Sun™ Federated Access Manager 8.0 Administration Guide

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

Part No: 820–3885

080612@20490

Contents

Preface

The Sun^{TM} Federated Access Manager 8.0 Administration Guide describes how to use the Sun Federated Access Manager console as well as manage user and service data via the command line interface.

Federated Access Manager is a component of the Sun Java Enterprise System (Java ES), a set of software components that provide services needed to support enterprise applications distributed across a network or Internet environment.

Who Should Use This Book

This book is intended for use by IT administrators and software developers who implement a web access platform using Sun Java System servers and software.

Before You Read This Book

Readers should be familiar with the following components and concepts:

- Access Manager technical concepts as described in the *Sun Java System Access Manager 7.1 Technical Overview*
- Deployment platform: Solaris^{TM} or Linux operating system
- Web container that will run Access Manager: Sun Java System Application Server, Sun Java System Web Server, BEA WebLogic, or IBM WebSphere Application Server
- Technical concepts: Lightweight Directory Access Protocol (LDAP), Java^{TM} technology, JavaServer Pages^{TM} (JSP) technology, HyperText Transfer Protocol (HTTP), HyperText Markup Language (HTML), and eXtensible Markup Language (XML)

Related Books

Related documentation is available as follows:

- "Access Manager Core Documentation" on page 12
- "Sun Java Enterprise System Product Documentation" on page 13

Access Manager Core Documentation

The Access Manager core documentation set contains the following titles:

- The *Sun Java System Access Manager 7.1 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 Access Manager 7.1 Technical Overview* provides an overview of how Access Manager components work together to consolidate access control functions, and to protect enterprise assets and web-based applications. It also explains basic Access Manager concepts and terminology.
- The *Sun Java System Access Manager 7.1 Deployment Planning Guide* provides planning and deployment solutions for Sun Federated Access Manager based on the solution life cycle
- The *Sun Java System Access Manager 7.1 Postinstallation Guide* provides information on configuring Access Manager after installation.
- The *Sun Java System Access Manager 7.1 Performance Tuning Guide* provides information on how to tune Access Manager and its related components for optimal performance.
- The *Sun Java System Access Manager 7.1 Administration Guide* describes how to use the Access Manager console as well as manage user and service data via the command line interface.
- The *Sun Java System Access Manager 7.1 Federation and SAML Administration Guide* provides information about the Federation module based on the Liberty Alliance Project specifications. It includes information on the integrated services based on these specifications, instructions for enabling a Liberty-based environment, and summaries of the application programming interface (API) for extending the framework.
- The *Sun Java System Access Manager 7.1 Developer's Guide* offers information on how to customize Access Manager and integrate its functionality into an organization's current technical infrastructure. It also contains details about the programmatic aspects of the product and its API.
- The *Sun Java System Access Manager 7.1 C API Reference* provides summaries of data types, structures, and functions that make up the public Access Manager C APIs.
- The *Java API Reference*provides information about the implementation of Java packages in Access Manager.

- The *Sun Java System Access Manager Policy Agent 2.2 User's Guide* provides an overview of the policy functionality and the policy agents available for Access Manager.

Updates to the *Release Notes* and links to modifications of the core documentation can be found on the Access Manager page at the Sun Java Enterprise System documentation web site. Updated documents will be marked with a revision date.

Sun Java Enterprise System Product Documentation

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

- Directory Server
- Web Server
- Application Server
- Web Proxy Server

Related Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – 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 or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (`http://www.sun.com/documentation/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://www.sun.com/training/`)

Typographic Conventions

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

TABLE P–1 Typographic Conventions

Typeface	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	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

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

TABLE P–2 Shell Prompts

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

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions.

To share your comments, go to `http://docs.sun.com` and click Send Comments. In the online form, provide 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 Federated Access Manager 8.0 Administration Guide*, and the part number is 820–3885.

PART I

Access Control

Note – Please be advised that this book has been published for the Federated Access Manager 8.0 Early Access release. The information contained in this book may not reflect the most current release of the software.

This is part one of the Sun™ Federated Access Manager 8.0 Administration Guide. The Access Control interface provides a way to create and manage authentication and authorization services to protect and regulate realm-based resources. When an enterprise user requests information, Federated Access Manager verifies the user's identity and authorizes the user to access the specific resource that the user has requested. The part contains the following chapters:

CHAPTER 1

The Federated Access Manager Console

The Federated Access Manager console is a web interface that allows administrators with different levels of access to, among other things, create realms and organizations, create or delete users to and from those realms and establish enforcement policies that protect and limit access to realms' resources. In addition, administrators can view and terminate current user sessions and manage their federation configurations (create, delete and modify authentication domains and providers). Users without administrative privileges, on the other hand, can manage personal information (name, e-mail address, telephone number, and so forth), change their password, subscribe and unsubscribe to groups, and view their roles. The Federated Access Manager Console has two, basic views:

- “Administration View” on page 19
- “User Profile View” on page 22

Administration View

When a user with an administrative role authenticates to Federated Access Manager, the default view is the Administration view. In this view, the administrator can perform most administrative tasks related to Federated Access Manager. Federated Access Manager can be installed in two different modes; Realms mode and Legacy Mode. Each mode has its own console. For more information on Realm and Legacy Modes, see the *Sun Java System Access Manager 7.1 Technical Overview*.

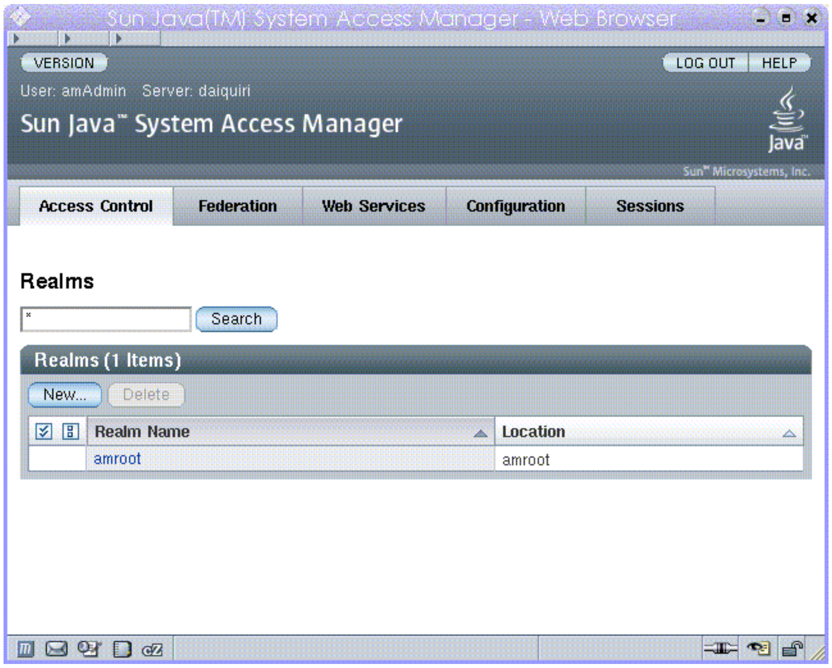
Note – If you install Federated Access Manager 8.0 in Realm Mode, you cannot revert to Legacy Mode. If you install Federated Access Manager in Legacy Mode, you can change to Realm Mode by using the amadmin command. See Changing from Legacy Mode to Realm Mode in the *Access Manager Administration Reference* for more information.

Realms Mode Console

The Administration console in realms mode enables administrators to manage realm-based access control, default service configuration, Web services and Federation. To access the administrator login screen, use the following address syntax in your browser:

protocol`://`*servername*`/amserver/UI/Login`

`protocol` is either http: or https, depending upon your deployment.



Realms (1 Items)

Realm Name	Location
amroot	amroot

FIGURE 1–1 Realms Mode Administration View

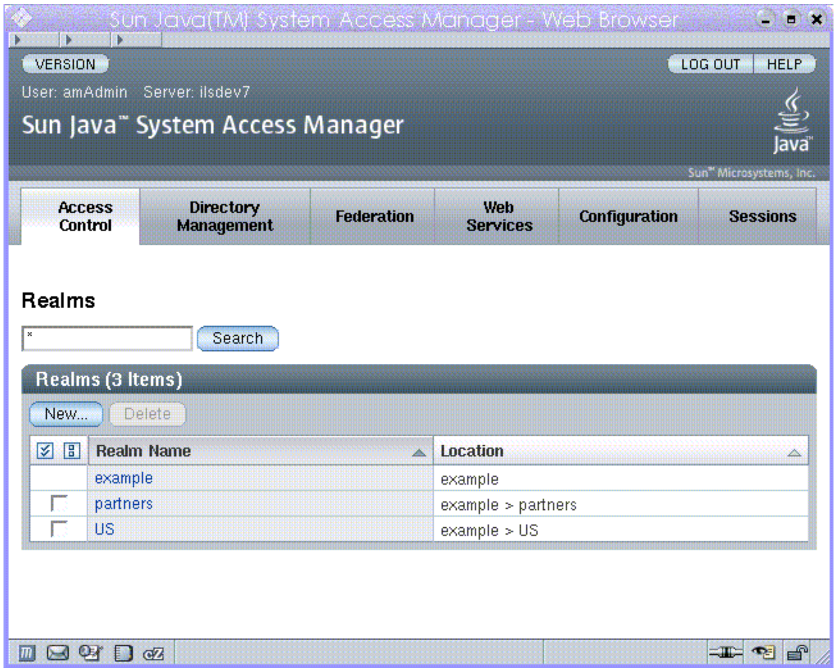
Legacy Mode Console

Legacy Mode console is based on the Access Manager 6.3 architecture. This legacy Access Manager architecture uses the LDAP directory information tree (DIT) that comes with Sun Directory Server. In Legacy Mode, both user information and access control information are stored in LDAP organizations. When you choose Legacy Mode, an LDAP organization is the equivalent of an access control realm. Realm information is integrated within LDAP organizations. In Legacy Mode, the Directory Management tab is available for Access Manager-based identity management.

To access the administrator login screen, use the following address syntax in your browser:

protocol`://`*servername*`/amserver/console`

`protocol` is either http: or https:, depending upon your deployment.



Realm Name	Location
example	example
partners	example > partners
US	example > US

FIGURE 1–2 Legacy Mode Administration View

Legacy Mode 6.3 Console

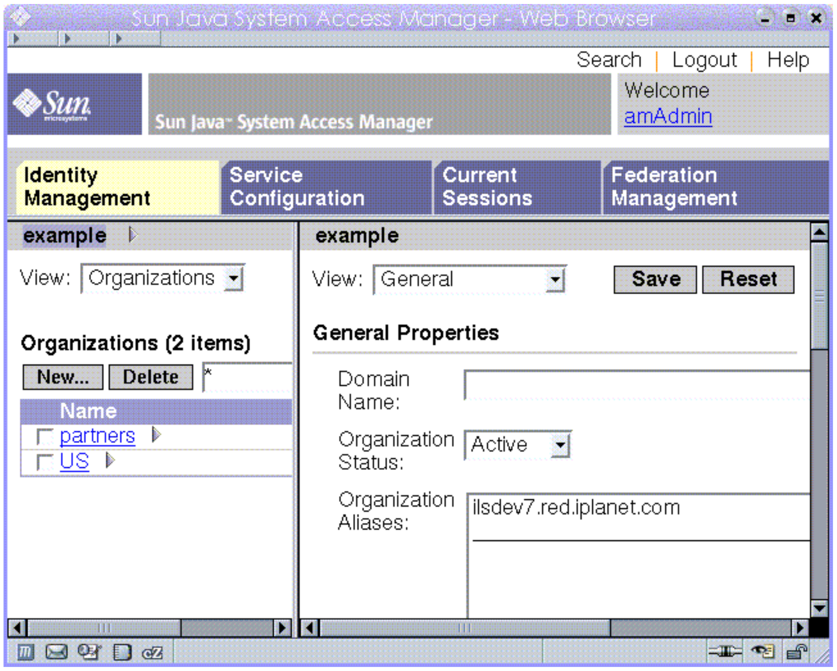
Some features of Access Manager 6.3 are not available in the Federated Access Manager 8.0 console. Because of this, administrators can log into the 6.3 console through a 8.0 Legacy deployment. This console is typically used where Federated Access Manager is built upon Sun Java System Portal Server or other Sun Java System communication products that require the use of Sun Directory Server as the central identity repository. Other features, such Delegated Administration and Class of Service, are accessed only through this console.

Note – Do not interchange between using the 6.3 and 8.0 Legacy mode consoles.

To access the 6.3 console, use the following address syntax in your browser:

protocol`://`*servername*`/amconsole`

protocol is either http: or https:, depending upon your deployment.



View: Organizations

View: General

Domain Name:

Organization Status: Active

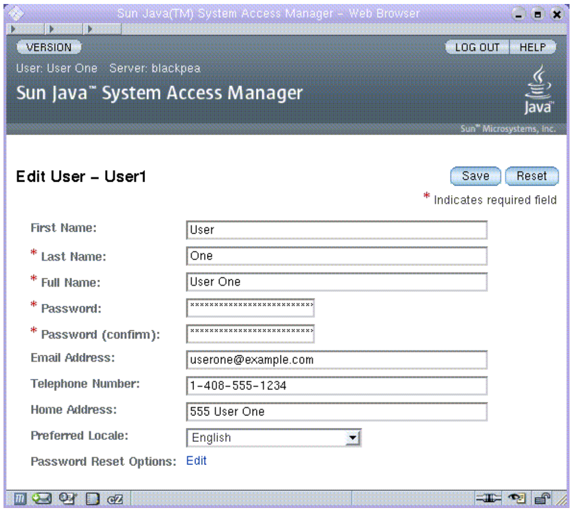
Organization Aliases: ilsdev7.red.iplanet.com

FIGURE 1–3 Legacy 6.3–based Console

User Profile View

When a user who has not been assigned an administrative role authenticates to the Federated Access Manager the default view is the user's own User Profile. The User Profile view can be accessed in either Realm or Legacy Mode. The user must enter the user's own username and password at the Login page in order to access this view.

In this view the user can modify the values of the attributes particular to the user's personal profile. This can include, but is not limited to, name, home address and password. The attributes displayed in the User Profile View can be extended.


Sun Java(TM) System Access Manager - Web Browser
VERSION
LOG OUT
HELP
User: User One Server: blackpea
Sun Java™ System Access Manager
Java
Sun™ Microsystems, Inc.

Edit User - User1

Save Reset

* Indicates required field

First Name:	User
* Last Name:	One
* Full Name:	User One
* Password:	**************************
* Password (confirm):	**************************
Email Address:	userone@example.com
Telephone Number:	1-408-555-1234
Home Address:	555 User One
Preferred Locale:	English
Password Reset Options:	Edit

FIGURE 1–4 User Profile View

CHAPTER 2

Managing Realms

An access control realm is a group of authentication properties and authorization policies you can associate with a user or group of users. Realm data is stored in a proprietary information tree that Federated Access Manager creates within a data store you specify. The Federated Access Manager framework aggregates policies and properties contained in each realm within the Federated Access Manager information tree. By default, Federated Access Manager automatically inserts the Federated Access Manager information tree as a special branch in Sun Java Enterprise System Directory Server, apart from the user data. You can use access control realms while using any LDAPv3 database.

For more information on realms, see the *Sun Java System Access Manager 7.1 Technical Overview*.

In the Realms tab, you can configure the following properties for access control:

- "Authentication" on page 27
- "Services" on page 27
- "Privileges" on page 28

Creating and Managing Realms

This section describes how to create and manage realms.

▼ To Create a New Realm

1 **Select New from the Realms list under the Access Control tab.**

2 **Define the following general attributes:**

Name	Enter a name for the Realm.

Parent	Defines the location of the realm that you are creating. Select the parent realm under which the new realm will exist.

3 **Define the following realm attributes:**

Realm Status	Choose a status of active or inactive. The default is active. This can be changed at any time during the life of the realm by selecting the Properties icon. Choosing inactive disables user access when logging in.
Realm/DNS Aliases	Allows you to add alias names for the DNS name for the realm. This attribute only accepts "real" domain aliases (random strings are not allowed).

4 **Click OK to save or Cancel to return to the previous page.**

General Properties

The General Properties page displays the basic attributes for a realm. To modify these properties, click the realm from the Realm Names list under the Access Control tab. Then, edit the following properties:

Realm Status	Choose a status of active or inactive. The default is active. This can be changed at any time during the life of the realm by selecting the Properties icon. Choosing inactive disables user access when logging in.
Realm/DNS Aliases	Allows you to add alias names for the DNS name for the realm. This attribute only accepts "real" domain aliases (random strings are not allowed).

Once you edit the properties, click Save.

Note – The `recursive=true` flag in the `AMAdmin.dtd` does not work for searching for objects in sub-realms in realm mode. This flag only works in legacy mode because all sub-organizations are located under the same root suffix. In realm mode, each sub-realm can have a different root suffix and may even be located on a different server. If searching for objects, such as groups, in a sub-realm, you must specify the sub-realm in which you are searching in the XML data file.

Authentication

The general authentication service must be registered as a service to a realm before any user can log in using the other authentication modules. The core authentication service allows the Federated Access Manager administrator to define default values for a realm's authentication parameters. These values can then be used if no overriding value is defined in the specified authentication module. The default values for the Core Authentication Service are defined in the `amAuth.xml` file and stored in Directory Server after installation.

For more information, see Managing Authentication

Services

In Federated Access Manager, a service is a group of attributes that are managed together by the Federated Access Manager console. The attributes can be just bits of related information such as an employee's name, job title, and email address. But attributes are typically used as configuration parameters for a software module such as a mail application or payroll service.

Through the Services tab, you can add and configure a number of Federated Access Manager default services to a realm. You can add the following services:

- Administration
- Discovery Service
- Globalization Settings
- Password Reset
- Session
- User

Note – Federated Access Manager enforces that required attributes in service .xml files have some default values. If you have services with required attributes with no values, you need to add default values and reload the service.

▼ To Add a Service to a Realm

1 **Click the name of the realm for which you wish to add a new service.**

2 **Select the Services tab.**

3 **Click Add in the Services list.**

4 **Select the service you wish to add for the realm.**

5 **Click Next.**

6 **Configure the service by defining the realm attributes. See Configuration in the online help for a description of the service attributes.**

7 **Click Finish.**

8 **To edit the properties of a service, click the name in the Service list.**

Privileges

The delegation model in Federated Access Manager is based on privileges (or entitlements) that have been assigned to the administrators. A privilege is an operation (or action) that can be performed on a resource; for example, a READ operation on Policy objects. The set of operations that are defined are READ, MODIFY and DELEGATE. The resources are objects on which the actions can be performed, and can be either a configuration object or an identity object.

Examples of configuration objects are Authentication Configuration, Policies, Data Stores, and so forth. Examples of identity objects are Users, Groups, Roles, and Agents. A set of privileges can be dynamically created and added to Federated Access Manager dynamically, however during installation, a small set of privileges are added to get Federated Access Manager to properly run. Once the privileges are loaded, the privileges can be assigned to roles and groups. Users belonging to these roles and groups would be the delegated administrators and would be able to perform the assigned operations. Basically, administrators are users who are members of roles and groups to which a set of one or more privileges are assigned.

Federated Access Manager 8.0 allows you to configure permissions for the following administrator types:

- Realm administrators — Realm administrators have all the permissions for READ, MODIFY and DELEGATE operations for all objects (both configuration and identity objects). Realm administrators can be considered as “root” within a Unix system. Realm administrators can create sub-realms, modify configurations for all the services and also create, modify and delete Users, Groups, Roles and Agents.
- Policy administrators — Policy administrators have permissions to manage policies and policy service configurations only. They can create, modify and delete policies which consists of Rules, Subjects, Conditions and Response Attributes. However in order to manage policies, these administrators need read permissions for Identity Repository Subjects and also Authentication configuration. These administrators are able to view the identities and authentication configurations.

- Log administrators — Log administrators have permissions to read and/or write log records which can be used to protect the audit logs from being maliciously abused by rouge applications. Since logging interfaces are public, it is possible that any authenticated user can read and write logs records, and this privilege is added to prevent such abuse. The main users of logging interfaces are J2EE and Web Agents and these require only MODIFY privilege, and should not have READ privilege. Similarly, administrators who view the logs should have only READ privilege, and should not have MODIFY privilege. In order to cater for the these types of usages, the logging privileges are further sub-divided as follows:
  - Log administrators with Write Access – These administrators have permissions to write all log files.
  - Log administrators with Read Access – These administrators have permissions to read all log files.
  - Log administrators with Read and Write Access – These administrators have permissions to read and write to all log files.

Defining Privileges for Federated Access Manager

A new installation instance of Federated Access Manager 8.0 provides access permissions for policy administrators, realm administrators (or organization administrators in Legacy mode) and Log Administrators. To assign or modify privileges, click the name of the role or group you wish to edit. You can select from the following:

Read and write access to all log files
  Defines both read and write access privileges to log administrators.

Write access to all log files
  Defines only write access privileges to log administrators.

Read access to all log files
  Defines only read access privileges to log administrators.

Read and write access only for policy properties
  Defines read and write access privileges for policy administrators.

Read and write access to all realm and policy properties
  Defines read and write access privileges for realm administrators.

Defining Privileges for an Access Manager 7.0 to 7.1 Upgrade

If you have upgraded Access Manager from version 7.0 to 7.1, the privilege configuration differs from that of a new Access Manager 7.1 installation, however privileges for policy administrators, realm administrators and log administrators are still supported. To assign or modify privileges, click the name of the role or group you wish to edit. You can select from the following:

Read only access to data stores
: Defines read access privileges to datastores for policy administrators.

Read and write access to all log files
: Defines both read and write access privileges for log administrators.

Write access to all log files
: Defines only write access privileges for log administrators.

Read access to all log files
: Defines only read access privileges for log administrators.

Read and write access only for policy properties
: Defines read and write access privileges for policy administrators.

Read and write access to all realm and policy properties
: Defines read and write access privileges for realm administrators.

Read only access to all properties and services
: Defines read access privileges to all properties and services for policy administrators.

Access Manager does not support the following definitions used either separately or together:

- Read only access to data stores
- Read only access to all properties and services

These privilege definitions must be used with the "Read and write access only for policy properties" definition to define delegation control for policy administrators.

