

6 **Click Save on the SOAP Binding Service page to complete the service configuration.**

Web Service Authenticator

This attribute takes as a value the implementation class for the Web Service Authenticator interface. This class authenticates a request and generates a credential for a web service consumer.

Supported Authentication Mechanisms

This attribute specifies the authentication mechanisms supported by the SOAP Receiver. Authentication mechanisms offer user authentication, as well as data integrity and encryption. By default, all available authentication mechanisms are selected. If one is not selected, and a web services consumer sends a request using it, the request is rejected. Following is a list of the supported authentication mechanisms:

- `urn:liberty:security:2003-08:null:null`
- `urn:liberty:security:2003-08:null:X509`
- `urn:liberty:security:2003-08:null:SAML`
- `urn:liberty:security:2004-04:null:Bearer`

- urn:liberty:security:2003-08:TLS:null
- urn:liberty:security:2003-08:TLS:X509
- urn:liberty:security:2003-08:TLS:SAML
- urn:liberty:security:2004-04:TLS:Bearer
- urn:liberty:security:2003-08:ClientTLS:null
- urn:liberty:security:2003-08:ClientTLS:X509
- urn:liberty:security:2003-08:ClientTLS:SAML
- urn:liberty:security:2004-04:ClientTLS:Bearer

Authentication Web Service (Authentication Service)

The Authentication Web Service defines how to perform authentication using SOAP. The exchange of authentication information between a web service consumer (WSC) and the web service provider (WSP) is accomplished using SOAP-bound messages. The messages are a series of client requests and server responses specific to the defined Simple Authentication and Security Layer (SASL) mechanism (or mode of authentication). After receiving a request for authentication (or any request from the WSC), the WSP may issue additional challenges, or indicate authentication failure or success. The Authentication Web Service is for service-to-service (non-user) authentication. The following steps describe the sequence between the WSC and the Authentication Web Service (a WSP).

1. The authentication exchange begins with a WSC sending an SASL authentication request to the Authentication Web Service on behalf of a principal.

   The request message contains an identifier for the principal and indicates one or more SASL mechanisms from which the service can choose.

2. The Authentication Web Service responds by asserting the method to use and, if applicable, initiating a challenge.

   If the Authentication Web Service does not support any of the cited methods, it responds by aborting the exchange.

3. The WSC responds with the necessary credentials for the chosen method of authentication.
4. The Authentication Web Service replies by approving or disproving the authentication.

   If approved, the response includes the credentials the WSC needs to invoke other web services (like the Discovery Service).

The following Authentication Web Service global attribute can be configured for your implementation.

- "Mechanism Handlers List" on page 227

The following task is associated with configuring the Authentication Web Service:

- "To Configure a Mechanism Handler" on page 227

Mechanism Handlers List

The Mechanism Handler List attribute stores information about the SASL mechanisms supported by the Authentication Web Service. To add or modify these values, see "To Configure a Mechanism Handler" on page 227.

▼ To Configure a Mechanism Handler

1. **In the Federation Manager Console, click the Web Services tab.**
2. **Under Web Services, select the Authentication Service tab.**
3. **Under Mechanism Handlers List, click Add or Edit to display the Mechanism Handler attributes.**
4. **Provide values for the attributes.**

   Key
   Defines the SASL mechanism supported by the Authentication Web Service.

   Class
   The Authentication Web Service provides a handler interface that needs to be implemented in order for each SASL mechanism to process the requested message and return a response. The `class` parameter specifies the name of the implementation class for the SASL mechanism. Two authentication mechanisms are supported out-of-the-box by the following classes:

   `com.sun.identity.liberty.ws.authnsvc.mechanism.PlainMechanismHandler`
   This class is the default implementation for the PLAIN authentication mechanism. It maps user identifiers and passwords in the PLAIN mechanism to the user identifiers and passwords in the LDAP authentication module under the root organization.

   `com.sun.identity.liberty.ws.authnsvc.mechanism.CramMD5MechanismHandler`
   This class is the default implementation for the CRAM-MD5 authentication mechanism.

5. **Click OK or Save to complete the Mechanism Handler configuration.**
6. **Click Save on the Authentication Web Service page to complete the service configuration.**

Creating a New Web Service

Federation Manager contains a sample that illustrates how to develop a simple web service. The ample web service sends stock data based on a defined user type. It shows how to extract authentication information from an authenticated SAML Assertion (SAML Bearer Token). The sample instructions can be modified to develop your own web service. The sample is located in the */FederationManager-base*`/SUNWam/fm/samples/liberty/webservices/stockticker` directory.

Index

D

E

F

R

S

T

U

W