CHAPTER 3

Data Stores

A data store is a database where you can store user attributes and user configuration data. Federated Access Manager provides identity repository plug-ins that connect to an LDAPv3 identity repository framework. These plug-ins enable you to view and retrieve Federated Access Manager user information without having to make changes in your existing user database. The Federated Access Manager framework integrates data from the identity repository plug-in with data from other Federated Access Manager plug-ins to form a virtual identity for each user. Federated Access Manager can then use the universal identity in authentication and authorization processes among more than one identity repository. The virtual user identity is destroyed when the user's session ends.

Federated Access Manager Data Store Types

This section explains the types of data stores that you can configure, and also provides the steps to create new data store types and how to configure them.

You can create a new, data store instance for any of the following data store types:

Federated Access Manager Repository Plug-in

This data store type resides in a Sun Directory Server instance and holds the Federated Access Manager information tree. This data store type makes use of Directory Server features that are not part of the LDAP version 3 specification, such as roles and class of service, and is compatible with previous versions of Federated Access Manager.

Active Directory

This data store type uses the LDAP version 3 specification to write identity data to an instance of Microsoft Active Directory.

Flat Files Repository

This repository allows you to store data and identities in a flat DIT structure on the local installation instance of Federated Access Manager without having to create a separate data store. This is generally used for testing or proof of concept deployments.

Generic LDAPv3

This data store type allows identity data to written to any LDAPv3–compliant database. If the LDAPv3 database you are using does not support Persistent Search, then you can not use the caching feature.

Sun Directory Server With Federated Access Manager Schema

This data store type resides in a Sun Directory Server instance and holds the Federated Access Manager information tree. It differs from the Federated Access Manager Repository Plug-in, in that more configuration attributes allow you to better customize the data store.

▼ To Create a New Data Store

The following section describes the steps to connect a data store.

1. **Select the realm to which you wish to add a new data store.**
2. **Click the Data Store tab.**
3. **Click New from the Data Stores list.**
4. **Enter a name for the data store.**
5. **Select the type of data store you wish to create.**
6. **Click Next.**
7. **Configure the data store by entering the appropriate attribute values.**
8. **Click Finish.**

Data Store Attributes

This section defines the attributes for configuring each new Federated Access Manager data store. The data store attributes are:

- "Federated Access Manager Repository Attributes" on page 33
- "Flat Files Repository Attributes" on page 35
- "LDAPv3 Attributes" on page 36

Note – The Active Directory, Generic LDAPv3, and Sun Directory Server with Federated Access Manager Schema data store types share the same underlying plug—in, so the configuration attributes are the same. However, the default values for some of the attributes are different for each datastore type and are displayed accordingly in the Federated Access Manager console.

Federated Access Manager Repository Attributes

The following attributes are used to configure an Federated Access Manager Repository plug-in:

Class Name

Specifies the location of the class file which implements the Federated Access Manager repository plug-in.

Federated Access Manager Supported Types and Operations

Specifies the operations that are permitted to or can be performed on this LDAP server. The default operations that are the only operations that are supported by this LDAPv3 repository plug-in. The following are operations supported by LDAPv3 Repository Plugin:

- group — read, create, edit, delete
- user — read, create, edit, delete, service
- agent — read, create, edit, delete

You can remove permissions from the above list based on your LDAP server settings and the tasks, but you can not add more permissions.

If the configured LDAPv3 Repository plug—in is pointing to an instance of Sun Directory Server, then permissions for the type `role` can be added. Otherwise, this permission may not be added because other data stores may not support roles. The permission for the type 'role' is:

- role — read, create, edit, delete

If you have `user` as a supported type for the LDAPv3 repository, the read, create, edit, and delete service operations are possible for that user. In other words, if `user` is a supported type, then the read, edit, create, edit, and delete operations allow you to read, edit, create, and delete user entries from the identity repository. The `user=service` operation lets Federated Access Manager services access attributes in user entries. Additionally, the user is allowed to access the dynamic service attributes if the service is assigned to the realm or role to which the user belongs.

The user is also allowed to manage the user attributes for any assigned service. If the user has `service` as the operation (`user=service`), then it specifies that all service-related operations are supported. These operations are`assignService`, `unassignService`, `getAssignedServices`, `getServiceAttributes`, `removeServiceAttributes` and `modifyService`.

Organization DN Value

Defines the DN that points to an organization in the Directory Server to be managed by Federated Access Manager. This will be the base DN of all operations performed in the data store.

People Container Naming Attribute

Specifies the naming attribute of the people container if a user resides in a people container. This field is left blank if the user does not reside in a people container.

People Container Value

Specifies the value of the people container. The default is `people`.

Agent Container Naming Attribute

The naming attribute of agent container if the agent resides in a agent container. This field is left blank if the agent does not reside in agent container.

Agent Container Value

Specifies the value of the agent container. The default is `agents`.

Recursive Search

If enabled, the search performed in the Federated Access Manager repository will conduct a recursive search for the specified identities. For example, a recursive search performed on the following data structure:

```
root
realm1
  subrealm11
    user5
  subrealm12
    user6
```

```
realm2
   user1
   user2
   subrealm21
      user3
      user4
```

will produce the following results:

- If a search is performed from the root and no users are defined at this level (aside from `amadmin` and `anonymous`), the search will return users 1–6.
- If a search is performed from `realm1` and no users are defined, the search will return `user5` and `user6`.
- If a search is performed from `realm2` (two users defined), the search will return users 1–4.

Copy Realm Configuration

When this attribute is enabled in a realm-mode installation, Federated Access Manager will create an equivalent organization and sub-organization for each realm and sub-realm that exists in the repository. In addition, the services that are registered to the realm/sub-realm will be registered to the new created organization/sub-organization. Both the realm DIT and the organization DIT exist within the datastore.

Flat Files Repository Attributes

The following attributes are used to configure a flat file repository:

Files Repository Plug-in Classname

This attribute specifies the Java class file that provides the implementation for flat files. This attribute should not be modified.

Files Repository Directory

Defines the base directory where the identities and their attributes are stored.

Cache

When enabled (default), the identities and their attributes will be cached. Subsequent requests will not access the file system.

Time to Update Cache

When caching is enabled, this attribute determines the time interval (in minutes) after which the entries in the cache are checked for any changes made to the file system. The checking mechanism is based on timestamps.

File User Object Classes

Defines the object classes that are automatically added to the users when they are created.

Password Attribute

Provides the attribute name that contains the password used for authentication. This attribute is used to authenticate the user when the Data Store authentication module is enabled.

Status Attribute

Provides the attribute name that stores the identity's status. Values for the status attribute are either `active` or `inactive`. This is used during the authentication of the identity. If an identity is `inactive`, the use will not be authenticated.

Hashed Attributes

Provides a list of attributes whose values will be hashed and stored in the files. Once hashed, the original values cannot be obtained. Only hashed values are retrieved. This is used to ensure privacy where certain attributes should not be permanently stored, but are used for verification. An identity's password attribute, is an example of this type of attribute.

Encrypted Attributes

Provides a list of attributes whose values will be encrypted and stored in the files. Although they are encrypted and stored, calling the Identity Repository APIs would return the original decrypted values. This is prevent users from accessing the file system directly and reading sensitive attributes.

LDAPv3 Attributes

The following attributes are used to configure a LDAPv3 repository plug-in:

LDAP Server

Enter the name of the LDAP server to which you will be connection. The format should be `hostname.domainname:portnumber`.

If more than one `host:portnumber` entries are entered, an attempt is made to connect to the first host in the list. The next entry in the list is tried only if the attempt to connect to the current host fails.

LDAP Bind DN

Specifies the DN name that Federated Access Manager will use to authenticate to the LDAP server to which you are currently connected. The user with the DN name used to bind should have the correct add/modification/delete privileges that you configured in the"LDAPv3 Plugin Supported Types and Operations" on page 38 attribute.

LDAP Bind Password

Specifies the DN password that Federated Access Manager will use to authenticate to the LDAP server to which you are currently connected

LDAP Bind Password (confirm)

Confirm the password.

LDAP Organization DN

The DN to which this data store repository will map. This will be the base DN of all operations performed in this data store.

LDAP SSL

When enabled, Federated Access Manager will connect to the primary server using the HTTPS protocol.

LDAP Connection Pool Minimum Size

Specifies the initial number of connections in the connection pool. The use of connection pool avoids having to create a new connection each time.

LDAP Connection Pool Maximum Size

Specifies the maximum number of connections to allowed.

Maximum Results Returned from Search

Specifies the maximum number of entries returned from a search operation. If this limit is reached, Directory Server returns any entries that match the search request.

Search Timeout

Specifies the maximum number of seconds allocated for a search request. If this limit is reached, Directory Server returns any search entries that match the search request.

LDAP Follows Referral

If enabled, this option specifies that referrals to other LDAP servers are followed automatically.

LDAPv3 Repository Plugin Class Name

Specifies the location of the class file which implements the LDAPv3 repository.

General Attribute Name Mapping

Enables common attributes known to the framework to be mapped to the native data store. For example, if the framework uses `inetUserStatus` to determine user status, it is possible that the native data store actually uses `userStatus`. The attribute definitions are case-sensitive.

LDAPv3 Plugin Supported Types and Operations

Specifies the operations that are permitted to or can be performed on this LDAP server. The default operations that are the only operations that are supported by this LDAPv3 repository plug-in. The following are operations supported by LDAPv3 Repository Plugin:

- group — read, create, edit, delete
- user — read, create, edit, delete, service
- agent — read, create, edit, delete

You can remove permissions from the above list based on your LDAP server settings and the tasks, but you can not add more permissions.

If the configured LDAPv3 Repository plug—in is pointing to an instance of Sun Directory Server, then permissions for the type `role` can be added. Otherwise, this permission may not be added because other data stores may not support roles. The permission for the type 'role' is:

- role — read, create, edit, delete

If you have `user` as a supported type for the LDAPv3 repository, the read, create, edit, and delete service operations are possible for that user. In other words, if `user` is a supported type, then the read, edit, create, edit, and delete operations allow you to read, edit, create, and delete user entries from the identity repository. The `user=service` operation lets Federated Access Manager services access attributes in user entries. Additionally, the user is allowed to access the dynamic service attributes if the service is assigned to the realm or role to which the user belongs.

The user is also allowed to manage the user attributes for any assigned service. If the user has `service` as the operation (`user=service`), then it specifies that all service-related operations are supported. These operations are`assignService`, `unassignService`, `getAssignedServices`, `getServiceAttributes`, `removeServiceAttributes` and `modifyService`.

LDAPv3 Plug-in Search Scope

Defines the scope to be used to find LDAPv3 plug-in entries. The scope must be one of the following:

- SCOPE_BASE

- SCOPE_ONE
- SCOPE_SUB (default)

LDAP Users Search Attribute

This field defines the attribute type for which to conduct a search on a user. For example, if the user's DN is `uid=user1,ou=people,dc=iplanet,dc=com` , then the naming attribute is `uid`.

LDAP Users Search Filter

Specifies the search filter to be used to find user entries.

LDAP User Object Class

Specifies the object classes for a user. When a user is created, this list of user object classes will be added to the user's attributes list.

LDAP User Attributes

Defines the list of attributes associated with a user. Any attempt to read/write user attributes that are not on this list is not allowed. The attributes are case-sensitive. The object classes and attribute schema must be defined in Directory Server before you define the object classes and attribute schema here.

LDAP User Creation Attribute Mappings

Specifies which attributes are required when a user is created. This attribute uses the following syntax:

```
DestinationAttributeName=SourceAttributeName
```

If the source attribute name is missing, the default is the user ID (`uid`). For example:

cn
sn=givenName

Both `cn` and `sn` are required in order to create a user profile. `cn` gets the value of the attribute named `uid`, and `sn` gets the value of the attribute named `givenName.`

User Status Attribute

Specifies the attribute name to indicate the user's status.

User Status Active Value

Specifies the attribute name for an active user status. The default is `active`.

User Status Inactive Value

Specifies the attribute name for an inactive user status. The default is `inactive`.

LDAP Groups Search Attribute

This field defines the attribute type for which to conduct a search on a group. The default is `cn`.

LDAP Group Search Filter

Specifies the search filter to be used to find group entries. The default is `(objectclass=groupOfUniqueNames)`.

LDAP Groups Container Naming Attribute

Specifies the naming attribute for a group container, if groups resides in a container. Otherwise, this attribute is left empty. For example, if a group DN of `cn=group1,ou=groups,dc=iplanet,dc=com`resides in `ou=groups`, then the group container naming attribute is `ou`.

LDAP Groups Container Value

Specifies the value for the group container. For example, a group DN of `cn=group1,ou=groups,dc=iplanet,dc=com` resides in a container name `ou=groups`, then the group container value would be `groups`.

LDAP Groups Object Classes

Specifies the object classes for groups. When a group is created, this list of group object classes will be added to the group's attributes list.

LDAP Groups Attributes

Defines the list of attributes associated with a group. Any attempt to read/write group attributes that are not on this list is not allowed. The attributes are case-sensitive. The object classes and attribute schema must be defined in Directory Server before you define the object classes and attribute schema here.

Group Membership Attribute

Specifies the name of the attribute whose values are the names of all the groups to which DN belongs. The default is `memberOf`.

Unique Member Attribute

Specifies the attribute name whose values is a DN belonging to this group. The default is `uniqueMember`.

Group Member URL Attribute

Specifies the name of the attribute whose value is an LDAP URL which resolves to members belonging to this group. The default is `memberUrl`.

LDAP People Container Naming Attribute

Specifies the naming attribute of the people container if a user resides in a people container. This field is left blank if the user does not reside in a people container.

LDAP People Container Value

Specifies the value of the people container. The default is `people`.

LDAP Agents Search Attribute

This field defines the attribute type for which to conduct a search on an agent. The default is `uid`.

LDAP Agents Container Naming Attribute

The naming attribute of agent container if the agent resides in a agent container. This field is left blank if the agent does not reside in agent container.

LDAP Agents Container Value

Specifies the value of the agent container. The default is `agents`.

LDAP Agents Search Filter

Defines the filter used to search for an agent. The LDAP Agent Search attribute is prepended to this field to form the actual agent search filter.

For example, if the LDAP Agents Search Attribute is `uid` and LDAP Users Search Filter is `(objectClass=sunIdentityServerDevice)` , then the actual user search filter will be: `(&(uid=*)(objectClass=sunIdentityServ erDevice))`

LDAP Agents Object Class

Defines the object classes for agents. When an agent is created, the list of user object classes will be added to the agent's attributes list

LDAP Agents Attributes

Defines the list of attributes associated with an agent. Any attempt to read/write agent attributes that are not on this list is not allowed. The attributes are case-sensitive. The object classes and attribute schema must be defined in Directory Server before you define the object classes and attribute schema here.

Identity Types that can be Authenticated

Specifies that this data store can authenticate user and/or agent identity types when the authentication module mode for the realm is set to Data Store.

Persistent Search Base DN

Defines the base DN to use for persistent search. Some LDAPv3 servers only support persistent search at the root suffix level.

Persistent Search Filter

Defines the filter that will return the specific changes to directory server entries. The data store will only receive the changes that match the defined filter.

Persistent Search Maximum Idle Time Before Restart

Defines the maximum idle time before restarting the persistence search. The value must be great than 1. Values less than or equal to 1 will restart the search irrespective of the idle time of the connection.

If Federated Access Manager is deployed with a load balancer, some load balancers will time out if it has been idle for a specified amount of time. In this case, you should set the Persistent Search Maximum Idle Time Before Restart to a value less than the specified time for the load balancer.

Maximum Number of Retries After Error Code

Defines the maximum number of retries for the persistent search operation if it encounters the error codes specified in LDAPException Error Codes to Retry On.

The Delay Time Between Retries

Specifies the time to wait before each retry. This only applies to persistent search connection.

LDAPException Error Codes to Retry

Specifies the error codes to initiate a retry for the persistent search operation. This attribute is only applicable for the persistent search, and not for all LDAP operations.

Caching

If enabled, this allows Federated Access Manager to cache data retrieved from the data store.

Maximum Age of Cached Items

Specifies the maximum time data is stored in the cache before it is removed. The values are defined in seconds.

Maximum Size of the Cache

Specifies the maximum size of the cache. The larger the value, the more data can be stored, but it will require more memory. The values are defined in bytes.

CHAPTER 4

Managing Authentication

Federation Access Manager provides the Authentication Service, which allows you to configure any number of authentication options that allow or deny a user access to a particular resource. To configure the authentication process, you must define one or more instances of an authentication module, establish an authentication chain, or both. The following sections describe how to configure authentication for your deployment.

Configuring the Authentication Service

Before you configure the authentication process for your deployment, you fist configure the Federated Access Manager Authentication service. There are different levels at which you can configure this service:

1. Use global properties if no overriding value is defined in the configured realm or a specified authentication module. These default global values are defined in the Core authentication service, or in `amAuth.xml`.
2. Use the Core authentication service to define properties under the realm's General Properties for authentication configuration for the realm's users, roles, services, and so forth.
3. Use the specific authentication module properties to configure a module type for a configured realm or authentication chain. Authentication chains are one or more authentication module instances that are configured so that a user must pass authentication credentials to all of them. For more information, see XXXXXXXXX.

General Authentication Properties

The General authentication attributes are globally defined for the Federated Access Manager deployment, or more specifically, for each configured realm. These attributes are defined in the Core authentication service and are added to each new realm when it is created. Once added to a realm. the new values can be modified by the realm's administrator. The values are then used if no overriding value is defined in the specified authentication module instance or authentication chain.

▼ To Modify Global General Authentication Properties

1. **Login as the administrator of the top-level realm.**
2. **Click the Configuration tab.**
3. **Click Core in the Service Name list under the Authentication heading.**
4. **Modify the global attributes by adding or changing the values.**

   Click Help for attribute descriptions.
5. **(Optional) Modify the attributes specific to the top-level realm by adding or changing the values of the Realm attributes.**

   These attributes values are specific to the top-level realm only. These modifications can also made by navigating to the General Authentication properties under the top-level realm. See XXXXXXXXxRRRRRREFFF.
6. **Click Save.**
7. **Click Back to Configuration.**
8. **Logout.**

▼ To Modify the General Authentication Properties for a Realm

Realm attributes are applied to the realm under which they are configured. Many Authentication service attributes are defined as realm attributes because authentication is performed at the realm level.

1. **Login as the administrator of the realm you are configuring.**
2. **Click the Access Control tab.**
3. **Click the name of the realm to be modified.**
4. **Click the Authentication tab.**
5. **Click Advanced Properties button.**

   This displays the attributes in the Core authentication service.

6 **Modify the attributes.**

Click Help for attribute descriptions.

7 **Click Save.**

8 **Click Back to Authentication.**

9 **Logout.**

Authentication Configuration Service

The Authentication Configuration properties allow administrators to define an authentication chain for a specific role within a configured realm. When you assign Authentication Configuration to a role, you can specify a particular authentication chain. For more information on configuring authentication for roles, see XXXXXXXXXXXXXXXXXXXx.

▼ To Assign Authentication Configuration to a Role

1 **Login as the administrator of the realm you will be configuring.**

2 **Select the Access Control tab.**

3 **Click the name of the realm which you are modifying.**

4 **Select the Subjects tab.**

5 **Select the Roles sub-tab.**

6 **Click the name of the role you are configuring.**

7 **Select the Services tab.**

8 **Click Add under the Services heading.**

9 **Select Authentication Configuration and click Next.**

10 **Click the radio button next the appropriate authentication chain.**

This step assumes that an authentication has already been created for the role. You may click Empty and define the authentication chain later. For more information, see XXXXXXXXXXXXXXXXXXXXXXx.

11 **Click Finish and then Back to Subjects.**

12 **Logout.**

Authentication Chaining

One or more authentication modules can be configured so a user must pass authentication credentials to all of them. This is referred to as *authentication chaining* . Authentication chaining in Federated Access Manager is achieved using the JAAS framework integrated in the Authentication Service.

▼ To Create a New Authentication Chain

1 **Click the name of the realm for which you wish to add a new authentication chain.**

2 **Select the Authentication tab.**

3 **Click New in the Authentication Chaining list.**

4 **Enter a name for the authentication chain.**

5 **Click Create.**

6 **Click Add to define the authentication module instance that you wish to include in the chain. To do so, select the module instance name from the Instance list. The module instance names displayed in this list are created in the Module Instances attribute.**

7 **Select the criteria for the chain. These flags establish an enforcement criteria for the authentication module for which they are defined. There is hierarchy for enforcement. Required is the highest and Optional is the lowest:**

Requisite	The module instance is required to succeed. If it succeeds, authentication continues down the Authentication Chaining list. If it fails, control immediately returns to the application (authentication does not proceed down the Authentication Chaining list).
Required	Authentication to this module is required to succeed. If any of the required modules in the chain fails, the whole authentication chain will ultimately fail. However, whether a required module succeeds or fails, the control will continue down to the next module in the chain.
Sufficient	The module instance is not required to succeed. If it does succeed, control immediately returns to the application (authentication does not proceed down the module instance list). If it fails, authentication continues down the Authentication Chaining list.

Optional	The module instance is not required to succeed. If it succeeds or fails, authentication still continues to proceed down the Authentication Chaining list.

8 **Enter options for the chain. This enables additional options for the module as a key=value pair. Multiple options are separated by a space.**

9 **Define the following attributes:**

Successful Login URL	Specifies the URL that the user will be redirected to upon successful authentication.
Failed Login URL	Specifies the URL that the user will be redirected to upon unsuccessful authentication.
Authentication Post Processing Class	Defines the name of the Java class used to customize the post authentication process after a login success or failure.

10 **Click Save.**

Authentication Modules and Chains

An authentication module is a plug-in that collects information from a user requesting access to resource, such as a user ID and password, and then checks the information against entries in a database. If a user provides information that meets the authentication criteria, then the user is granted access to the requested resource. If the user provides information that does not meet authentication criteria, the user is denied access to the requested resource. Federated Access Manager provides a number of authentication modules that you can configure within a realm and you can configure multiple instances of an authentication module. For example, you can configure two instances of the LDAP authentication module, each pointing to a different LDAP database within the same realm.

Additionally, you can configure different authentication modules so a user must pass authentication credentials to all of them in order to gain access to a protected resource. This is called an *authentication chain*.

The following sections contain information on how to configure authentication module instances, provide detailed information about the individual authentication modules that Federated Access Manager provides, and how to create authentication chains:

Adding Authentication Module Instances

Before you add authentication module instances, you can define default, global values for the module. These values will then be carried over the created instance. Once the module instance is added to the realm, you can modify the values as you see fit.

The steps to add an authentication module instance to a realm are the same for each authentication module type, however some authentication modules may require pre-configuration steps for your system. See XXXXXXXXXXXX for definitions of each Federated Access Manager authentication module instance and pre-configuring steps, if required.

▼ To Add a New Authentication Module Instance

1. **Click the name of the realm for which you wish to add a new authentication module instance.**
2. **Select the Authentication tab.**

   Note – The Administrator Authentication Configuration button 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 Federated Access Manager console is accessed.

3. **Click New in the Module Instances list.**
4. **Enter a Name for the authentication module instance. The names must be unique.**
5. **Select the Type of authentication module type for the realm.**
6. **Click OK.**
7. **Click the name of the newly created module instance and edit the properties for that module.**

   See the Authentication section in the Federated Access Manager, or click Help for definitions for the properties for each module type.

8. **Repeat these steps to add multiple module instances.**

Authentication Modules

Note – Some of the authentication module types require pre-configuration before they can be used as authentication instances. The configuration steps, if necessary, are listed in the module type descriptions.

Active Directory

The Active Directory authentication module performs authentication in a similar manner to the LDAP module, but uses Microsoft's Active Directory™ server (as opposed to Directory Server in LDAP authentication module). Although the LDAP authentication module can be configured for an Active Directory server, this module allows you have both LDAP and Active Directory authentication exist under the same realm.

Note – For this release, the Active Directory authentication module only supports user authentication. Password policy is only supported in the LDAP authentication module.

Anonymous

By default, when this module is enabled, a user can log in to Federated Access Manager as an *anonymous* user. A list of anonymous users can also be defined for this module by configuring the Valid Anonymous User List attribute. Granting anonymous access means that it can be accessed without providing a password. Anonymous access can be limited to specific types of access (for example, access for read or access for search) or to specific subtrees or individual entries within the directory.

Certificate

Certificate-based Authentication involves using a personal digital certificate (PDC) to identify and authenticate a user. A PDC can be configured to require a match against a PDC stored in Directory Server, and verification against a Certificate Revocation List.

There are a number of things that need to be accomplished before adding the Certificate-based Authentication module to a realm. First, the web container that is installed with the Federated Access Manager needs to be secured and configured for Certificate-based Authentication.

Note – If you are configuring Federated Access Manager Certificate authentication with an SSL-enabled Sun Java System WebsServer 6.1 instance, and wish to have the WebServer defined to accept both certificate based and non certificate based authentication requests, you must set the following value in the WebServer's `obj.conf` file:

```
PathCheck fn="get-client-cert" dorequest="1" require="0"
```

This is due to a limitation in the WebServer console when setting the optional attribute for this behavior.

Before enabling the Certificate-based module, see Chapter 6, "Using Certificates and Keys" in the *Sun ONE Web Server 6.1 Administrator's Guide for* these initial Web Server configuration steps. This document can be found at the following location:

```
http://docs.sun.com/db/prod/s1websrv#hic
```

Or, see the *Sun ONE Application Server Administrator's Guide to Security* at the following location:

http://docs.sun.com/db/prod/s1appsrv#hic (`http://docs.sun.com/db/prod/s1appsrv#hic`)

Note – Each user that will authenticate using the certificate-based module must request a PDC for the user's browser. Instructions are different depending upon the browser used. See your browser's documentation for more information.

In order to add this module, you must log in to Federated Access Manager as the realm Administrator and have Federated Access Manager and the web container configured for SSL and with client authentication enabled. For more information, see Configuring Access Manager in SSL Mode in the *Access Manager Post Installation Guide*.

Data Store

The Data Store authentication module allows a login using the Identity Repository of the realm to authenticate users. Using the Data Store module removes the requirement to write an authentication plug- in module, load, and then configure the authentication module if you need to authenticate against the same data store repository. Additionally, you do not need to write a custom authentication module where flat-file authentication is needed for the corresponding repository in that realm.

This authentication type provides a degree of convenience when configuring Federated Access Manager authentication. In releases prior to Federated Access Manager 8.0, if you wanted users in an LDAPv3 data store to be able to authenticate to their realm, you had to:

- Configure the LDAPv3 data store
- Configure an LDAP authentication module instance to reference the same realm subjects

The Data Store authentication module lets users defined in the realm's identity repository authenticate. No LDAP authentication configuration is necessary. For example, suppose a realm's identity repository includes an LDAPv3 data store, and suppose the same realm uses data store authentication. In this case, any user defined in the identity repository could authenticate to that realm.

HTTP Basic

This module uses basic authentication, which is the HTTP protocol's built-in authentication support. The web server issues a client request for username and password, and sends that information back to the server as part of the authorized request. Federated Access Manager retrieves the username and password and then internally authenticates the user to the LDAP authentication module. In order for HTTP Basic to function correctly, the LDAP authentication module must be added (adding the HTTP Basic module alone will not work). Once the user successfully authenticates, the user will be able to re-authenticate without being prompted for username and password.

JDBC

The Java Database Connectivity (JDBC) Authentication module provides a mechanism to allow Federated Access Manager to authenticate users through any SQL databases that provide JDBC technology-enabled drivers. The connection to the SQL database can be either directly through a JDBC driver, or a JNDI connection pool.

Note – This module has been tested on MySQL4.0 and Oracle 8i.

LDAP

With the LDAP Authentication module, when a user logs in, the user is required to bind to the LDAP Directory Server with a specific user DN and password. This is the default authenticating module for all realm-based authentication. If the user provides a user ID and password that are in the Directory Server, the user is allowed access to, and is set up with, a valid Federated Access Manager session. Both the Core and LDAP Authentication modules are automatically enabled for the default realm

Membership

Membership authentication is implemented similarly to personalized sites such as `my.site.com`, or `mysun.sun.com`. When this module is enabled, a user creates an account and personalizes it without the aid of an administrator. With this new account, the user can access it as a added user. The user can also access the viewer interface, saved on the user profile database as authorization data and user preferences.

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.

RADIUS

Federated Access Manager can be configured to work with a RADIUS server that is already installed. This is useful if there is a legacy RADIUS server being used for authentication in your enterprise. Enabling the RADIUS authentication module is a two-step process:

1. Configure the RADIUS server.

   For detailed instructions, see the RADIUS server documentation.
2. Register and enable the RADIUS authentication module.

Configuring RADIUS with Sun Java System Application Server

When the RADUIS client forms a socket connection to its server, by default, only the connect permission of the SocketPermissions is allowed in the Application Server's `server.policy` file. In order for RADUIS authentication to work correctly, permissions need to be granted for the following actions:

- accept
- connect
- listen
- resolve

To grant a permission for a socket connection, you must add an entry into Application Server's `server.policy` file. A SocketPermission consists of a host specification and a set of actions specifying ways to connect to that host. The host is specified as the following:

```
host = hostname | IPaddress:portrange:portrange = portnumber
| -portnumberportnumber-portnumber
```

The host is expressed as a DNS name, as a numerical IP address, or as local host (for the local machine). The wildcard "*" may be included once in a DNS name host specification. If it is included, it must be in the left-most position, as in `*.example.com`.

The port (or port range) is optional. A port specification of the form `N-`, where `N` is a port number, signifies all ports numbered `N` and above. A specification of the form `-N` indicates all ports numbered `N` and below.

The `listen` action is only meaningful when used with a local host. The `resolve` (resolve host/IP name service lookups) action is implied when any of the other actions are present.

For example, when creating SocketPermissions, note that if the following permission is granted to some code, it allows that code to connect to `port 1645` on `machine1.example.com`, and to accept connections on that port:

```
permission java.net.SocketPermission machine1.example.com:1645, "connect,accept";
```

Similarly, if the following permission is granted to some code, it allows that code to accept connections on, connect to, or listen to any port between 1024 and 65535 on the local host:

```
permission java.net.SocketPermission "machine1.example.com:1645", "connect,accept";
permission java.net.SocketPermission "localhost:1024-", "accept,connect,listen";
```

Note – Granting code permission to accept or make connections to remote hosts may cause problems, because malevolent code can then more easily transfer and share confidential data among parties who may not otherwise have access to the data. Make sure to give only appropriate permissions by specifying exact port number instead of allowing a range of port numbers

---

SafeWord

Federated Access Manager can be configured to handle SafeWord Authentication requests to Secure Computing's SafeWord™ or SafeWord PremierAccess™ authentication servers. Federated Access Manager provides the client portion of SafeWord authentication. The SafeWord server may exist on the system on which Federated Access Manager is installed, or on a separate system.

Configuring SafeWord with Sun Java System Application Server

When the SafeWord client forms a socket connection to its server, by default, only the `connect` permission of the SocketPermissions is allowed in the Application Server's `server.policy` file. In order for SafeWord authentication to work correctly, permissions need to be granted for the following actions:

- accept
- connect
- listen
- resolve

To grant a permission for a socket connection, you must add an entry into Application Server's `server.policy` file. A SocketPermission consists of a host specification and a set of actions specifying ways to connect to that host. The host is specified as the following:

```
host = (hostname | IPaddress)[:portrange] portrange =
portnumber | -portnumberportnumber-[portnumber]
```

The host is expressed as a DNS name, as a numerical IP address, or as localhost (for the local machine). The wildcard "*" may be included once in a DNS name host specification. If it is included, it must be in the left-most position, as in `*.example.com`.

The port (or portrange) is optional. A port specification of the form `N-`, where `N` is a port number, signifies all ports numbered `N` and above. A specification of the form `-N` indicates all ports numbered `N` and below.

The `listen` action is only meaningful when used with a localhost. The `resolve` (resolve host/IP name service lookups) action is implied when any of the other actions are present.

For example, when creating SocketPermissions, note that if the following permission is granted to some code, it allows that code to connect to `port 1645` on `machine1.example.com,` and to accept connections on that port:

```
permission java.net.SocketPermission machine1.example.com:5030, "connect,accept";
```

Similarly, if the following permission is granted to some code, it allows that code to accept connections on, connect to, or listen to any port between 1024 and 65535 on the local host:

```
permission java.net.SocketPermission "machine1.example.com:5030", "connect,accept";
permission java.net.SocketPermission "localhost:1024-", "accept,connect,listen";
```

Note – Granting code permission to accept or make connections to remote hosts may cause problems, because malevolent code can then more easily transfer and share confidential data among parties who may not otherwise have access to the data. Make sure to give only appropriate permissions by specifying exact port number instead of allowing a range of port numbers

---

SAML

The Security Assertion Markup Language (SAML) authentication module receives and validates SAML Assertions on a target server. SAML SSO will only work if this module is configured on the target machine, including after an upgrade (for example, Access Manager 8.0 to Federated Access Manager 8.0).

SecurID

Federated Access Manager can be configured to handle SecurID Authentication requests to RSA's ACE/Server authentication servers. Federated Access Manager provides the client portion of SecurID authentication. The ACE/Server may exist on the system on which Federated Access Manager is installed, or on a separate system. In order to authenticate locally-administered user IDs (see `admintool` (1M)), root access is required.

SecurID Authentication makes use of an authentication *helper*, `amsecuridd`, which is a separate process from the main Federated Access Manager process. Upon startup, this helper listens on a port for configuration information. If Federated Access Manager is installed to run as `nobody`, or a userid other than root, then the *FederatedAccessManager-base*`/SUNWam/share/bin/amsecuridd` process must still execute as root. For more information on the `amsecuridd` helper, see "The amSecurID Helper" in the Federated Access Manager Administration Reference.

Note – For this release of Federated Access Manager, 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 SPARC systems.

UNIX

Federated Access Manager can be configured to process authentication requests against Unix user IDs and passwords known to the Solaris or Linux system on which Federated Access Manager is installed. While there is only one realm attribute, and a few global attributes for Unix authentication, there are some system-oriented considerations. In order to authenticate locally-administered user IDS (see `admintool` (1M)), root access is required

Unix Authentication makes use of an authentication *helper*, `amunixd`, which is a separate process from the main Federated Access Manager process. Upon startup, this helper listens on a port for configuration information. There is only one Unix helper per Federated Access Manager to serve all of its realms.

If Federated Access Manager is installed to run as `nobody`, or a userid other than root, then the *FederatedAccessManager-base*`/SUNWam/share/bin/amunixd` process must still execute as root. The Unix authentication module invokes the `amunixd` daemon by opening a socket to `localhost:58946` to listen for Unix authentication requests. To run the `amunixd` helper process on the default port, enter the following command:

```
./amunixd
```

To run `amunixd` on a non-default port, enter the following command:

```
./amunixd [-c portnm] [ipaddress]
```

The ipaddress and portnumber is located in the `UnixHelper.ipadrs` (in IPV4 format) and `UnixHelper.port` attributes in `AMConfig.properties` . You can run `amunixd` through the `amserver` command line utility (`amserver` runs the process automatically, retrieving the port number and IP address from `AMConfig.properties`).

The `passwd` entry in the `/etc/nsswitch.conf` file determines whether the `/etc/passwd` and `/etc/shadow` files, or NIS are consulted for authentication.

Windows Desktop SSO

The Windows Desktop SSO Authentication module is a Kerberos-based authentication plug-in module used for Windows 2000™. It allows a user who has already authenticated to a Kerberos Distribution Center (KDC) to authenticate to Federated Access Manager without re-submitting the login criteria (Single Sign-on).

The user presents the Kerberos token to the Federated Access Manager through the SPNEGO (Simple and Protected GSS-API Negotiation Mechanism) protocol. In order to perform Kerberos-based Single Sign-on to Federated Access Manager through this authentication module, the user must, on the client side, support the SPNEGO protocol to authenticate itself. In general, any user that supports this protocol should be able to use this module to authenticate to Federated Access Manager. Depending on the availability of the token on the client side, this module provides a SPENGO token or a Kerberos token (in both cases, the protocols are the same). Microsoft Internet Explorer (5.01 or later) running on Windows 2000 (or later) currently supports this protocol. In addition, Mozilla 1.4 on Solaris (9 and 10) has SPNEGO support, but the token returned is only a KERBEROS token, because SPNEGO is not supported on Solaris.

Note – You must use JDK 1.4 or above to utilize the new features of Kerberos V5 authentication module and Java GSS API to perform Kerberos based SSO in this SPNEGO module.

Known Restriction with Internet Explorer

If you are using Microsoft Internet Explorer 6.x when for WindowsDesktopSSO authentication and the browser does not have access to the user's Kerberos/SPNEGO token that matches the (KDC) realm configured in the WindowsDesktopSSO module, the browser will behave incorrectly to other modules after it fails authenticating to the WindowsDesktopSSO module. The direct cause of the problem is that after Internet Explorer fails the WindowsDesktopSSO module, the browser becomes incapable of passing callbacks (of other modules) to Federated Access Manager, even if the callbacks are prompted, until the browser is restarted. Therefore all the modules coming after WindowsDesktopSSO will fail due to null user credentials.

See the following documentation for related information:

http://support.microsoft.com/default.aspx?scid=kb;en-us;308074 (`http://support.microsoft.com/default.aspx?scid=kb;en-us;308074`)

http://www.wedgetail.com/jcsi/sso/doc/guide/troubleshooting.html#ieNTLM (`http://support.microsoft.com/default.aspx?scid=kb;en-us;308074`)

Note – As of this release of Federated Access Manager, this restriction has been fixed by Microsoft. For more information, see the following documentation:

`http://www.microsoft.com/technet/security/bulletin/ms06-042.mspx`

Configuring Windows Desktop SSO

Enabling Windows Desktop SSO Authentication is a two-step process:

1. Create a User in the Windows 2000 Domain Controller.
2. Setup Internet Explorer.

▼ To Create a User in the Windows 2000 Domain Controller

1 In the domain controller, create a user account for the Federated Access Manager authentication module.

a. From the Start menu, go to Programs>Administration Tools.

b. Select Active Directory Users and Computers.

c. Go to Computers>New>computer and add the client computer's name. If you are using Windows XP, this step is performed automatically during the domain controller account configuration.

d. Go to Users>New>Users and create a new user with the Federated Access Manager host name as the User ID (login name). The Federated Access Manager host name should not include the domain name.

2 Associate the user account with a service provider name and export the keytab files to the system in which Federated Access Manager is installed. To do so, run the following commands:

```
ktpass -princ host/hostname.domainname@DCDOMAIN -pass password -mapuser userName-out
hostname.host.keytab
ktpass -princ HTTP/hostname.domainname@DCDOMAIN -pass
password -mapuser userName-out hostname.HTTP.keytab
```

Note – The `ktpass` utilities are not installed as part of the Windows 2000 server. You must install it from the installation CD to the `c:\program files\support` tools directory.

The `ktpass` command accepts the following parameters:

hostname. The host name (without the domain name) on which Federated Access Manager runs.

domainname . The Federated Access Manager domain name.

DCDOMAIN. The domain name of the domain controller. This may be different from the Federated Access Manager domain name.

password . The password of the user account. Make sure that password is correct, as `ktpass` does not verify passwords.

userName. The user account ID. This should be the same as hostname.

Note – Make sure that both keytab files are kept secure.

The service template values should be similar to the following example:

Service Principal: `HTTP/machine1.EXAMPLE.COM@ISQA.EXAMPLE.COM`

Keytab File Name: `/tmp/machine1.HTTP.keytab`

Kerberos Realm: `ISQA.EXAMPLE.COM`

Kerberos Server Name: `machine2.EXAMPLE.com`

Return Principal with Domain Name: `false`

Authentication Level: `22`

Note – If you are using Windows 2003 or Windows 2003 Service Packs,, use the following `ktpass` command syntax:

```
ktpass /out filename /mapuser username /princ HTTP/hostname.domainname
     /crypto encryptiontype /rndpass /ptype principaltype /target domainname
```

For example:

```
ktpass /out demo.HTTP.keytab /mapuser http
/princ HTTP/demo.identity.sun.com@IDENTITY.SUN.COM /crypto RC4-HMAC-NT
/rndpass /ptype KRB5_NT_PRINCIPAL /target IDENTITY.SUN.COM
```

For syntax definitions, see `http://technet2.microsoft.com/WindowsServer/en/library/64042138-9a5a-4981-84e9-d576a8db0d051033.mspx?mfr=true` web site.

---

3 Restart the server.

▼ To Set Up Internet Explorer

These steps apply to Microsoft Internet Explorer™ 6 and later. If you are using an earlier version, make sure that Federated Access Manager is in the browser's internet zone and enable Native Windows Authentication.

1 In the Tool menu, go to Internet Options>Advanced/Security>Security.

2 Select the Integrated Windows Authentication option.

3 Go to Security>Local Internet.

a. Select Custom Level. In the User Authentication/Logon panel, select the Automatic Logon Only in Intranet Zone option.

b. Go to Sites and select all of the options.

c. Click Advanced and add the Federated Access Manager to the local zone (if it is not added already).

Windows NT

Federated Access Manager can be configured to work with an Windows NT /Windows 2000 server that is already installed. Federated Access Manager provides the client portion of NT authentication.

1. Configure the NT server. For detailed instructions, see the Windows NT server documentation.
2. Before you can add and enable the Windows NT authentication module, you must obtain and install a Samba client to communicate with Federated Access Manager on your Solaris system.

Installing the Samba Client

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

```
FederatedAccessManager-base/SUNWam/bin
```

Samba Client is a file and print server for blending Windows and UNIX machines together without requiring a separate Windows NT/2000 Server. More information, and the download itself, can be accessed at http://wwws.sun.com/software/download/products/3e3af224.html.

Red Hat Linux ships with a Samba client, located in the following directory:

```
/usr/bin
```

In order to authenticate using the Windows NT Authentication module for Linux, copy the client binary to the following Federated Access Manager directory:

```
FederatedAccessManager-base/sun/identity/bin
```

Note – If you have multiple interfaces, extra configuration is required. Multiple interfaces can be set by configuration in the `smb.conf` file so it passes to the `mbclient`.

Authentication Types

The Authentication Service provides different ways in which authentication can be applied. These different authentication methods can be accessed by specifying Login URL parameters, or through the authentication APIs (see Chapter 2, "Using Authentication APIs and SPIs," in *Sun Java System Access Manager 7.1 Developer's Guide* in the Developer's Guide for more information). Before an authentication module can be configured, the Core authentication service attribute realm Authentication Modules must be modified to include the specific authentication module name.

The Authentication Configuration service is used to define authentication modules for any of the following authentication types:

- "Realm-based Authentication" on page 64
- "Organization-based Authentication" on page 67
- "Role-based Authentication" on page 70
- "Service-based Authentication" on page 73
- "User-based Authentication" on page 76
- "Authentication Level-based Authentication" on page 78
- "Module-based Authentication" on page 80

Once an authentication module is defined for one of these authentication types, the module can be configured to supply redirect URLs, as well as a post-processing Java class specification, based on a successful or failed authentication process.

How Authentication Types Determine Access

For each of these methods, the user can either pass or fail the authentication. Once the determination has been made, each method follows this procedure. Step 1 through Step 3 follows a successful authentication; Step 4 follows both successful and failed authentication.

1. Federated Access Manager confirms whether the authenticated user(s) is defined in the Directory Server data store and whether the profile is active.

   The User Profile attribute in the Core Authentication module can be defined as `Required`, `Dynamic`, `Dynamic with User Alias`, or `Ignored`. Following a successful authentication, Federated Access Manager confirms whether the authenticated user(s) is defined in the Directory Server data store and, if the User Profile value is `Required`, confirms that the profile is active. (This is the default case.) If the User Profile is `Dynamically Configured`, the Authentication Service will create the user profile in the Directory Server data store. If the User Profile is set to `Ignore`, the user validation will not be done.
2. Execution of the Authentication Post Processing SPI is accomplished.

   The Core Authentication module contains an Authentication Post Processing Class attribute which may contain the authentication post-processing class name as its value. `AMPostAuthProcessInterface` is the post-processing interface. It can be executed on either successful or failed authentication or on logout.
3. The following properties are added to, or updated in, the session token and the user's session is activated.

   realm. This is the DN of the realm to which the user belongs.

   Principal. This is the DN of the user.

   Principals. This is a list of names to which the user has authenticated. (This property may have more then one value defined as a pipe separated list.)

   UserId. This is the user's DN as returned by the module, or in the case of modules other than LDAP or Membership, the user name. (All Principals must map to the same user. The UserID is the user DN to which they map.)

   Note – This property may be a non-DN value.

   UserToken. This is a user name. (All Principals must map to the same user. The UserToken is the user name to which they map.)

   Host. This is the host name or IP address for the client.

   authLevel. This is the highest level to which the user has authenticated.

   AuthType. This is a pipe separated list of authentication modules to which the user has authenticated (for example, `module1|module2|module3`).

clientType. This is the device type of the client browser.

Locale. This is the locale of the client.

CharSet. This is the determined character set for the client.

Role. Applicable for role-based authentication only, this is the role to which the user belongs.

Service. Applicable for service-based authentication only, this is the service to which the user belongs.

4. Federated Access Manager looks for information on where to redirect the user after either a successful or failed authentication.

   URL redirection can be to either an Federated Access Manager page or a URL. The redirection is based on an order of precedence in which Federated Access Manager looks for redirection based on the authentication method and whether the authentication has been successful or has failed. This order is detailed in the URL redirection portions of the following authentication methods sections.

URL Redirection

In the Authentication Configuration service, you can assign URL redirection for successful or unsuccessful authentication. The URLs, themselves, are defined in the Login Success URL and Login Failure URL attributes in this service. In order to enable URL redirection, you must add the Authentication Configuration service to your realm to make it available to configure for a role, realm, or user. Make sure that you add an authentication module, such as LDAP - REQUIRED, when adding the Authentication Configuration service.

Realm-based Authentication

This method of authentication allows a user to authenticate to an realm or sub-realm. It is the default method of authentication for Federated Access Manager. The authentication method for an realm is set by registering the Core Authentication module to the realm and defining the realm Authentication Configuration attribute.

Realm-based Authentication Login URLs

The realm for authentication can be specified in the User Interface Login URL by defining the `realm` Parameter or the `domain` Parameter. The realm of a request for authentication is determined from the following, in order of precedence:

1. The `domain` parameter.
2. The `realm` parameter.

3. The value of the `DNS Alias Names` attribute in the Administration service.

   After calling the correct realm, the authentication module(s) to which the user will authenticate are retrieved from the realm Authentication Configuration attribute in the Core Authentication Service. The login URLs used to specify and initiate realm-based authentication are:

   ```
   http://server_name.domain_name:port/amserver/UI/Login
   http://server_name.domain_name:port/amserver/UI/Login?domain=domain_name
   http://server_name.domain_name:port/amserver/UI/Login?realm=realm_name
   ```

   If there is no defined parameter, the realm will be determined from the server host and domain specified in the login URL.

Note – If a user is member of and is authenticated to a specific realm, and tries to authenticate to different realm, the only two parameters that are passed are `realm` and `module`. For example, if `User1` is a member of and authenticates to `realmA` and then tries to switch to or authenticate to `realmB`, the user will receive a warning page requesting to either start a new authentication to `realmB` with the module instance specified for `realmB`, or return to the existing authenticated session with `realmA`. If the user chooses to authenticate to `realmB`, only the realm name and module name (if specified) are passed and honored for determining the new authentication process.

Realm-based Authentication Redirection URLs

Upon a successful or failed organization-based authentication, Federated Access Manager looks for information on where to redirect the user. Following is the order of precedence in which the application will look for this information.

Successful realm-based Authentication Redirection URLs

The redirection URL for successful realm-based authentication is determined by checking the following places in order of precedence:

1. A URL set by the authentication module.
2. A URL set by a `goto` Login URL parameter.
3. A URL set in the `clientType` custom files for the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).
4. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's role entry.
5. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.

6. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute as a global default.
7. A URL set in the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).
8. A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's role entry.
9. A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.
10. A URL set in the `iplanet-am-auth-login-success-url` attribute as a global default.

Failed Realm-based Authentication Redirection URLs

The redirection URL for failed realm-based authentication is determined by checking the following places in the following order:

1. A URL set by the authentication module.
2. A URL set by a `gotoOnFail` Login URL parameter.
3. A URL set in the `clientType` custom files for the `iplanet-am-user-failure-url` attribute of the user's entry (`amUser.xml`).
4. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.
5. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.
6. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute as a global default.
7. A URL set for the `iplanet-am-user-failure-url` attribute in the user's entry (`amUser.xml`).
8. A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.
9. A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.
10. A URL set for the `iplanet-am-auth-login-failure-url` attribute as the global default.

To Configure Realm-Based Authentication

Authentication modules are set for realms by first adding the Core Authentication service to the realm.

▼ To Configure The Realms's Authentication Attributes

1 **Navigate to the realm for which you wish to add the Authentication Chain.**

2 **Click the Authentication tab.**

3 Select the Default Authentication Chain.

4 Select the Administrator Authentication Chain from the pull down menu. This attribute can be used if the authentication module for administrators needs to be different from the module for end users. The default authentication module is LDAP.

5 Once you have defined the authentication chains, click Save.

Organization-based Authentication

This authentication type only applies to Federated Access Manager deployments that have been installed in Legacy mode.

This method of authentication allows a user to authenticate to an organization or sub-organization. It is the default method of authentication for Federated Access Manager . The authentication method for an organization is set by registering the Core Authentication module to the organization and defining the Organization Authentication Configuration attribute.

Organization-based Authentication Login URLs

The organization for authentication can be specified in the User Interface Login URL by defining the `org` Parameter or the `domain` Parameter. The organization of a request for authentication is determined from the following, in order of precedence:

1. The `domain` parameter.
2. The `org` parameter.
3. The value of the `DNS Alias Names` (Organization alias names) attribute in the Administration Service.

   After calling the correct organization, the authentication module(s) to which the user will authenticate are retrieved from the Organization Authentication Configuration attribute in the Core Authentication Service. The login URLs used to specify and initiate organization-based authentication are:

   ```
   http://server_name.domain_name:port/amserver/UI/Login
   http://server_name.domain_name:port/amserver/UI/Login?domain=domain_name
   http://server_name.domain_name:port/amserver/UI/Login?org=org_name
   ```

   If there is no defined parameter, the organization will be determined from the server host and domain specified in the login URL.

Note – If a user is member of and is authenticated to a specific organization, and tries to authenticate to different organization, the only two parameters that are passed are `org` and `module`. For example, if `User1` is a member of and authenticates to `orgA` and then tries to switch to or authenticate to `orgB`, the user will receive a warning page requesting to either start a new authentication to `orgB` with the module instance specified for `orgB`, or return to the existing authenticated session with `orgA`. If the user chooses to authenticate to `orgB`, only the organization name and module name (if specified) are passed and honored for determining the new authentication process.

Organization-based Authentication Redirection URLs

Upon a successful or failed organization-based authentication, Federated Access Manager looks for information on where to redirect the user. Following is the order of precedence in which the application will look for this information.

Successful Organization-based Authentication Redirection URLs

The redirection URL for successful organization-based authentication is determined by checking the following places in order of precedence:

1. A URL set by the authentication module.
2. A URL set by a `goto` Login URL parameter.
3. A URL set in the `clientType` custom files for the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).
4. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's role entry.
5. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's organization entry.
6. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute as a global default.
7. A URL set in the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).
8. A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's role entry.
9. A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's organization entry.
10. A URL set in the `iplanet-am-auth-login-success-url` attribute as a global default.

Failed Organization-based Authentication Redirection URLs

The redirection URL for failed organization-based authentication is determined by checking the following places in the following order:

1. A URL set by the authentication module.
2. A URL set by a `gotoOnFail` Login URL parameter.
3. A URL set in the `clientType` custom files for the `iplanet-am-user-failure-url` attribute of the user's entry (`amUser.xml`).
4. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.
5. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's organization entry.
6. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute as a global default.
7. A URL set for the `iplanet-am-user-failure-url` attribute in the user's entry (`amUser.xml`).
8. A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.
9. A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's organization entry.
10. A URL set for the `iplanet-am-auth-login-failure-url` attribute as the global default.

To Configure Organization-Based Authentication

Authentication modules are set for an organization by first adding the Core Authentication service to the organization.

▼ To Configure The Organizations's Authentication Attributes

1 **Navigate to the organization for which you wish to add the Authentication Chain.**

2 **Click the Authentication tab.**

3 **Select the Default Authentication Chain.**

4 **Select the Administrator Authentication Chain from the pull down menu. This attribute can be used if the authentication module for administrators needs to be different from the module for end users. The default authentication module is LDAP.**

5 **Once you have defined the authentication chains, click Save.**

Role-based Authentication

This method of authentication allows a user to authenticate to a role (either static or filtered) within an realm or sub realm.

Note – The Authentication Configuration Service must first be registered to the realm before it can be registered as an instance to the role.

For authentication to be successful, the user must belong to the role and they must authenticate to each module defined in the Authentication Configuration Service instance configured for that role. For each instance of role-based authentication, the following attributes can be specified:

Conflict Resolution Level. This sets a priority level for the Authentication Configuration Service instance defined for different roles that both may contain the same user. For example, if `User1` is assigned to both `Role1` and `Role2`, a higher conflict resolution level can be set for `Role1` so when the user attempts authentication, `Role1` will have the higher priority for success or failure redirects and post-authentication processes.

Authentication Configuration. This defines the authentication modules configured for the role's authentication process.

Login Success URL. This defines the URL to which a user is redirected on successful authentication.

Login Failed URL. This defines the URL to which a user is redirected on failed authentication.

Authentication Post Processing Classes. This defines the post-authentication interface.

Role-based Authentication Login URLs

Role-based authentication can be specified in The User Interface Login URL by defining a role Parameter. After calling the correct role, the authentication module(s) to which the user will authenticate are retrieved from the Authentication Configuration Service instance defined for the role.

The login URLs used to specify and initiate this role-based authentication are:

```
http://server_name.domain_name:port/amserver/UI/Login?role=role_name
http://server_name.domain_name:port/amserver/UI/Login?realm=realm_name&role=role_name
```

If the realm Parameter is not configured, the realm to which the role belongs is determined from the server host and domain specified in the login URL itself.

Role-based Authentication Redirection URLs

Upon a successful or failed role-based authentication, Federated Access Manager looks for information on where to redirect the user. Following is the order of precedence in which the application will look for this information.

Successful Role-based Authentication Redirection URLs

The redirection URL for successful role-based authentication is determined by checking the following places in the following order:

1. A URL set by the authentication module.
2. A URL set by a `goto` Login URL parameter.
3. A URL set in the `clientType` custom files for the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).
4. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the role to which the user has authenticated.
5. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of another role entry of the authenticated user. (This option is a fallback if the previous redirection URL fails.)
6. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.
7. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute as a global default.
8. A URL set in the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).
9. A URL set in the `iplanet-am-auth-login-success-url` attribute of the role to which the user has authenticated.
10. A URL set in the `iplanet-am-auth-login-success-url` attribute of another role entry of the authenticated user. (This option is a fallback if the previous redirection URL fails.)
11. A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.
12. A URL set in the `iplanet-am-auth-login-success-url` attribute as a global default.

Failed Role-based Authentication Redirection URLs

The redirection URL for failed role-based authentication is determined by checking the following places in the following order:

1. A URL set by the authentication module.
2. A URL set by a `goto` Login URL parameter.

3. A URL set in the `clientType` custom files for the `iplanet-am-user-failure-url` attribute of the user's profile (`amUser.xml`).
4. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the role to which the user has authenticated.
5. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of another role entry of the authenticated user. (This option is a fallback if the previous redirection URL fails.)
6. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.
7. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute as a global default.
8. A URL set in the `iplanet-am-user-failure-url` attribute of the user's profile (`amUser.xml).`
9. A URL set in the `iplanet-am-auth-login-failure-url` attribute of the role to which the user has authenticated.
10. A URL set in the `iplanet-am-auth-login-failure-url` attribute of another role entry of the authenticated user. (This option is a fallback if the previous redirection URL fails.)
11. A URL set in the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.
12. A URL set in the `iplanet-am-auth-login-failure-url` attribute as a global default.

▼ To Configure Role-Based Authentication

1 Navigate to the realm (or organization) to which you will add the authentication configuration service.

2 Click the Subjects tab.

3 Filtered Roles or Roles.

4 Select the role for which to set the authentication configuration.

5 Select the Default Authentication Chain that you wish to enable.

6 Click Save.

Note – If you are creating a new role, the Authentication Configuration service is not automatically assigned to it. Make sure that you select the Authentication Configuration service option at the top of the role profile page before you create it.

When role-based authentication is enabled, the LDAP authentication module can be left as the default, as there is no need to configure Membership.

Service-based Authentication

This method of authentication allows a user to authenticate to a specific service or application registered to an realm or sub realm. The service is configured as a Service Instance within the Authentication Configuration Service and is associated with an Instance Name. For authentication to be successful, the user must authenticate to each module defined in the Authentication Configuration service instance configured for the service. For each instance of service-based authentication, the following attributes can be specified:

Authentication Configuration. This defines the authentication modules configured for the service's authentication process.

Login Success URL. This defines the URL to which a user is redirected on successful authentication.

Login Failed URL. This defines the URL to which a user is redirected on failed authentication.

Authentication Post Processing Classes. This defines the post-authentication interface.

Service-based Authentication Login URLs

Service-based authentication can be specified in the User Interface Login URL by defining a service Parameter. After calling the service, the authentication module(s) to which the user will authenticate are retrieved from the Authentication Configuration service instance defined for the service.

The login URLs used to specify and initiate this service-based authentication are:

```
http://server_name.domain_name:port/amserver/UI/
Login?service=auth-chain-name
```

and

```
http://server_name.domain_name:port/amserver
     /UI/Login?realm=realm_name&service=auth-chain-name
```

If there is no configured `org` parameter, the realm will be determined from the server host and domain specified in the login URL itself.

Service-based Authentication Redirection URLs

Upon a successful or failed service-based authentication, Federated Access Manager looks for information on where to redirect the user. Following is the order of precedence in which the application will look for this information.

Successful Service-based Authentication Redirection URLs

The redirection URL for successful service-based authentication is determined by checking the following places in the following order:

1. A URL set by the authentication module.
2. A URL set by a `goto` Login URL parameter.
3. A URL set in the `clientType` custom files for the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).
4. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the service to which the user has authenticated.
5. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's role entry.
6. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.
7. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute as a global default.
8. A URL set in the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).
9. A URL set in the `iplanet-am-auth-login-success-url` attribute of the service to which the user has authenticated.
10. A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's role entry.
11. A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.
12. A URL set in the `iplanet-am-auth-login-success-url` attribute as a global default.

Failed Service-based Authentication Redirection URLs

The redirection URL for failed service-based authentication is determined by checking the following places in the following order:

1. A URL set by the authentication module.
2. A URL set by a `goto` Login URL parameter.

3. A URL set in the `clientType` custom files for the `iplanet-am-user-failure-url` attribute of the user's profile (`amUser.xml`).
4. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the service to which the user has authenticated.
5. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.
6. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.
7. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute as a global default.
8. A URL set in the `iplanet-am-user-failure-url` attribute of the user's profile (`amUser.xml`).
9. A URL set in the `iplanet-am-auth-login-failure-url` attribute of the service to which the user has authenticated.
10. A URL set in the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.
11. A URL set in the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.
12. A URL set in the `iplanet-am-auth-login-failure-url` attribute as a global default.

▼ To Configure Service-Based Authentication

Authentication modules are set for services after adding the Authentication Configuration service. To do so:

1 **Chose the realm to which you wish to configure service-based authentication.**

2 **Click the Authentication tab.**

3 **Create the authentication module instances.**

4 **Create the authentication chains.**

5 **Click Save.**

6 **To access service-based authentication for the realm, enter the following address:**

`http://`*server_name.domain_name:port*`/amserver/UI/Login?`
`realm=`*realm_name&service=auth-chain-name*

User-based Authentication

This method of authentication allows a user to authenticate to an authentication process configured specifically for the user. The process is configured as a value of the User Authentication Configuration attribute in the user's profile. For authentication to be successful, the user must authenticate to each module defined.

User-based Authentication Login URLs

User-based authentication can be specified in the User Interface Login URL by defining a user Parameter. After calling the correct user, the authentication module(s) to which the user will authenticate are retrieved from the User Authentication Configuration instance defined for the user.

The login URLs used to specify and initiate this role-based authentication are:

```
http://server_name.domain_name:port/amserver/UI/Login?user=user_name
http://server_name.domain_name:port/amserver/UI/Login?org=org_name&user=user_name
```

If there is no configured realm Parameter, the realm to which the role belongs will be determined from the server host and domain specified in the login URL itself.

User Alias List Attribute

On receiving a request for user-based authentication, the Authentication service first verifies that the user is a valid user and then retrieves the Authentication Configuration data for them. In the case where there is more then one valid user profile associated with the value of the user Login URL parameter, all profiles must map to the specified user. The User Alias Attribute (`iplanet-am-user-alias-list`) in the User profile is where other profiles belonging to the user can be defined. If mapping fails, the user is denied a valid session. The exception would be if one of the users is a top-level admin whereby the user mapping validation is not done and the user is given top—level Admin rights.

User-based Authentication Redirection URLs

Upon a successful or failed user-based authentication, Federated Access Manager looks for information on where to redirect the user. Following is the order of precedence in which the application will look for this information.

Successful User-based Authentication Redirection URLs

The redirection URL for successful user-based authentication is determined by checking the following places in order of precedence:

1. A URL set by the authentication module.

2. A URL set by a `goto` Login URL parameter.
3. A URL set in the `clientType` custom files for the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).
4. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's role entry.
5. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.
6. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute as a global default.
7. A URL set in the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).
8. A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's role entry.
9. A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.
10. A URL set in the `iplanet-am-auth-login-success-url` attribute as a global default.

Failed User-based Authentication Redirection URLs

The redirection URL for failed user-based authentication is determined by checking the following places in the following order:

1. A URL set by the authentication module.
2. A URL set by a `gotoOnFail` Login URL parameter.
3. A URL set in the `clientType` custom files for the `iplanet-am-user-failure-url` attribute of the user's entry (`amUser.xml`).
4. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.
5. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.
6. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute as a global default.
7. A URL set for the `iplanet-am-user-failure-url` attribute in the user's entry (`amUser.xml`).
8. A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.
9. A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.
10. A URL set for the `iplanet-am-auth-login-failure-url` attribute as the global default.

▼ To Configure User-Based Authentication

1. **Navigate to the realm in which you wish to configure authentication for the user.**

2. **Click the Subjects tab and click Users.**

3. **Click the name of the user you wish to modify**

   The User Profile is displayed.

   Note – If you are creating a new user, the Authentication Configuration service is not automatically assigned to the user. Make sure that you select the Authentication Configuration service option in the Service profile before you create the user. If this option is not selected, the user will not inherit the authentication configuration defined at for the role.

   ---

4. **In the User Authentication Configuration attribute, select the authentication chain you wish to apply.**

5. **Click Save.**

Authentication Level-based Authentication

Each authentication module can be associated with an integer value for its *authentication level*. Authentication levels can be assigned changing the corresponding value for the module's Authentication Level attribute. Higher authentication levels define a higher level of trust for the user once that user has authenticated to one or more authentication modules.

The authentication level will be set on a user's SSO token after the user has successfully authenticated to the module. If the user is required to authenticate to multiple authentication modules, and does so successfully, the highest authentication level value will be set in user's SSO token.

If a user attempts to access a service, the service can determine if the user is allowed access by checking the authentication level in user's SSO token. It then redirects the user to go through the authentication modules with a set authentication level.

Users can also access authentication modules with specific authentication level. For example, a user performs a login with the following syntax:

```
http://hostname:port/deploy_URI/UI/Login?authlevel=
auth_level_value
```

All modules whose authentication level is larger or equal to *auth_level_value* will be displayed as an authentication menu for the user to choose. If only one matching module is found, then the login page for that authentication module will be directly displayed.

This method of authentication allows an administrator to specify the security level of the modules to which identities can authenticate. Each authentication module has a separate Authentication Level attribute and the value of this attribute can be defined as any valid integer. With Authentication Level-based authentication, the Authentication Service displays a module login page with a menu containing the authentication modules that have authentication levels equal to or greater then the value specified in the Login URL parameter. Users can select a module from the presented list. Once the user selects a module, the remaining process is based on Module-based Authentication.

Authentication Level-based Authentication Login URLs

Authentication level-based authentication can be specified in the User Interface Login URL by defining the authlevel Parameter. After calling the login screen with the relevant list of modules, the user must choose one with which to authenticate. The login URLs used to specify and initiate authentication level-based authentication are:

```
http://server_name.domain_name:port/amserver/UI/Login?authlevel=authentication_level
```

and

```
http://server_name.domain_name:port/amserver/UI/
Login?realm=realm_name&authlevel=authentication_level
```

If there is no configured `realm` parameter, the realm to which the user belongs will be determined from the server host and domain specified in the login URL itself.

Authentication Level-based Authentication Redirection URLs

Upon a successful or failed authentication level-based authentication, Federated Access Manager looks for information on where to redirect the user. Following is the order of precedence in which the application will look for this information.

Successful Authentication Level-based Authentication Redirection URLs

The redirection URL for successful authentication level-based authentication is determined by checking the following places in order of precedence:

1. A URL set by the authentication module.
2. A URL set by a `goto` Login URL parameter.
3. A URL set in the `clientType` custom files for the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).

4. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's role entry.
5. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.
6. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute as a global default.
7. A URL set in the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).
8. A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's role entry.
9. A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.
10. A URL set in the `iplanet-am-auth-login-success-url` attribute as a global default.

Failed Authentication Level-based Authentication Redirection URLs

The redirection URL for failed authentication level-based authentication is determined by checking the following places in the following order:

1. A URL set by the authentication module.
2. A URL set by a `gotoOnFail` Login URL parameter.
3. A URL set in the `clientType` custom files for the `iplanet-am-user-failure-url` attribute of the user's entry (`amUser.xml`).
4. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.
5. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.
6. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute as a global default.
7. A URL set for the `iplanet-am-user-failure-url` attribute in the user's entry (`amUser.xml`).
8. A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.
9. A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.
10. A URL set for the `iplanet-am-auth-login-failure-url` attribute as the global default.

Module-based Authentication

Users can access a specific authentication module using the following syntax:

```
http://hostname:port/deploy_URI/UI/Login?module=
module_name
```

Before the authentication module can be accessed, the Core authentication service attribute realm Authentication Modules must be modified to include the authentication module name. If the authentication module name is not included in this attribute, the "authentication module denied" page will be displayed when the user attempts to authenticate.

This method of authentication allows a user to specify the module to which they will authenticate. The specified module must be registered to the realm or sub-realm that the user is accessing. This is configured in the realm Authentication Modules attribute of the realm's Core Authentication Service. On receiving this request for module-based authentication, the Authentication Service verifies that the module is correctly configured as noted, and if the module is not defined, the user is denied access.

Module-based Authentication Login URLs

Module-based authentication can be specified in the User Interface Login URL by defining a module Parameter. The login URLs used to specify and initiate module-based authentication are:

```
http://server_name.domain_name:port/amserver/UI/Login?module=authentication_module_name
http://server_name.domain_name:port/amserver/UI/
Login?org=org_name&module=authentication_module_name
```

If there is no configured `org` parameter, the realm to which the user belongs will be determined from the server host and domain specified in the login URL itself.

Module-based Authentication Redirection URLs

Upon a successful or failed module-based authentication, Federated Access Manager looks for information on where to redirect the user. Following is the order of precedence in which the application will look for this information.

Successful Module-based Authentication Redirection URLs

The redirection URL for successful module-based authentication is determined by checking the following places in order of precedence:

1. A URL set by the authentication module.
2. A URL set by a `goto` Login URL parameter.
3. A URL set in the `clientType` custom files for the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).

4. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's role entry.
5. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.
6. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute as a global default.
7. A URL set in the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).
8. A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's role entry.
9. A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.
10. A URL set in the `iplanet-am-auth-login-success-url` attribute as a global default.

Failed Module-based Authentication Redirection URLs

The redirection URL for failed module-based authentication is determined by checking the following places in the following order:

1. A URL set by the authentication module.
2. A URL set by a `gotoOnFail` Login URL parameter.
3. A URL set in the `clientType` custom files for the `iplanet-am-user-failure-url` attribute of the user's entry (`amUser.xml`).
4. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.
5. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.
6. A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute as a global default.
7. A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.
8. A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.
9. A URL set for the `iplanet-am-auth-login-failure-url` attribute as the global default.

The User Interface Login URL

The Authentication Service user interface is accessed by entering a login URL into the Location Bar of a web browser. This URL is:

`http://`*AccessManager-root/.domain_name:port* `/`*service_deploy_uri* `/UI/Login`

Note – During installation, the *service_deploy_uri* is configured as `amserver`. This default service deployment URI will be used throughout this document.

The user interface login URL can also be appended with Login URL Parameters to define specific authentication methods or successful/failed authentication redirection URLs.

Login URL Parameters

A URL parameter is a name/value pair appended to the end of a URL. The parameter starts with a question mark (`?`) and takes the form `name=value`. A number of parameters can be combined in one login URL, for example:

```
http://server_name.domain_name:port/amserver/UI/
Login?module=LDAP&locale=ja&goto=http://www.sun.com
```

If more than one parameter exists, they are separated by an ampersand (&). The combinations though must adhere to the following guidelines:

- Each parameter can occur only once in one URL. For example, `module=LDAP&module=NT` is not computable.
- Both the `org` parameter and the `domain` parameter determine the login realm. In this case, only one of the two parameters should be used in the login URL. If both are used and no precedence is specified, only one will take effect.
- The parameters `user`, `role`, `service`, `module` and `authlevel` are for defining authentication modules based on their respective criteria. Due to this, only one of them should be used in the login URL. If more than one is used and no precedence is specified, only one will take effect.

The following sections describe parameters that, when appended to the User Interface Login URL and typed in the Location bar of a web browser, achieve various authentication functionality.

Note – To simplify an authentication URL and parameters for distribution throughout an realm, an administrator might configure an HTML page with a simple URL that possesses links to the more complicated login URLs for all configured authentication methods.

---

goto Parameter

A `goto=`*successful_authentication_URL* parameter overrides the value defined in the Login Success URL of the Authentication Configuration service. It will link to the specified URL when a successful authentication has been achieved. A `goto=logout_URL` parameter can also be used to link to a specified URL when the user is logging out. For an example of a successful authentication URL:

```
http://server_name.domain_name:port/amserver/
UI/Login?goto=http://www.sun.com/homepage.html
```

An example `goto` logout URL:

```
http://server_name.domain_name:port/amserver/
UI/Logout?goto=http://www.sun.com/logout.html.
```

Note – There is an order of precedence in which Federated Access Manager looks for successful authentication redirection URLs. Because these redirection URLs and their order are based on the method of authentication, this order (and related information) is detailed in the Authentication Types section.

---

gotoOnFail Parameter

A `gotoOnFail=failed_authentication_URL` parameter overrides the value defined in the Login Failed URL of the Authentication Configuration service. It will link to the specified URL if a user has failed authentication. An example `gotoOnFail` URL might be `http://`*server_name.domain_name*`:`*port* `/amserver/UI/Login?gotoOnFail=http://www.sun.com/auth_fail.html`.

Note – There is an order of precedence in which Federated Access Manager looks for failed authentication redirection URLs. Because these redirection URLs and their order are based on the method of authentication, this order (and related information) is detailed in Authentication Types section.

---

realm Parameter

The `org=`*realmName* parameter allows a user to authenticate as a user in the specified realm.

Note – A user who is not already a member of the specified realm will receive an error message when they attempt to authenticate with the `realm` parameter. A user profile, though, can be dynamically created in the Directory Server if all of the following are TRUE:

- The User Profile attribute in the Core Authentication Service must be set to `Dynamic` or `Dynamic with User Alias.`
- The user must successfully authenticate to the required module.
- The user does not already have a profile in Directory Server.

From this parameter, the correct login page (based on the realm and its locale setting) will be displayed. If this parameter is not set, the default is the top-level realm. For example, an `org` URL might be:

```
http://server_name.domain_name:port/amserver/UI/Login?realm=sun
```

---

org Parameter

The `org=`*orgName* parameter allows a user to authenticate as a user in the specified organization.

Note – A user who is not already a member of the specified organization will receive an error message when they attempt to authenticate with the `org` parameter. A user profile, though, can be dynamically created in the Directory Server if all of the following are TRUE:

- The User Profile attribute in the Core Authentication Service must be set to `Dynamic` or `Dynamic with User Alias.`
- The user must successfully authenticate to the required module.
- The user does not already have a profile in Directory Server.

From this parameter, the correct login page (based on the organization and its locale setting) will be displayed. If this parameter is not set, the default is the top-level organization. For example, an `org` URL might be:

```
http://server_name.domain_name:port/amserver/UI/Login?org=sun
```

---

user Parameter

The `user=`*userName* parameter forces authentication based on the module configured in User Authentication Configuration attribute of the user's profile. For example, one user's profile can be configured to authenticate using the Certification module while another user might be configured to authenticate using the LDAP module. Adding this parameter sends the user to their configured authentication process rather than the method configured for their organization. For example:

```
http://server_name.domain_name:port/amserver/UI/Login?user=jsmith
```

role Parameter

A `role`=roleName parameter sends the user to the authentication process configured for the specified role. A user who is not already a member of the specified role will receive an error message when they attempt to authenticate with this parameter. For example:

```
http://server_name.domain_name:port/amserver/UI/Login?role=manager.
```

locale Parameter

Federated Access Manager has the capability to display localized screens (translated into languages other than English) for the authentication process as well as for the console itself. The `locale=`*localeName* parameter allows the specified locale to take precedence over any other defined locales. The login locale is displayed by the client after searching for the configuration in the following places, order-specific:

1. Value of locale parameter in Login URL

   The value of the `locale=`*localeName* parameter takes precedence over all other defined locales.
2. Locale defined in user's profile

   If there is no URL parameter, the locale is displayed based on the value set in the User Preferred Language attribute of the user profile.
3. Locale defined in the HTTP header

   This locale is set by the web browser.
4. Locale defined in Core Authentication Service

   This is the value of the Default Auth Locale attribute in the Core Authentication module.
5. Locale defined in Platform Service

   This is the value of the Platform Locale attribute in the Platform service.

Operating system locale

The locale derived from this pecking order is stored in the user's session token and Federated Access Manager uses it for loading the localized authentication module only. After successful authentication, the locale defined in the User Preferred Language attribute of the user's profile is used. If none is set, the locale used for authentication will be carried over. For example:

```
http://server_name.domain_name:port/amserver/UI/Login?locale=ja.
```

Note – Information on how to localize the screen text and error messages can be found in the Federated Access Manager.

---

module Parameter

The `module=`*moduleName* parameter allows authentication via the specified authentication module. Any of the modules can be specified although they must first be registered under the realm to which the user belongs and selected as one of that realm's authentication modules in the Core Authentication module. For example:

```
http://server_name.domain_name:port/amserver/UI/Login?module=Unix.
```

Note – The authentication module names are case-sensitive when used in a URL parameter.

---

service Parameter

The `service=`*serviceName* parameter allows a user to authenticate via a service's configured authentication scheme. Different authentication schemes can be configured for different services using the Authentication Configuration service. For example, an online paycheck application might require authentication using the more secure Certificate Authentication module while an realm's employee directory application might require only the LDAP Authentication module. An authentication scheme can be configured, and named, for each of these services. For example:

```
http://server_name.domain_name:port/amserver/UI/Login?service=sv1.
```

Note – The Authentication Configuration service is used to define a scheme for service-based authentication.

---

arg Parameter

The `arg=`*newsession* parameter is used to end a user's current session and begin a new one. The Authentication Service will destroy a user's existing session token and perform a new login in one request. This option is typically used in the Anonymous Authentication module. The user first authenticates with an anonymous session, and then hits the register or login link. For example:

```
http://server_name.domain_name:port/amserver/UI/Login?arg=newsession.
```

authlevel Parameter

An `authlevel=`*value* parameter tells the Authentication Service to call a module with an authentication level equal to or greater than the specified authentication level value. Each authentication module is defined with a fixed integer authentication level. For example:

```
http://server_name.domain_name:port/amserver/UI/Login?authlevel=1.
```

Note – The Authentication Level is set in each module's specific profile. .

---

domain Parameter

This parameter allows a user to login to an realm identified as the specified domain. The specified domain must match the value defined in the Domain Name attribute of the realm's profile. For example:

```
http://server_name.domain_name:port/amserver/UI/Login?domain=sun.com.
```

Note – A user who is not already a member of the specified domain/realm will receive an error message when they attempt to authenticate with the `org` parameter. A user profile, though, can be dynamically created in the Directory Server if all of the following points are TRUE:

- The User Profile attribute in the Core Authentication Service must be set to `Dynamic` or `Dynamic With User Alias .`
- The user must successfully authenticate to the required module.
- The user does not already have a profile in Directory Server.

---

iPSPCookie Parameter

The `iPSPCookie=yes` parameter allows a user to login with a persistent cookie. A persistent cookie is one that continues to exist after the browser window is closed. In order to use this parameter, the realm to which the user is logging in must have Persistent Cookies enabled in their Core Authentication module. Once the user authenticates and the browser is closed, the user can login with a new browser session and will be directed to console without having to re-authenticate. This will work until the value of the Persistent Cookie Max Time attribute specified in the Core Service elapses. For example:

```
http://server_name.domain_name:port/amserver/UI/Login?org=example&iPSPCookie=yes
```

IDTokenN Parameters

This parameter option to enables a user to pass authentication credentials using a URL or HTML forms. With the `IDTokenN=`*value* parameters, a user can be authenticated without accessing the Authentication Service User Interface. This process is called *Zero Page Login*. Zero page login works only for authentication modules that use one login page. The values of `IDToken0, IDToken1, ..., IDTokenN` map to the fields on the authentication module's login page. For example, the LDAP authentication module might use `IDToken1` for the `userID` information, and `IDToken2` for password information. In this case, the LDAP module IDTokenN URL would be:

```
http://server_name.domain_name:port/amserver/UI/
Login?module=LDAP&IDToken1=userID&IDToken2=password
```

(`module=LDAP` can be omitted if LDAP is the default authentication module.)

For Anonymous authentication, the login URL parameter would be:

```
http://server_name.domain_name:port/amserver/UI/Login?module=Anonymous&IDToken1=anonymousUserID.
```

Note – The token names `Login.Token0, Login.Token1, ..., Login.TokenN` (from previous releases) are still supported but will be deprecated in a future release. It is recommended to use the new `IDTokenN` parameters.

Account Locking

The Authentication Service provides a feature where a user will be *locked out* from authenticating after a defined number of failures. This feature is turned off by default, but can be enabled using the Federated Access Manager console.

Note – Only modules that throw an Invalid Password Exception can leverage the Account Locking feature.

The Core Authentication service contains attributes for enabling and customizing this feature including (but not limited to):

- **Login Failure Lockout Mode** which enables account locking.
- **Login Failure Lockout Count** which defines the number of tries that a user may attempt to authenticate before being locked out. This count is valid per user ID only; the same user ID needs to fail for the given count after which that user ID would be locked out.
- **Login Failure Lockout Interval** defines (in minutes) the amount of time in which the Login Failure Lockout Count value must be completed before a user is locked out.
- **Email Address to Send Lockout Notification** specifies an email address to which user lockout notifications will be sent.

- **Warn User After N Failure** specifies the number of authentication failures that can occur before a warning message will be displayed to the user. This allows an administrator to set additional login attempts after the user is warned about an impending lockout.
- **Login Failure Lockout Duration** defines (in minutes) how long the user will have to wait before attempting to authenticate again after lockout.
- **Lockout Attribute Name** defines which LDAP attribute in the user's profile will be set to `inactive` for Physical Locking.
- **Lockout Attribute Value** defines to what the LDAP attribute specified in **Lockout Attribute Name** will be set: `inactive` or `active`.

Email notifications are sent to administrators regarding any account lockouts. (Account locking activities are also logged.)

Note – For special instructions when using this feature on a Microsoft® Windows 2000 operating system, see "Simple Mail Transfer Protocol (SMTP)" in Appendix A, "AMConfig.properties File."

Federated Access Manager supports two types of account locking are supported: Physical Locking and Memory Locking, defined in the following sections.

Physical Locking

This is the default locking behavior for Federated Access Manager The locking is initiated by changing the status of a LDAP attribute in the user's profile to inactive. The `Lockout Attribute Name` attribute defines the LDAP attribute used for locking purposes.

Note – An aliased user is one that is mapped to an existing LDAP user profile by configuring the User Alias List Attribute (`iplanet-am-user-alias-list` in `amUser.xml`) in the LDAP profile. Aliased users can be verified by adding `iplanet-am-user-alias-list` to the Alias Search Attribute Name field in the Core Authentication Service. That said, if an aliased user is locked out, the actual LDAP profile to which the user is aliased will be locked. This pertains only to physical lockout with authentication modules other than LDAP and Membership.

Memory Locking

Memory locking is enabled by changing the `Login Failure Lockout Duration` attribute to a value greater then 0. The user's account is then locked in memory for the number of minutes specified. The account will be unlocked after the time period has passed. Following are some special considerations when using the memory locking feature:

- If Federated Access Manager is restarted, all accounts locked in memory are unlocked.
- If a user's account is locked in memory and the administrator changes the account locking mechanism to physical locking (by setting the lockout duration back to 0), the user's account will be unlocked in memory and the lock count reset.
- After memory lockout, when using authentication modules other than LDAP and Membership, if the user attempts to login with the correct password, a *User does not have profile in this realm error.* is returned rather than a *User is not active.* error.

Note – If the Failure URL attribute is set in the user's profile, neither the lockout warning message nor the message indicating that their account has been locked will not be displayed; the user will be redirected to the defined URL.

Authentication Service Failover

Authentication service failover automatically redirects an authentication request to a secondary server if the primary server fails because of a hardware or software problem or if the server is temporarily shut down.

An authentication context must first be created on an instance of Federated Access Manager where the authentication service is available. If this instance of Federated Access Manager is not available, an authentication context can then be created on a different instance of Federated Access Manager through the authentication failover mechanism. The authentication context will check for server availability in the following order:

1. The authentication service URL is passed to the AuthContext API. For example:

   ```
   AuthContext(orgName, url)
   ```

   If this API is used, it will only use the server referenced by the URL. No failover will occur even if the authentication service is available on that server.
2. The authentication context will check the server defined in the `com.iplanet.am.server*` attribute of the `AMConfig.properties` file.
3. If step 2 fails, then the authentication context queries the platform list from a server where the Naming service is available This platform list is automatically created when multiple instances of Federated Access Manager are installed (generally, for failover purposes) sharing a one instance of Directory Server.

   For example, if the platform list contains URLs for `Server1`, `Server2` and `Server3`, then the authentication context will loop through `Server1` , `Server2` and `Server3` until authentication succeeds on one of them.

The platform list may not always be obtained from the same server, as it depends on the availability of the Naming service. Furthermore, Naming service failover may occur first. Multiple Naming service URLs are specified in the `com.iplanet.am.naming.url` property (in `AMConfing.properties`). The first available Naming service URL will be used to identify the server, which will contain the list of servers (in its platform server list) on which authentication failover will occur.

Fully Qualified Domain Name Mapping

Fully Qualified Domain Name (FQDN) mapping enables the Authentication Service to take corrective action in the case where a user may have typed in an incorrect URL (such as specifying a partial host name or IP address to access protected resources). FQDN mapping is enabled by modifying the `com.sun.identity.server.fqdnMap` attribute in the `AMConfig.properties` file. The format for specifying this property is:

`com.sun.identity.server.fqdnMap[`*invalid-name* `]=`*valid-name*

The value *invalid-name* would be a possible invalid FQDN host name that may be typed by the user, and *valid-name* would be the actual host name to which the filter will redirect the user. Any number of mappings can be specified (as illustrated in Code Example 1-1) as long as they conform to the stated requirements. If this property is not set, the user would be sent to the default server name configured in the `com.iplanet.am.server.host=` *server_name* property also found in the `AMConfig.properties` file.

EXAMPLE 4–1 FQDN Mapping Attribute In `AMConfig.properties`

```
com.sun.identity.server.fqdnMap[isserver]=isserver.mydomain.com
com.sun.identity.server.fqdnMap[isserver.mydomain]=isserver.mydomain.com
com.sun.identity.server.fqdnMap[
             IP address]=isserver.mydomain.com
```

Possible Uses For FQDN Mapping

This property can be used for creating a mapping for more than one host name which may be the case if applications hosted on a server are accessible by more than one host name. This property can also be used to configure Federated Access Manager to not take corrective action for certain URLs. For example, if no redirect is required for users who access applications by using an IP address, this feature can be implemented by specifying a map entry such as:

`com.sun.identity.server.fqdnMap[`*IP address* `]=`*IP address*.

Note – If more than one mapping is defined, ensure that there are no overlapping values in the invalid FQDN name. Failing to do so may result in the application becoming inaccessible.

Persistent Cookie

A persistent cookie is one that continues to exist after the web browser is closed, allowing a user to login with a new browser session without having to re-authenticate. The name of the cookie is defined by the `com.iplanet.am.pcookie.name` property in `AMConfig.properties`; the default value is `DProPCookie`. The cookie value is a 3DES-encrypted string containing the userDN, realm name, authentication module name, maximum session time, idle time, and cache time.

▼ To Enable Persistent Cookies

1. **Turn on the `Persistent Cookie Mode` in the Core Authentication module.**

2. **Configure a time value for the `Persistent Cookie Maximum Time` attribute in the Core Authentication module.**

3. **Append the iPSPCookie Parameter with a value of `yes` to the User Interface Login URL.**

   Once the user authenticates using this URL, if the browser is closed, they can open a new browser window and will be redirected to the console without re-authenticating. This will work until the time defined in Step 2 elapses.

   Persistent Cookie Mode can be turned on using the Authentication SPI method:

   `AMLoginModule.setPersistentCookieOn()`.

Multi-LDAP Authentication Module Configuration In Legacy Mode

As a form of failover or to configure multiple values for an attribute when the Federated Access Manager console only provides one value field, an administrator can define multiple LDAP authentication module configurations under one realm. Although these additional configurations are not visible from the console, they work in conjunction with the primary configuration if an initial search for the requesting user's authorization is not found. For example, one realm can define a search through LDAP servers for authentication in two different domains or it can configure multiple user naming attributes in one domain. For the latter, which has only one text field in the console, if a user is not found using the primary search criteria, the LDAP module will then search using the second scope. Following are the steps to configure additional LDAP configurations.

▼ To Add An Additional LDAP Configuration

1 **Write an XML file including the complete set of attributes and new values needed for second (or third) LDAP authentication configuration.**

The available attributes can be referenced by viewing the `amAuthLDAP.xml` located in `etc/opt/SUNWam/config/xml`. This XML file created in this step though, unlike the `amAuthLDAP.xml`, is based on the structure of the `amadmin.dtd`. Any or all attributes can be defined for this file. Code Example 1-2 is an example of a sub-configuration file that includes values for all attributes available to the LDAP authentication configuration.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
  Copyright (c) 2002 Sun Microsystems, Inc. All rights reserved.
  Use is subject to license terms.
-->
<!DOCTYPE Requests
    PUBLIC "-//iPlanet//Sun ONE Access Manager 6.0 Admin CLI DTD//EN"
    "jar://com/iplanet/am/admin/cli/amAdmin.dtd"
>
<!--
  Before adding subConfiguration load the schema with
GlobalConfiguration defined and replace corresponding
 serviceName and subConfigID in this sample file OR load
 serviceConfigurationRequests.xml before loading this sample
-->
<Requests>
<realmRequests DN="dc=iplanet,dc=com">
    <AddSubConfiguration subConfigName = "ssc"
        subConfigId = "serverconfig"
        priority = "0" serviceName="iPlanetAMAuthLDAPService">

              <AttributeValuePair>
            <Attribute name="iplanet-am-auth-ldap-server"/>
            <Value>vbrao.red.iplanet.com:389</Value>
        </AttributeValuePair>
        <AttributeValuePair>
            <Attribute name="iplanet-am-auth-ldap-base-dn"/>
            <Value>dc=iplanet,dc=com</Value>
        </AttributeValuePair>
        <AttributeValuePair>
            <Attribute name="planet-am-auth-ldap-bind-dn"/>
            <Value>cn=amldapuser,ou=DSAME Users,dc=iplanet,dc=com</Value>
        </AttributeValuePair>
        <AttributeValuePair>
            <Attribute name="iplanet-am-auth-ldap-bind-passwd"/>
            <Value>
                  plain text password</Value>
        </AttributeValuePair>
        <AttributeValuePair>
            <Attribute name="iplanet-am-auth-ldap-user-naming-attribute"/>
            <Value>uid</Value>
        </AttributeValuePair>
        <AttributeValuePair>
            <Attribute name="iplanet-am-auth-ldap-user-search-attributes"/>
            <Value>uid</Value>
        </AttributeValuePair>
        <AttributeValuePair>
            <Attribute name="iplanet-am-auth-ldap-search-scope"/>
            <Value>SUBTREE</Value>
        </AttributeValuePair>
        <AttributeValuePair>
            <Attribute name="iplanet-am-auth-ldap-ssl-enabled"/>
            <Value>false</Value>
        </AttributeValuePair>
        <AttributeValuePair>
            <Attribute name="iplanet-am-auth-ldap-return-user-dn"/>
            <Value>true</Value>
        </AttributeValuePair>
        <AttributeValuePair>
            <Attribute name="iplanet-am-auth-ldap-auth-level"/>
            <Value>0</Value>
        </AttributeValuePair>
        <AttributeValuePair>
            <Attribute name="iplanet-am-auth-ldap-server-check"/>
            <Value>15</Value>
        </AttributeValuePair>

    </AddSubConfiguration>

</realmRequests>
</Requests>
```

2 **Copy the plain text password as the value for the iplanet-am-auth-ldap-bind-passwd in the XML file created in Step 1.**

The value of this attribute is formatted in bold in the code example.

3 **Load the XML file using the `amadmin` command line tool.**

```
./amadmin -u amadmin -w administrator_password -v -t name_of_XML_file.
```

Note that this second LDAP configuration can not be seen or modified using the console.

Tip – There is a sample available for multi-LDAP configuration. See the `serviceAddMultipleLDAPConfigurationRequests.xml` command line template in `/FederatedAccessManager-base/SUNWam/samples/admin/cli/bulk-ops/`. Instructions can be found in `Readme.html` at `/AccesManager-base/SUNWam/samples/admin/cli/`.

Session Upgrade

The Authentication service enables you to upgrade a valid session token based on a second, successful authentication performed by the same user to one realm. If a user with a valid session token attempts to authenticate to a resource secured by his current realm and this second authentication request is successful, the session is updated with the new properties based on the new authentication. If the authentication fails, the user's current session is returned without an upgrade. If the user with a valid session attempts to authenticate to a resource secured by a different realm, the user will receive a message asking whether they would like to authenticate to the new realm. The user can, at this point, maintain the current session or attempt to authenticate to the new realm. Successful authentication will result in the old session being destroyed and a new one being created.

During session upgrade, if a login page times out, redirection to the original success URL will occur. Timeout values are determined based on:

- The page timeout value set for each module (default is 1 minute)
- `com.iplanet.am.invalidMaxSessionTime` property in `AMConfig.properties` (default is 10 minutes)
- `iplanet-am-max-session-time` (default is 120 minutes)

The values of `com.iplanet.am.invalidMaxSessionTimeout` and `iplanet-am-max-session-time` should be greater than the page timeout value, or the valid session information during session upgrade will be lost and URL redirection to the previous successful URL will fail.

Validation Plug-in Interface

An administrator can write username or password validation logic suitable to their realm, and plug this into the Authentication Service. (This functionality is supported only by the LDAP and Membership authentication modules.) Before authenticating the user or changing the password, Federated Access Manager will invoke this plug-in. If the validation is successful, authentication continues; if it fails, an authentication failed page will be thrown. The plug-in extends the `com.iplanet.am.sdk.AMUserPasswordValidation` class which is part of the Service Management SDK. Information on this SDK can be found in the `com.iplanet.am.sdk` package in the Federated Access Manager Javadoc.

▼ To Write and Configure a Validation Plug-in

1. **The new plug-in class will extend the** `com.iplanet.am.sdk. AMUserPasswordValidation` **class and implement the** `validateUserID()` **and** `validatePassword()` **methods.** `AMException` **should be thrown if validation fails.**

2. **Compile the plug-in class and place the** `.class` **file in the desired location. Update the classpath so that it is accessible by the Federated Access Manager during runtime.**

3. **Login to the Federated Access Manager console as top-level administrator. Click on the Configuration tab, and go to the attributes for the Administration Service. Type the name of the plug-in class (including the package name) in the** `UserID & Password Validation Plugin Class` **field.**

4. **Logout and login.**

JAAS Shared State

The JAAS shared state provides sharing of both user ID and password between authentication modules. Options are defined for each authentication module for:

- Realm (or, Organization)
- User
- Service
- Role

Upon failure, the module prompts for its required credentials. After failed authentication, the module stops running, or the logout shared state clears.

Enabling JAAS Shared State

To configure the JAAS shared state:

- Use the `iplanet-am-auth-shared-state-enabled` option.
- The usage for the shared state option is:`iplanet-am-auth-shared-state-enabled=true`
- The default for this option is true.
- This variable is specified in the Options column of the authentication chaining configuration.

Upon failure, the authentication module will prompt for the required credentials as per the `tryFirstPass` option behavior suggested in the JAAS specification.

JAAS Shared State Store Option

To configure the JAAS shared state store option:

- Use the `iplanet-amauth-store-shared-state-enabled` option.
- The usage for the store shared state option is:`iplanet-am-auth-store-shared-state-enabled=true`
- The default for this option is false.
- This variable is specified in the Options column of the authentication chaining configuration.

After a commit, an abort or a logout, the shared state will be cleared.

CHAPTER 5

Managing Policies

This chapter describes the Policy Management feature of Sun Java^{TM} System Federated Access Manager. Federated Access Manager's Policy Management feature enables the top-level administrator or top-level policy administrator to view, create, delete and modify policies for a specific service that can be used across all realms. It also provides a way for a realm or sub realm administrator or policy administrator to view, create, delete and modify policies at the realm level.

This chapter contains the following sections:

Overview

A *policy* defines rules that specify access privileges to an organization's protected resources. Businesses posses resources, applications and services that they need to protect, manage and monitor. Policies control the access permissions and usage of these resources by defining when and how a user can perform an action on a given resource. A policy defines the resources for a particular principal.

Note – A *principal* can be an individual, a corporation, a role, or a group; anything that can have an identity. for more information, see the Java^{TM} 2 Platform Standard Edition Javadoc (`http://java.sun.com/j2se/1.4.2/docs/api/java/security/Principal.html`).

A single policy can define either binary or non-binary decisions. A binary decision is *yes*/*no*, *true*/ *false* or *allow*/*deny*. A non-binary decision represents the value of an attribute. For example, a mail service might include a `mailboxQuota` attribute with a maximum storage value set for each user. In general, a policy is configured to define what a principal can do to which resource and under what conditions.

Policy Management Feature

The Policy Management feature provides a *policy service* for creating and managing policies. The policy service allows administrators to define, modify, grant, revoke and delete permissions to protect resources within the Federated Access Manager deployment. Typically, a policy service includes a data store, a library of interfaces that allows for the creation, administration and evaluation of policies, and a policy enforcer or *policy agent*. By default, Federated Access Manager uses Sun Java Enterprise System Directory Server for data storage, and provides Java and C APIs for policy evaluation and policy service customization (see the *Sun Java System Access Manager 7.1 Developer's Guide* for more information). It also allows administrator to use the Federated Access Manager console for policy management. Federated Access Manager provides one policy—enabled service, the URL Policy Agent service, which uses down-loadable policy agents to enforce the policies.

URL Policy Agent Service

Upon installation, Federated Access Manager provides the URL Policy Agent service to define policies to protect HTTP URLs. This service allows administrators to create and manage policies through a policy enforcer or *policy agent.*

Policy Agents

The Policy Agent is the Policy Enforcement Point (PEP) for a server on which an enterprise's resources are stored. The policy agent is installed separately from Federated Access Manager onto a web server and serves as an additional authorization step when a user sends a request for a web resource that exists on the protected web server. This authorization is in addition to any user authorization request which the resource performs. The agent protects the web server, and in turn, the resource is protected by the authorization plug-in.

For example, a Human Resources web server protected by a remotely-installed Federated Access Manager might have an agent installed on it. This agent would prevent personnel without the proper policy from viewing confidential salary information or other sensitive data. The policies are defined by the Federated Access Manager administrator, stored within the Federated Access Manager deployment and used by the policy agent to allow or deny users access to the remote web server's content.

The most current Federated Access Manager Policy Agents can be downloaded from the Sun Microsystems Download Center.

More information on installing and administrating the policy agents can be found in the *Sun Java System Access Manager Policy Agent 2.2 User's Guide*.

Note – Policy is evaluated in no particular order although as they are evaluated, if one action value evaluates to *deny*, subsequent policies are not evaluated, unless the Continue Evaluation On Deny Decision attribute is enabled in the Policy Configuration service.

Federated Access Manager Policy agents enforce decisions only on web URLs (`http://...`, or `https//...`). However, agents can be written using the Java and C Policy Evaluation APIs to enforce policy on other resources.

In addition, the Resource Comparator attribute in the Policy Configuration Service would also need to be changed from its default configuration to:

`serviceType=`*Name_of_LDAPService*
`|class=com.sun.identity.policy.plugins.SuffixResourceName|wildcard=*`

`|delimiter=,|caseSensitive=false`

Alternately, providing an implementation such as `LDAPResourceName` to implement `com.sun.identity.policy.interfaces.ResourceName` and configuring the Resource Comparator appropriately would also work.

The Policy Agent Process

The process for protected web resources begins when a web browser requests a URL that resides on a server protected by the policy agent. The server's installed policy agent intercepts the request and checks for existing authentication credentials (a session token).

If the agent has intercepted a request and validated the existing session token, the following process is followed.

1. If the session token is valid, the user is allowed or denied access. If the token is invalid, the user is redirected to the Authentication Service, as outlined in the following steps.

   Assuming the agent has intercepted a request for which there is no existing session token, the agent redirects the user to the login page even if the resource is protected using a different authentication method.
2. Once the user's credentials are properly authenticated, the agent issues a request to the Naming Service which defines the URLs used to connect to Federated Access Manager's internal services.
3. If the resource matches the non-enforced list, configured at the agent, access is allowed.
4. The Naming Service returns locators for the policy service, session service and logging service.
5. The agent sends a request to the Policy Service to get policy decisions applicable to the user.

6. Based on the policy decisions for the resource being accessed, the user is either allowed or denied access. If advice on the policy decision indicates a different authentication level or authentication mechanism, the agent redirects the request to the Authentication Service until all criteria is validated.

Policy Types

There are two types of policies that can be configured using Federated Access Manager:

- "Normal Policy" on page 102
- "Referral Policy" on page 107

Normal Policy

In Federated Access Manager, a policy that defines access permissions is referred to as a *normal* policy. A normal policy consists of *rules* , *subjects*, *conditions*, and *response providers*.

Rules

A *rule* contains a service type, one or more actions, and a value. The rule, basically, defines the policy.

- A service type defines the type of resource that is being protected.
- An *action* is the name of an operation that can be performed on the resource; examples of web server actions are POST or GET. An allowable action for a human resources service might be to be able to change a home telephone number.
- A *value* defines the permission for the action, for example, allow or deny.

Note – It is acceptable to define an action without resources for some services.

Subjects

A *subject* defines the user or collection of users (for instance, a group or those who possess a specific role) that the policy affects. The general rule for subjects is that the policy would apply only if the user is a member of at least one subject in the policy. The default subjects are:

Federated Access Manager Identity Subject	This subject implies that the identities you create and manage under the Realms Subject tab can be added as a member of the subject.
Authenticated Users	This subject type implies that any user with a valid SSOToken is a member of this subject.

	All authenticated users would be member of this Subject, even if they have authenticated to a realm that is different from the organization in which the policy is defined. This is useful if the resource owner would like to give access to resources that is managed for users from other organizations. If you want to restrict access to resources being protected to members of a specific organization, please use the Organization subject.
Web Services Clients	his subject type implies that a web service client (WSC) identified by the SSOToken is a member of this subject, if the DN of any principal contained in the SSOToken matches any selected value of this subject. Valid values are the DNs of trusted certificates in the local JKS keystore, which correspond to the certificates of trusted WSCs. This subject has dependency on the Liberty Web Services Framework and should be used only by Liberty Service Providers to authorize WSCs. Make sure that you have created the keystore before you add this Subject to a policy. Information on setting up the keystore can be found in the following location: *FederatedAccessManager-base* `/SUNWam/samples/saml/xmlsig/keytool.html`

The following additional subjects are available by selecting them in the Policy Configuration Service of the realm:

Federated Access Manager Roles	This subject type implies that any member of an Federated Access Manager role is a member of this subject. An Federated Access Manager role is created using Federated Access Manager running in legacy mode and using the 6.3–based console. These roles have object classes mandated by Federated Access Manager. Federated Access Manager roles can only be accessed through the hosting Federated Access Manager Policy Service.
LDAP Groups	This subject type implies that any member of an LDAP group is member of this subject.

LDAP Roles	This subject type implies that any member of an LDAP role is a member of this subject. An LDAP Role is any role definition that uses the Directory Server role capability. These roles have object classes mandated by Directory Server role definition. The LDAP Role Search filter can be modified in the Policy Configuration Service to narrow the scope and improve performance.
LDAP Users	This subject type implies that any LDAP user is a member of this subject.
Organization	This subject type implies that any member of a realm is a member of this subject

Federated Access Manager Roles Versus LDAP Roles

An Federated Access Manager role is created using Federated Access Manager These roles have object classes mandated by Federated Access Manager. An LDAP role is any role definition that uses the Directory Server role capability. These roles have object classes mandated by Directory Server role definition. All Federated Access Manager roles can be used as Directory Server roles. However, all Directory Server roles are not necessarily Federated Access Manager roles. LDAP roles can be leveraged from an existing directory by configuring the "Policy Configuration Service" on page 127. Federated Access Manager roles can only be accessed through the hosting Federated Access Manager Policy Service. The LDAP Role Search filter can be modified in the Policy Configuration Service to narrow the scope and improve performance.

Nested Roles

Nested roles can be evaluated correctly as LDAP Roles in the subject of a policy definition.

Conditions

A condition allows you to define constraints on the policy. For example, if you are defining policy for a paycheck application, you can define a condition on this action limiting access to the application only during specific hours. Or, you may wish to define a condition that only grants this action if the request originates from a given set of IP addresses or from a company intranet.

The condition might additionally be used to configure different policies on different URIs on the same domain. For example, `http://org.example.com/hr/*jsp` can only be accessed by `org.example.net` from 9 a.m. to 5 p.m. This can be achieved by using an IP Condition along with a Time Condition. And specifying the rule resource as `http://org.example.com/hr/*.jsp`, the policy would apply to all the JSPs under `http://org.example.com/hr` including those in the sub directories.

Note – The terms referral, rule, resource, subject, condition, action and value correspond to the elements *Referral*, *Rule*, *ResourceName*, *Subject*, *Condition*, *Attribute* and *Value* in the `policy.dtd`.

The default conditions you can add are:

Active Session Time

Sets the condition based on user session data. The fields you can modify are:

Max Session Time	Specifies the maximum duration to which the policy is applicable starting from when the session was initiated.
Terminate Session	If selected, the user session will be terminated if the session time exceeds the maximum allowed as defined in the Max Session Time field.

Authentication Chain

The policy applies if the user has successfully authenticated to the authentication chain in the specified realm. If the realm is not specified, authentication to any realm at the authentication chain will satisfy the condition.

Authentication Level (greater than or equal to)

The policy applies if the user's authentication level is greater than or equal to the Authentication level set in the condition. This attribute indicates the level of trust for authentication within the specified realm.

Authentication Level (less than or equal to)

The policy applies if the user's authentication level is less than or equal to the Authentication level set in the condition. This attribute indicates the level of trust for authentication within the specified realm.

Authentication Module Instance

The policy applies if the user has successfully authenticated to the authentication module in the specified realm. If the realm is not specified, authentication to any realm at the authentication module will satisfy the condition.

Current Session Properties

Decides whether a policy is applicable to the request based on values of properties set in the user's Federated Access Manager session. During policy evaluation, the condition returns true only if the user's session has every property value defined in the condition. For properties defined with multiple values in the condition, it is sufficient if the token has at least one value listed for the property in the condition.

IP Address/DNS Name

Sets the condition based on a range of IP Addresses. The fields you can define are:

IP Address From/To	Specifies the range of the IP address.
DNS Name	Specifies the DNS name. This field can be a fully qualified hostname or a string in one of the following formats: *domainname* **.domainname*

LDAP Filter Condition

The policy is applicable when the defined LDAP filter locates the user entry in the LDAP directory that was specified in the Policy Configuration service. This is only applicable within the realm the policy is defined.

Realm Authentication

The policy applies if the user has authenticated to the specified realm.

Time (day, date, time, and timezone)

Sets the condition based on time constraints. The fields are:

Date From/To	Specifies the range of the date.
Time	Specifies the range of time within a day.
Day	Specifies a range of days.
Timezone	Specifies a timezone, either standard or custom. Custom timezones can only be a timezone ID recognized by Java (for example, PST). If no value is specified, the default value is the Timezone set in the Federated Access Manager JVM.

Response Providers

Response providers are plug-ins that provide policy-based response attributes. The response provider attributes are sent with policy decisions to the PEP. Federated Access Manager includes one implementation, the `IDResponseProvider.` Agents, PEPs, typically pass these response attributes as headers to applications. Applications typically use these attributes to personalize application pages such as a portal page.

Policy Advices

If a policy is not applicable as determined by the condition, the condition can produce advice messages that indicates why the policy was not applicable to the request. These advice messages are propagated in the policy decision to the Policy Enforcement Point. The Policy Enforcement Point can retrieve this advice and try to take the appropriate action, such as redirecting the user back to the authentication mechanism to authenticate to a higher level. The user may then be prompted for higher level authentication and may be able to access to the resource, if the policy becomes applicable, after proper action for the advice is taken.

More information can be found in the following class:

```
com.sun.identity.policy.ConditionDecision.getAdvices()
```

Only `AuthLevelCondiiton` and `AuthSchemeCondition` provide advices if the condition is not satisfied.

`AuthLevelCondition` advice is associated with the following key:

```
com.sun.identity.policy.plugin.AuthLevelCondition.AUTH_LEVEL_CONDITION_ADVICE
```

`AuthSchemeCondition` advice is associated with the following key:

```
com.sun.identity.policy.plugin.AuthLevelCondition.AUTH_SCHEME_CONDITION_ADVICE
```

Custom conditions can also produce advices. However, the Federated Access Manager Policy Agents respond only for Auth Level Advice and Auth Scheme Advice. Custom agents could be written to understand and respond to more advices and existing Federated Access Manager agents can be extended to understand and respond to more advices. For more information, see the *Sun Java System Access Manager Policy Agent 2.2 User's Guide*.

Referral Policy

An administrator may need to delegate one realm's policy definitions and decisions to another realm. (Alternatively, policy decisions for a resource can be delegated to other policy products.) A *referral* policy controls this policy delegation for both policy creation and evaluation. It consists of one or more *rules* and one or more *referrals*.

The Policy Configuration service contains a global attribute called Organization Alias Referrals This attribute allows you to create policies in sub-realms without having to create referral policies from the top-level or parent realm. You can only create policies to protect HTTP or HTTPS resources whose fully qualified hostname matches the realm/DNS Alias of the realm. By default, this attribute is defined as No.

Rules

A rule defines the resource whose policy definition and evaluation is being referred.

Referrals

The referral defines the organization to which the policy evaluation is being referred. By default, there are two types of referrals: peer realm and sub realm. They delegate to an realm on the same level and an realm on a sub level, respectively. See "Creating Policies for Peer Realms and Sub Realms" on page 119 for more information.

Note – The realm that is referred to can define or evaluate policies only for those resources (or sub-resources) that have been referred to it. This restriction, however, does not apply to the top-level realm.

Policy Definition Type Document

Once a policy is created and configured, it is stored in Directory Server in XML. In Directory Server, the XML-encoded data is stored in one place. Although policy is defined and configured using the `amAdmin.dtd` (or the console), it is actually stored in Directory Server as XML that is based on the `policy.dtd` . The `policy.dtd` contains the policy element tags extracted from the `amAdmin.dtd` (without the policy creation tags). So, when the Policy Service loads policies from Directory Server, it parses the XML based on the `policy.dtd`. The `amAdmin.dtd` is only used when creating policy with the command line. This section describes the structure of `policy.dtd`. The `policy.dtd` exists in the following location:

```
FederatedAccessManager-base/SUNWam/dtd (Solairs)
FederatedAccessManager-base/identity/dtd (Linux)
FederatedAccessManager-base/identity/dtd (HP-UX)
FederatedAccessManager-base\identity\dtd (Windows)
```

Note – Throughout the rest of this chapter, only the Solaris directory information will be given. Please note that the directory structure for Linux,HP-UX and Windows is different.

Policy Element

Policy is the root element that defines the permissions or *rules* of a policy and to whom/what the rule applies or the *subject*. It also defines whether or not the policy is a *referral* (delegated) policy and whether there are any restrictions (or *conditions*) to the policy. It may contain one or more of the following sub-elements: *Rule*, *Conditions*, *Subjects*,*Referrals*, or *response providers*. The required XML attribute is `name` which specifies the name of the policy. The `referralPolicy` attribute identifies whether or not the policy is a referral policy; it defaults to a normal policy if not defined. Optional XML attributes include *name* and *description*.

Note – When tagging a policy as *referral*, subjects and conditions are ignored during policy evaluation. Conversely, when tagging a policy as *normal*, any Referrals are ignored during policy evaluation.

Rule Element

The *Rule* element defines the specifics of the policy and can take three sub-elements: *ServiceName*, *ResourceName* , or *AttributeValuePair*. It defines the type of service or application for which the policy has been created as well as the resource name and the actions which are performed on it. A rule can be defined without any actions; for example, a referral policy rule doesn't have any actions.

Note – It is acceptable to have a defined policy that does not include a defined *ResourceName* element.

ServiceName Element

The *ServiceName* element defines the name of the service to which the policy applies. This element represents the service type. It contains no other elements. The value is exactly as that defined in the service's XML file (based on the `sms.dtd`). The XML service attribute for the *ServiceName* element is the name of the service (which takes a string value).

ResourceName Element

The *ResourceName* element defines the object that will be acted upon. The policy has been specifically configured to protect this object. It contains no other elements. The XML service attribute for the *ResourceName* element is the name of the object. Examples of a *ResourceName* might be `http://www.sunone.com:8080/images` on a web server or `ldap://sunone.com:389/dc=example,dc=com` on a directory server. A more specific resource might be `salary://uid=jsmith,ou=people,dc=example,dc=com` where the object being acted upon is the salary information of John Smith.

AttributeValuePair Element

The *AttributeValuePair* element defines an action and its values. It is used as a sub-element to "Subject Element" on page 111, "Referral Element" on page 111 and "Condition Element" on page 112. It contains both the *Attribute* and *Value* elements and no XML service attributes.

Attribute Element

The *Attribute* element defines the name of the action. An action is an operation or event that is performed on a resource. POST or GET are actions performed on web server resources, READ or SEARCH are actions performed on directory server resources. The *Attribute* element must be paired with a *Value* element. The *Attribute* element itself contains no other elements. The XML service attribute for the *Attribute* element is the name of the action.

Value Element

The *Value* element defines the action values. Allow/deny or yes/no are examples of action values. Other action values can be either boolean, numeric, or strings. The values are defined in the service's XML file (based on the `sms.dtd`). The *Value* element contains no other elements and it contains no XML service attributes.

Note – Deny rules always take precedence over allow rules. For example, if one policy denies access and another allows it, the result is a deny (provided all other conditions for both policies are met). It is recommended that deny policies be used with extreme caution as they can lead to potential conflicts. If explicit deny rules are used, policies assigned to a user through different subjects (such as role and/or group membership) may result in denied access. Typically, the policy definition process should only use allow rules. The default deny may be used when no other policies apply.

Subjects Element

The *Subjects* sub-element identifies a collection of principals to which the policy applies; this collection is chosen based on membership in a group, ownership of a role or individual users. It takes the *Subject* sub-element. The XML attributes that can be defined are:

name. This defines a name for the collection.

description. This defines a description of the subject

includeType. This is not currently used.

Subject Element

The *Subject* sub-element identifies a collection of principals to which the policy applies; this collection pinpoints more specific objects from the collection defined by the Subjects element. Membership can be based on roles, group membership or simply a listing of individual users. It contains a sub-element, the "AttributeValuePair Element" on page 110. The required XML attribute is `type`, which identifies a generic collection of objects from which the specifically defined subjects are taken. Other XML attributes include `name` which defines a name for the collection and `includeType` which defines whether the collection is as defined, or whether the policy applies to users who are NOT members of the subject.

Note – When multiple subjects are defined, at least one of the subjects should apply to the user for the policy to apply. When a subject is defined with `includeType` set to false, the user should not be a member of that subject for the policy to apply.

Referrals Element

The *Referrals* sub-element identifies a collection of policy referrals. It takes the *Referral* sub-element. The XML attributes it can be defined with are `name` which defines a name for the collection and `description` which takes a description.

Referral Element

The *Referral* sub-element identifies a specific policy referral. It takes as a sub-element the "AttributeValuePair Element" on page 110. It's required XML attribute is `type` which identifies a generic collection of assignments from which the specifically defined referrals are taken. It can also include the `name` attribute which defines a name for the collection.

Conditions Element

The *Conditions* sub-element identifies a collection of policy restrictions (time range, authentication level, and so forth). It must contain one or more of the *Condition* sub-element. The XML attributes it can be defined with are `name` which defines a name for the collection and `description` which takes a description.

Note – The conditions element is an optional element in a policy.

Condition Element

The *Condition* sub-element identifies a specific policy restriction (time range, authentication level, and sor forth). It takes as a sub-element the “AttributeValuePair Element” on page 110. Its required XML attribute is `type` which identifies a generic collection of restrictions from which the specifically defined conditions are taken. It can also include the `name` attribute which defines a name for the collection.

Adding a Policy Enabled Service

You can define policies for resources of a given service only if the service schema has the `<Policy>` element configures to `sms.dtd`.

By default, Federated Access Manager provides the URL Policy Agent service (`iPlanetAMWebAgentService`). This service is defined in an XML file located in the following directory:

```
/etc/opt/SUNWam/config/xml/
```

You can, however add additional policy services to Federated Access Manager. Once the policy service is created, you add it to Federated Access Manager through the `amadmin` command line utility.

▼ To Add a New Policy Enabled Service

1 **Develop the new policy service in an XML file based on the `sms.dtd`. Federated Access Manager provides two policy service XML files that you may wish to use as the basis for the new policy service file:**

`amWebAgent.xml` - This the XML file for the default URL Policy Agent service. It is located in `/etc/opt/SUNWam/config/xml/`.

`SampleWebService.xml` .- This is the sample policy service file located `inFederatedAccessManager-base/samples/policy`.

2 **Save the XML file to the directory from which you will load the new policy service. For example:**

```
/config/xml/newPolicyService.xml
```

3 **Load the new policy service with the `amadmin` command line utility. For example:**

```
FederatedAccessManager-base/SUNWam/bin/amadmin
    --runasdn “uid=amAdmin,ou=People,default_org,
root_suffix
```

```
--password password
--schema /config/xml/newPolicyService.xml
```

4 **After you load the new policy service, you can define rules for the policy definitions through the Federated Access Manager console or by loading a new policy through `amadmin`.**

Creating Policies

You can create, modify and delete policies through the Policy API and the Federated Access Manager console, and create and delete policies through the `amadmin` command line tool. You can also get and list policies in XML using the `amadmin` utility. This section focuses on creating policies through the `amadmin` command line utility and through the Federated Access Manager console. For more information on the Policy APIs, see the *Sun Java System Access Manager 7.1 Developer's Guide*.

Policies are generally created using an XML file and added to Federated Access Manager through the `amadmin` command line utility and then managed using the Federated Access Manager console (although policies can be created using the console). This is because policies cannot be modified using `amadmin` directly. To modify a policy, you must first delete the policy from Federated Access Manager and then add the modified policy using `amadmin`.

In general, policy is created at the realm (or sub realm) level to be used throughout the realm's tree.

▼ To Create Policies with amadmin

1 **Create the policy XML file based on the `amadmin.dtd`. This file is located in the following directory:**

FederatedAccessManager-base `/SUNWam/dtd.`

The following is an example of a policy XML file. This example contains all of the default subject and condition values. For definitions of these values, see "Policy Types" on page 102.

```
<Policy name="bigpolicy" referralPolicy="false" active="true" >
<Rule name="rule1">
<ServiceName name="iPlanetAMWebAgentService" />
<ResourceName name="http://thehost.thedomain.com:80/*.html" />
<AttributeValuePair>
<Attribute name="POST" />
<Value>allow</Value>
</AttributeValuePair>
<AttributeValuePair>
```

```
<Attribute name="GET" />
<Value>allow</Value>
</AttributeValuePair>
</Rule>
<Subjects name="subjects" description="desccription">
<Subject name="webservicescleint" type="WebServicesClients" includeType="inclusive">
<AttributeValuePair><Attribute name="Values"/><Value>CN=sun-unix,
OU=SUN  Federated Access Manager, O=Sun, C=US</Value>
</AttributeValuePair>
</Subject>
<Subject name="amrole" type="IdentityServerRoles" includeType="inclusive">
<AttributeValuePair><Attribute name="Values"/><Value>
cn=organization admin role,o=realm1,dc=red,dc=iplanet,dc=com</Value>
</AttributeValuePair>
</Subject>
<Subject name="au" type="AuthenticatedUsers" includeType="inclusive">
</Subject>
<Subject name="ldaporganization" type="Organization" includeType="inclusive">
<AttributeValuePair><Attribute name="Values"/>
<Value>dc=red,dc=iplanet,dc=com</Value>
</AttributeValuePair>
</Subject>
<Subject name="ldapuser" type="LDAPUsers" includeType="inclusive">
<AttributeValuePair><Attribute name="Values"/>
<Value>uid=amAdmin,ou=People,dc=red,dc=iplanet,dc=com</Value>
</AttributeValuePair>
</Subject>
<Subject name="ldaprole" type="LDAPRoles" includeType="inclusive">
<AttributeValuePair><Attribute name="Values"/>
<Value>cn=Organization Admin Role,o=realm1,dc=red,dc=iplanet,dc=com</Value>
</AttributeValuePair>
</Subject>
<Subject name="ldapgroup" type="LDAPGroups" includeType="inclusive">
<AttributeValuePair><Attribute name="Values"/>
<Value>cn=g1,ou=Groups,dc=red,dc=iplanet,dc=com</Value>
</AttributeValuePair>
</Subject>
<Subject name="amidentitysubject" type="AMIdentitySubject" includeType="inclusive">
<AttributeValuePair><Attribute name="Values"/>
<Value>id=amAdmin,ou=user,dc=red,dc=iplanet,dc=com</Value>
</AttributeValuePair>
</Subject>
</Subjects>
```

```
<Conditions name="conditions" description="description">
<Condition name="ldapfilter" type="LDAPFilterCondition">
<AttributeValuePair><Attribute name="ldapFilter"/>
<Value>dept=finance</Value>
</AttributeValuePair>
</Condition>
<Condition name="authlevelge-nonrealmqualified" type="AuthLevelCondition">
<AttributeValuePair><Attribute name="AuthLevel"/>
<Value>1</Value>
</AttributeValuePair>
</Condition>
<Condition name="authlevelle-realmqaulfied" type="LEAuthLevelCondition">
<AttributeValuePair><Attribute name="AuthLevel"/>
<Value>/:2</Value>
</AttributeValuePair>
</Condition>
<Condition name="sessionproperties" type="SessionPropertyCondition">
<AttributeValuePair><Attribute name="valueCaseInsensitive"/>
<Value>true</Value>
</AttributeValuePair>
<AttributeValuePair><Attribute name="a"/><Value>10</Value>
<Value>20</Value>
</AttributeValuePair>
<AttributeValuePair><Attribute name="b"/><Value>15</Value>
<Value>25</Value>
</AttributeValuePair>
</Condition>
<Condition name="activesessiontime" type="SessionCondition">
<AttributeValuePair><Attribute name="TerminateSession"/>
<Value>session_condition_false_value</Value>
</AttributeValuePair>
<AttributeValuePair><Attribute name="MaxSessionTime"/>
<Value>30</Value>
</AttributeValuePair>
</Condition>
<Condition name="authelevelle-nonrealmqualfied"
        type="LEAuthLevelCondition">
<AttributeValuePair><Attribute name="AuthLevel"/>
<Value>2</Value>
</AttributeValuePair>
</Condition>
<Condition name="ipcondition" type="IPCondition">
<AttributeValuePair><Attribute name="DnsName"/>
```

```
<Value>*.iplanet.com</Value>
</AttributeValuePair>
<AttributeValuePair><Attribute name="EndIp"/>
<Value>145.15.15.15</Value>
</AttributeValuePair>
<AttributeValuePair><Attribute name="StartIp"/>
<Value>120.10.10.10</Value>
</AttributeValuePair>
</Condition>
<Condition name="authchain-realmqualfied"
        type="AuthenticateToServiceCondition">
<AttributeValuePair><Attribute name="AuthenticateToService"/>
<Value>/:ldapService</Value>
</AttributeValuePair>
</Condition>
<Condition name="auth to realm"
     type="AuthenticateToRealmCondition">
<AttributeValuePair><Attribute name="AuthenticateToRealm"/>
<Value>/</Value>
</AttributeValuePair>
</Condition>
<Condition name="authlevelge-realmqualified"
     type="AuthLevelCondition">
<AttributeValuePair><Attribute name="AuthLevel"/>
<Value>/:2</Value>
</AttributeValuePair>
</Condition>
<Condition name="authchain-nonrealmqualfied"
    type="AuthenticateToServiceCondition">
<AttributeValuePair><Attribute name="AuthenticateToService"/>
<Value>ldapService</Value>
</AttributeValuePair>
</Condition>
<Condition name="timecondition" type="SimpleTimeCondition">
<AttributeValuePair><Attribute name="EndTime"/>
<Value>17:00</Value>
</AttributeValuePair>
<AttributeValuePair><Attribute name="StartTime"/>
<Value>08:00</Value>
</AttributeValuePair>
<AttributeValuePair><Attribute name="EndDate"/>
<Value>2006:07:28</Value>
</AttributeValuePair>
```

```
<AttributeValuePair><Attribute name="EnforcementTimeZone"/>
<Value>America/Los_Angeles</Value>
</AttributeValuePair>
<AttributeValuePair><Attribute name="StartDay"/>
<Value>mon</Value>
</AttributeValuePair>
<AttributeValuePair><Attribute name="StartDate"/>
<Value>2006:01:02</Value>
</AttributeValuePair>
<AttributeValuePair><Attribute name="EndDay"/>
<Value>fri</Value>
</AttributeValuePair>
</Condition>
</Conditions>
<ResponseProviders name="responseproviders"
    description="description">
<ResponseProvider name="idresponseprovidere"
   type="IDRepoResponseProvider">
<AttributeValuePair>
<Attribute name="DynamicAttribute"/>
</AttributeValuePair>
<AttributeValuePair>
<Attribute name="StaticAttribute"/>
<Value>m=10</Value>
<Value>n=30</Value>
</AttributeValuePair>
</ResponseProvider>
</ResponseProviders>
</Policy>
```

2 **Once the policy XML file is developed, you can use the following command to load it:**

```
FederatedAccessManager-base/SUNWam/bin/amadmin
--runasdn "uid=amAdmin,ou=People,default_org,
root_suffix"
--password password
--data policy.xml
```

To add multiple policies simultaneously, place the policies in one XML file, as opposed to having one policy in each XML file. If you load policies with multiple XML files in quick succession, the internal policy index may become corrupted and some policies may not participate in policy evaluation.

When creating policies through `amadmin`, ensure that the authentication module is registered with the realm while creating authentication scheme condition; that the corresponding LDAP objects realms, groups, roles and users) exist while creating realms, LDAP groups, LDAP roles and LDAP user subjects; that Federated Access Manager roles exist while creating `IdentityServerRoles` subjects; and that the relevant realms exist while creating sub realm or peer realm referrals.

Please note that in the text of Value elements in `SubrealmReferral`, `PeerRealmReferral`, `Realm` subject, `IdentityServerRoles` subject, `LDAPGroups` subject, `LDAPRoles` subject and `LDAPUsers` subject need to be the full DN.

▼ To Create a Normal Policy With the Federated Access Manager Console

1 **Choose the realm for which you would like to create a policy.**

2 **Click the Policies tab.**

3 **Click New Policy from the Policies list.**

4 **Add a name and a description for the policy.**

5 **If you wish the policy to be active, select Yes in the Active attribute.**

6 **It is not necessary to define all of the fields for normal policies at this time. You may create the policy, then add rules, subjects, conditions, and response providers later. See "Managing Policies" on page 121 for more information.**

7 **Click OK.**

▼ To Create a Referral Policy With the Federated Access Manager Console

1 **Choose the realm for which you would like to create the policy.**

2 **Click New Referral from the Policies tab.**

3 **Add a name and a description for the policy.**

4 **If you wish the policy to be active, select Yes in the Active attribute.**

5 **It is not necessary to define all of the fields for referral policies at this time. You may create the policy, then add rules and referrals later. See "Managing Policies" on page 121 for more information.**

6 **Click OK.**

Creating Policies for Peer Realms and Sub Realms

In order to create policies for peer or sub realms, you must first create a referral policy in the parent (or another peer) realm. The referral policy must contain, in its rule definition, the resource prefix that is being managed by the sub realm. Once the referral policy is created in the parent realm (or another peer realm) normal policies can be created at the sub realm (or peer realm).

In this example, `o=isp` is the parent realm and `o=example.com` is the sub realm that manages resources and sub-resources of `http://www.example.com`.

▼ To Create a Policy for a Sub Realm

1 **Create a referral policy at `o=isp`. For information on referral policies, see the procedure "Modifying a Referral Policy" on page 125.**

The referral policy must define `http://www.example.com` as the resource in the rule, and must contain a `SubRealmReferral` with `example.com` as the value in the referral.

2 **Navigate to the sub realm `example.com`.**

3 **Now that the resource is referred to `example.com` by `isp`, normal policies can be created for the resource `http://www.example.com`, or for any resource starting with `http://www.example.com`.**

To define policies for other resources managed by `example.com`, additional referral policies must be created at `o=isp`.

Exporting Policies to Other Federated Access Manager instances

Federated Access Manager allows you to export policies using the `amadmin` command line tool This is useful when you wish to move many existing policies to another Federated Access Manager instance, or if you wish to inspect changes that you have made to existing policies in batch mode. To export policies, use the `amadmin` command line utility to export the specified policies to a file. The syntax is:

`amamdin - u` *username* `–w` *password* `–ofilename` *output_file.xml* `–t` *policy_data_file.xml*

You can use the wildcard (*) in the policy name to match any string of characters.

The following is an example of the *policy_data_file.xml*:

```
<?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 Federated Access Manager 8 Admin CLI DTD//EN"
    "/opt/SUNWam/dtd/amAdmin.dtd"
>>

<!--  CREATE REQUESTS -->

<!-- to export to file use option -ofilename fileName -->

<Requests>

<RealmRequests >
<RealmGetPolicies realm="/" >
<AttributeValuePair>
<Attribute name="policyName"/>
<Value>p*</Value>
</AttributeValuePair>
</RealmGetPolicies>
</RealmRequests>

<RealmRequests >
<RealmGetPolicies realm="/" >
<AttributeValuePair>
<Attribute name="policyName"/>
<Value>g10</Value>
<Value>g11</Value>
</AttributeValuePair>
</RealmGetPolicies>

</RealmRequests>
<RealmRequests >
<RealmGetPolicies realm="/realm1" >
<AttributeValuePair>
<Attribute name="policyName"/>
```

```
<Value>*</Value>
</AttributeValuePair>
</RealmGetPolicies>
</RealmRequests>

</Requests>
```

The policies are exported to the *Output_file.xml* file. You can now make any changes to policy definitions contained in the file. You must alter the output file so that it is compatible with the `amadmin` command utility before importing the policies to another Federated Access Manager instance. For instructions on how to import the policies, including an example of an `amadmin`-compatible policy data file, see To Create Policies with amadmin

Managing Policies

Once a normal or referral policy is created and added to Federated Access Manager, you can manage the policy through the Federated Access Manager console by modifying the rules, subjects, conditions and referrals.

Modifying a Normal Policy

Through the Policies tab, you can modify a normal policy that defines access permissions. You can define and configure multiple rules, subjects, conditions and resource comparators. This section lists and describes the steps to do so.

▼ To Add or Modify a Rule to a Normal Policy

1. **If you have already created the policy, click the name of the policy for which you wish to add the rule. If not, see "To Create a Normal Policy With the Federated Access Manager Console" on page 118.**

2. **Under the Rules menu, click New.**

3. **Select one of the following default service types for the rule. You may see a larger list if more services are enabled for the policy:**

Discovery Service	Defines the authorization actions for Discovery service query and modify protocol invocations by web services clients for a specified resource.

Liberty Personal Profile Service	Defines the authorization actions for Liberty Personal Profile service query and modify protocol invocations by web services clients for a specified resource.
URL Policy Agent	Defines authorization actions for the URL Policy Agent service. This is used to define policies that protect HTTP and HTTPS URLs. This is the most common use case of Federated Access Manager policies.

4 **Click Next.**

5 **Enter a name and resource name for the rule.**

Currently, Federated Access Manager Policy Agents only support `http://` and `https://` resources and do not support IP addresses in place of the hostname.

Wildcards are supported for protocol, host, port and resource name. For example:

```
http*://*:*/*.html
```

For the URL Policy Agent service, if a port number is not entered, the default port number is 80 for `http://`, and 443 for `https://`.

6 **Select the action for the rule. Depending on the service type, you can select the following:**

- LOOKUP (Discovery Service)
- UPDATE (Discovery Service)
- MODIFY (Liberty Personal Profile Service)
- QUERY (Liberty Personal Profile Service)
- GET (URL Policy Agent)
- POST (URL Policy Agent)

7 **Select the Action Values.**

- Interaction for Consent — Invokes the Liberty interaction protocol for consent on a resource. This is for the Liberty Personal Profile service type only.
- Interaction for Value — Invokes the Liberty interaction protocol for a value on a resource. This is for the Liberty Personal Profile service type only.
- Allow — Enables you access the resource matching the resource defined in the rule.
- Deny — Denies access to the resource matching the resource defined in the rule.

  Denial rules always take precedence over allow rules in a policy. For example, if you have two policies for a given resource, one denying access and the other allowing access, the result is a deny access (provided that the conditions for both policies are met). It is recommended that deny policies be used with extreme caution as they may lead to potential conflicts between the policies. Typically, the policy definition process should only use allow rules, and use the default deny when no policies apply to accomplish the deny case.

If explicit deny rules are used, policies that are assigned to a given user through different subjects (such as role and/or group membership) may result in denied access to a resource even if one or more of the policies allow access. For example, if there is a deny policy for a resource applicable to an Employee role and there is another allow policy for the same resource applicable to Manager role, policy decisions for users assigned both Employee and Manager roles would be denied.

One way to resolve such problems is to design policies using Condition plug-ins. In the case above, a "role condition" that applies the deny policy to users authenticated to the Employee role and applies the allow policy to users authenticated to the Manager role helps differentiate the two policies. Another way could be to use the `authentication level` condition, where the Manager role authenticates at a higher authentication level.

8. **Click Finish.**

▼ To Add or Modify a Subject to a Normal Policy

1. **If you have already created the policy, click the name of the policy for which you wish to add the subject. If you have not yet created the policy, see "To Create a Normal Policy With the Federated Access Manager Console" on page 118.**

2. **Under the Subject list, click New.**

3. **Select one of the default subject types. For descriptions of the subject types, see "Subjects" on page 102**

4. **Click Next.**

5. **Enter a name for the subject.**

6. **Select or deselect the Exclusive field.**

   If this field is not selected (default), the policy applies to the identity that is a member of the subject. If the field is selected, the policy applies to the identity that is *not* a member of the subject.

   If multiple subjects exist in the policy, the policy applies to the identity when at least one of the subjects implies that the policy applies to the given identity.

7. **Perform a search in order to display the identities to add to the subject. This step is not applicable for the Authenticated Users subject or Web Services Client subjects.**

   The default (*) search pattern will display all qualified entries.

8 **Select the individual identities you wish to add for the subject, or click Add All to add all of the identities at once. Click Add to move the identities to the Selected list. This step is not applicable for the Authenticated Users subject.**

9 **Click Finish.**

10 **To remove a subject from a policy, select the subject and click Delete. You can edit any subject definition by clicking on the subject name.**

▼ To Add a Condition to a Normal Policy

1 **If you have already created the policy, click the name of the policy for which you wish to add the condition. If you have not yet created the policy, "To Create a Normal Policy With the Federated Access Manager Console" on page 118**

2 **Under the Conditions list, click New.**

3 **Select the condition type and click Next.**

4 **Define the fields for the condition type.**

5 **Click Finish.**

▼ To Add a Response Provider to a Normal Policy

1 **If you have already created the policy, click the name of the policy for which you wish to add the response provider. If you have not yet created the policy, see "To Create a Normal Policy With the Federated Access Manager Console" on page 118.**

2 **Under the Response Providers list, click New.**

3 **Enter a name for the response provider.**

4 **Define the following values:**

StaticAttribute	These are static attributes in attribute value format, defined in an instance of `IDResponseProvider`stored in the policy.
DynamicAttribute	The response attributes chosen here need to first be defined in the Policy Configuration Service for the corresponding realm. The attribute names defined should be a subset of those existing in the configured datastore (IDRepository). For details on how to define the attributes see the Policy Configuration attribute definitions. To select specific or multiple attributes, hold the Control key and click the left mouse button.

5 **Click Finish.**

6 **To remove response provider from a policy, select the subject and click Delete. You can edit any response provider definition by clicking on the name.**

Modifying a Referral Policy

You can delegate policy definitions and decisions of a realm to different realms using referral policies. Custom referrals can used to get policy decisions from any policy destination point. Once you have created a referral policy, you can add or modify associated the rules, referrals, and resource providers.

▼ To Add or Modify a Rule to a Referral Policy

1 **If you have already created the policy, click the name of the policy for which you wish to add the rule. If not, see "To Create a Referral Policy With the Federated Access Manager Console" on page 118.**

2 **Under the Rules menu, click New.**

3 **Select one of the following default service types for the rule. You may see a larger list if more services are enabled for the policy:**

Discovery Service	Defines the authorization actions for Discovery service query and modify protocol invocations by web services clients for a specified resource.
Liberty Personal Profile Service	Defines the authorization actions for Liberty Personal Profile service query and modify protocol invocations by web services clients for a specified resource.
URL Policy Agent	Defines authorization actions for the URL Policy Agent service. This is used to define policies that protect HTTP and HTTPS URLs. This is the most common use case of Federated Access Manager policies.

4 **Click Next.**

5 **Enter a name and resource name for the rule.**

Currently, Federated Access Manager Policy Agents only support `http://` and `https://` resources and do not support IP addresses in place of the hostname.

Wildcards are supported for protocol, host, port and resource name. For example:

```
http*://*:*/*.html
```

For the URL Policy Agent service, if a port number is not entered, the default port number is 80 for `http://`, and 443 for `https://`.

Note – Steps 6 and 7 are not applicable for a referral policy.

---

6 Click Finish.

▼ To Add or Modify Referrals to a Policy

1 If you have already created the policy, click the name of the policy for which you wish to add the response provider. If you have not yet created the policy, see "To Create a Referral Policy With the Federated Access Manager Console" on page 118.

2 Under the Referrals list, click New.

3 Define the resource in the Rules fields. The fields are:

Referral— Displays the current referral type.

Name— Enter the name of the referral.

Resource Name— Enter the name of the resource.

Filter— Specifies a filter for the realm names that will be displayed in the Value field. By default, it will display all realm names.

Value — Select the realm name of the referral.

4 Click Finish.

To remove a referral from a policy, select the referral and click Delete.

You can edit any referral definition by clicking on the Edit link next to the referral name.

▼ To Add a Response Provider to a Referral Policy

1 If you have already created the policy, click the name of the policy for which you wish to add the response provider. If you have not yet created the policy, see "To Create a Referral Policy With the Federated Access Manager Console" on page 118.

2 Under the Response Providers list, click New.

3 Enter a name for the response provider.

4 Define the following values:

StaticAttribute	These are static attributes in attribute value format, defined in an instance of `IDResponseProvider`stored in the policy.
DynamicAttribute	The response attributes chosen here need to first be defined in the Policy Configuration Service for the corresponding realm. The attribute names defined should be a subset of those existing in the configured datastore (IDRepository). For details on how to define the attributes see the Policy Configuration attribute definitions. To select specific or multiple attributes, hold the Control key and click the left mouse button.

5 **Click Finish.**

6 **To remove response provider from a policy, select the subject and click Delete. You can edit any response provider definition by clicking on the name.**

Policy Configuration Service

The Policy Configuration service is used to configure policy-related attributes for each organization through the Federated Access Manager console. You can also define resource name implementations and Directory Server data stores for use with the Federated Access Manager policy framework. The Directory Server specified in the Policy Configuration Service is used for membership evaluation of LDAP Users, LDAP Groups, LDAP Roles, and organization policy subjects.

Subjects Result Time To Live

To improve policy evaluation performance, membership evaluations are cached for a period of time as defined by the Subjects Result Time To Live attribute in the Policy Configuration service. These cached membership decisions are used until the time defined in the Subjects Result Time To Live attribute has elapsed. Membership evaluation after this is used to reflect the current state of users in the directory.

Dynamic Attributes

These are the allowed dynamic attribute names which are displayed in a list and chosen to define policy response provider dynamic attributes. The names that are defined need to be same as attribute names as defined in the data repository.

amldapuser Definition

`amldapuser` is a user created during installation used by default to the Directory Server specified in the Policy Configuration service. This can be changed, as necessary, by the administrator or policy administrator of the realm.

Adding Policy Configuration Services

When the realm is created, Policy Configuration service attributes are automatically set for the realm. You can, however, modify the attributes as needed.

Resource-Based Authentication

Some organizations require an advanced authentication scenario where a user authenticates against a particular module based on the resource that they are attempting to access. Resource-based authentication is a feature of Federated Access Manager in which a user must authenticate to a specific authentication module protecting the resource, and not to the default authentication module. This feature is only applicable to first time user authentications.

Note – This is a separate feature than the resource-based authentication described in "Session Upgrade" on page 96. That particular feature does not have any limitations.

Limitations

Resource—based authentication contains the following limitations:

- If the policies applicable to the resource have multiple authentication modules, the system will arbitrarily pick one authentication module.
- Level and scheme are the only conditions that can be defined for this policy.
- This feature does not work across different DNS domains.

▼ To Configure Resource—based Authentication

Once both the Federated Access Manager and a policy agent have been installed, resource—based authentication can be configured. To do this, it is necessary to point Federated Access Manager to the Gateway servlet.

1 **Open** `AMAgent.properties`**.**

`AMAgent.properties` can be found (in a Solaris environment) in `/etc/opt//SUNWam/agents/config/`.

2 **Comment out the following line:**

```
#com.sun.am.policy.am.loginURL = http://Federated Access
Manager_server_host.domain_name:port/amserver/UI/Login.
```

3 **Add the following line to the file:**

```
com.sun.am.policy.am.loginURL =
http://AccessManager_host.domain_name:port/amserver/gateway
```

Note – The gateway servlet is developed using the Policy Evaluation APIs and can be used to write a custom mechanism to accomplish resource-based authentication. See the Chapter 3, "Using the Policy APIs," in *Sun Java System Access Manager 7.1 Developer's Guide* in the Federated Access Manager Developer's Guide.

---

4 **Restart the agent.**

CHAPTER 6

Managing Subjects

The Subjects interface enables basic identity management within a realm. Any identity that you create in the Subjects interface can be used in the subject definition in the for a policy created with the Federated Access Manager Identity Subject type.

The identities you can create and modify are:

- "User" on page 131
- "Agents Profile" on page 133
- "Filtered Role" on page 135
- "Roles" on page 135
- "Groups" on page 136

User

A *user* represents an individual's identity. Users can be created and deleted in groups and can be added or removed from roles and/or groups. You can also assign services to the user.

▼ To Create or Modify a User

1 **Click on the User tab.**

2 **Click New.**

3 **Enter data for the following fields:**

UserId. This field takes the name of the user with which he or she will log into Federated Access Manager. This property may be a non-DN value.

First Name. This field takes the first name of the user.

Last Name. This field takes the last name of the user.

Full Name — This field takes the full name of the user.

Password. — This field takes the password for the name specified in the User Id field.

Password (Confirm) — Confirm the password.

User Status. This option indicates whether the user is allowed to authenticate through Federated Access Manager.

4. **Click Create.**

5. **Once the user is created, you can edit the user information by clicking the name of the user. For information on the user attributes, see the User attributes. Other modifications you can perform:**
   - "To Create or Modify a User" on page 131
   - "To Add a User to Roles and Groups" on page 132
   - "To Add Services to an Identity" on page 132

▼ To Add a User to Roles and Groups

1. **Click the name of the user you wish to modify.**

2. **Select Roles or Groups. Only the roles and groups that have already been assigned to the user are displayed.**

3. **Select the roles or groups from the Available list and click Add.**

4. **Once the roles or groups are displayed in the Selected list, click Save.**

▼ To Add Services to an Identity

1. **Select the identity to which you wish to add services.**

2. **Click on the Services tab.**

3. **Click Add.**

4. **Depending on the identity type you selected, the following list of services are displayed:**
   - Authentication Configuration
   - Discovery Service
   - Liberty Personal Profile Service
   - Session

- User

5 **Select the service you with to add and click Next.**

6 **Edit the attributes for the service. For a description of the services, click on the service name in Step 4.**

7 **Click Finish.**

Agents Profile

Federated Access Manager Policy Agents protect content on web servers and web proxy servers from unauthorized intrusions. They control access to services and web resources based on the policies configured by an administrator.

The *agent* object defines a Policy Agent profile, and allows Federated Access Manager to store authentication and other profile information about a specific agent that is protecting an Federated Access Manager resource. Through the Federated Access Manager console, administrators can view, create, modify and delete agent profiles.

the agent object creation page is the location where you can define the UID/password with which the agent authenticated to Federated Access Manager. If you have a multiple web containers set up using the same Federated Access Manager, this gives you the option of enabling multiple IDs for different agents and to enable and disable them independently of Federated Access Manager. You can also manage some preference values for the agents centrally, rather than editing the `AMAgent.properties` on each machine.

▼ To Create or Modify an Agent

1 **Click the Agents tab.**

2 **Click New.**

3 **Enter the values for the following fields:**

Name. Enter the name or identity of the agent. This is the name that the agent will use to log into Federated Access Manager. Multi-byte names are not accepted.

Password. Enter the agent password. This password must be different than the password used by the agent during LDAP authentication.

Confirm Password. Confirm the password.

Device Status. Enter the device status of the agent. If set to Active, the agent will be able to authenticate to and communicate with Federated Access Manager. If set to Inactive, the agent will not be able to authenticate to Federated Access Manager.

4 **Click Create.**

5 **Once you have crated the agent, you can additionally edit the following fields:**

Description. Enter a brief description of the agent. For example, you can enter the agent instance name or the name of the application it is protecting.

Agent Key Value. Set the agent properties with a key/value pair. This property is used by Federated Access Manager to receive agent requests for credential assertions about users. Currently, only one property is valid and all other properties will be ignored. Use the following format:

agentRootURL=`protocol://` *hostname:port*/

The entry must be precise and `agentRootURL` is case sensitive.

protocol	Represents the protocol used, either HTTP or HTTPS.
hostname	Represents the host name of the machine on which the agent resides. This machine also hosts the resources that the agent protects.
port	Represents the port number on which the agent is installed. The agent listens to incoming traffic on this port and intercepts all requests to access resources on the host.

Configuring Federated Access Manager to Protect Against Cookie Hijacking

Cookie hijacking refers to a situation where an imposter (a hacker, perhaps using an untrusted application) gains unauthorized access to cookies. When the cookies being hijacked are session cookies, cookie hijacking can potentially increase the threat of unauthorized access to protected web resources, depending on how the system is configured.

Sun documentation provides a technical note entitled, “Precautions Against Session-Cookie Hijacking in an Access Management Deployment” which provides information about precautions you can take to against specific security threats related to session-cookie hijacking. See the following document:

Technical Note: Precautions Against Cookie Hijacking in an Access Manager Deployment

Filtered Role

A filtered role is a dynamic role created through the use of an LDAP filter. All users are funneled through the filter and assigned to the role at the time of the role's creation. The filter looks for any attribute value pair (for example, `ca=user*`) in an entry and automatically assign the users that contain the attribute to the role.

▼ To Create a Filtered Role

1. **In the Navigation pane, go the organization where the role will be created.**
2. **Click New.**
3. **Enter a name for the filtered role.**
4. **Enter the information for the search criteria.**

   For example,

   ```
   (&(uid=user1)(|(inetuserstatus=active)(!(inetuserstatus=*))))
   ```

   If the filter is left blank, by default, the following role is created:

   ```
   (objectclass = inetorgperson)
   ```
5. **Click Create to initiate the search based on the filter criteria. The identities defined by the filter criteria are automatically assigned to the role.**
6. **Once the filtered role is created click the name of the role to view the Users that belong to the role. You can also add services to the role by clicking the Services tab.**

Roles

A role's members are LDAP entries that posses the role. The criteria of the role itself is defined as an LDAP entry with attributes, identified by the Distinguished Name (DN) attribute of the entry. Once the role is created, you manually add services and users.

▼ To Create or Modify a Role

1. **Click the Role tab.**
2. **Click New in the Role list.**
3. **Enter a name for the role.**
4. **Click Create.**

▼ To Add Users to a Role or Group

1. **Click the name of the role or group for which you wish to add users.**
2. **Click the Users tab.**
3. **Select the users you wish to add from the Available list and click Add.**
4. **Once the users are displayed in the Selected list, click Save.**

Groups

A *group* represents a collection of users with a common function, feature or interest. Typically, this grouping has no privileges associated with it. Groups can exist at two levels; within an organization and within other managed groups.

▼ To Create or Modify a Group

1. **Click the Group tab.**
2. **Click New from the Group list.**
3. **Enter a name for the group.**
4. **Click Create.**

   Once you have created the group, you can add users to the group by clicking the name of the group and then the User tab.

P A R T I I

Federation, Web Services, and SAML Administration

This is part two of the *Sun Federated Access Manager 8.0 Administration Guide* and describes how implement, configure and manage Federated Access Manager's Federation, SAML and SAMLv2 and Web Services features.

CHAPTER 7

Federation

Sun Java^{TM} System Federated Access Manager provides an interface for creating, modifying, and deleting authentication domains, service providers, and identity providers. This chapter explains how to use the Federation module to configure these components, allowing for Liberty-based provider federation. It covers the following topics:

Entities and Authentication Domains

The Federation component in the Federated Access Manager Console provides an interface for configuring, modifying, and deleting authentication domains, and its member identity providers and service providers. To enable provider federation using Federated Access Manager, create and populate an authentication domain using the following process:

1. Create an *entity* to hold the *metadata* (information that defines a particular identity services architecture) for each provider that will become a member of the authentication domain.

   See **Broken Link (Target ID: GASUV)**.
2. Configure and save an authentication domain.

   See **Broken Link (Target ID: ADMDJ)**.
3. Add an entity (a configured provider) to the authentication domain by configuring the entity's properties to add the authentication domain and configuring the authentication domain's properties to add the entity.

   Information on configuring the entity's properties can be found in **Broken Link (Target ID: GDWXJ)** or **Broken Link (Target ID: GDWWI)**. Information on configuring the authentication domain's properties can be found in **Broken Link (Target ID: FXIXJ)**.

Note – The establishment of contractual agreements between providers is beyond the scope of this guide. For information, see the Liberty Trust Model Guidelines.

The following sections contain more detailed information:

- **Broken Link (Target ID: GAFRF)**
- **Broken Link (Target ID: ADMDJ)**

Tip – In a federation setup, all service providers and identity providers must share a synchronized clock. You can implement the synchronization by pointing to an external clock source or by ensuring that, in case of delays in receiving responses, the responses are captured without fail through adjustments of the time outs.

Entities

An *entity* may be configured with metadata (configuration information that defines a particular identity service architecture) for an individual identity provider, an individual service provider, or one of each. Contrarily, an entity may be configured as an *affiliation*, a selected group of providers of either type. Both provider and affiliation entities can be configured using the Federated Access Manager Console.

Note – For general information about entities, see the *Liberty Metadata Description and Discovery Specification*.

Provider Entity	A *provider entity* holds the metadata for individual providers of either type. All identity providers and service providers (both hosted and remote) must be configured within a provider entity before they can be associated with an authentication domain, or chosen to be included in an *affiliate entity*. Using the attributes provided in the Federated Access Manager Console, one individual identity provider, one individual service provider, or one of each can be defined within a provider entity.
Affiliate Entity	A configured *affiliation* (referenced by an `affiliationID`) contains a grouping of provider sites. The affiliation is formed and maintained by an *affiliation owner* who chooses the member providers from already configured provider entities. (An affiliation is formed without regard to the boundaries of any authentication domains which might also include the providers as members.) The affiliation enables a user to federate amongst the group of associated sites. The chosen providers may invoke services either as a member of the affiliation, or individually as a provider. If services are invoked as an affiliation member, a service provider might issue an authentication request for a user on behalf of an affiliation. When authentication is secured, the user can achieve single sign-on with all members of the affiliation. An *affiliate entity* holds the metadata that defines the grouping of one or more provider entities that comprise the affiliation. It does not contain the

configuration information for any providers (which is defined in a *provider entity*), only the configuration information for the affiliation itself.

Tip – The name identifier (a single persistent randomized string) is used to achieve single sign-on between an identity provider and a group of service providers acting as a single affiliation. If there are several service providers and identity providers in the same circle of trust, use an affiliate entity to avoid having to generate different name identifiers for commonly shared services.

Configuring an entity using the Federated Access Manager Console is a two-step process. First, you create a provider or affiliate entity. Then, you populate the entity with either remote or hosted provider metadata (either service or identity) or affiliation information. This process is described in the following sections.

- **Broken Link (Target ID: GASUV)**
- **Broken Link (Target ID: GDWXC)**
- **Broken Link (Target ID: GASVW)**
- **Broken Link (Target ID: GASTS)**
- **Broken Link (Target ID: GDOHT)**

Note – This section contains information on how entities can be created and configured in one step using the `amadmin` command-line interface and prepared XML files (as opposed to the manual configuration illustrated in the previous sections).

Creating Entities

This section describes the process for creating a provider entity or an affiliate entity.

▼ To Create a Provider Entity or an Affiliate Entity

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). Once created and populated, the entity (and thus the member providers) can be added to an authentication domain.

1. **In the Federated Access Manager Console, select the Federation tab.**

2. **Under Federation, select the Entities tab.**

3. **Select New.**

   The new entity attributes are displayed.

4 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`.

5 (Optional) Enter a description of the entity in the Description field.

6 Select one of the following options to define the entity's type.

- **Select Provider and click OK.**

  The new entity is now displayed as a provider in the list of configured Entities. To configure the entity, see **Broken Link (Target ID: GDWWJ)**.

- **Select Affiliate, type a value for both Affiliate Name and Affiliate Owner, and click OK.**

  The Affiliate Name (or `affiliationID`) specifies a URI that uniquely represents the affiliate entity. For example, `http://shivalik.sun.com` or `http://provider2.com:875`. The Affiliate Owner (or `providerID`) is the value assigned to the Entity Name attribute of the provider entity that is forming the affiliation. After entering these values and clicking OK, the new entity is displayed as an affiliate in the list of configured Entities. To configure the entity, see **Broken Link (Target ID: ADMDF)**.

  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 as both.

Configuring Provider Entities

After you create a provider entity, you populate it with remote or hosted provider information (either service or identity). This section contains the following procedures:

- **Broken Link (Target ID: GDWWJ)**
- **Broken Link (Target ID: GDWXA)**
- **Broken Link (Target ID: GDWXJ)**
- **Broken Link (Target ID: GDWWI)**

▼ To Configure a Provider Entity

When you configure a provider entity, you are populating it with remote or hosted provider information (either service or identity). You might also be defining values for attributes that were not available when the entity was initially created. Before performing this procedure, you must have completed the steps in **Broken Link (Target ID: ADMDA)**.

1 In the Federated Access Manager Console, select the Federation tab.

2 **Under Federation, select the Entities tab.**

3 **Select the provider entity that you want to configure.**

Ensure that you select an entity marked as type Provider.

4 **Define values for the General, Identity Provider or Service Provider attributes by choosing from the View menu.**

- **To define values for General attributes, see Broken Link (Target ID: GDWXA).**
- **To define values for Identity Provider attributes, see Broken Link (Target ID: GDWXJ).**
- **To define values for Service Provider attributes, see Broken Link (Target ID: GDWWI).**

▼ To Configure General Attributes for a Provider Entity

Before performing this procedure, you must have completed the steps in **Broken Link (Target ID: GDWWJ)**.

1 **Choose General from the View menu, and provide information for the Entity Common Attributes.**

Entity Common Attributes contain values that define the entity itself.

Entity Name
: The static value of this attribute is the name that you provided when creating the entity.

Type
: The static value of this attribute is Provider.

Description
: The value of this optional attribute is the description that you provided when creating the entity. You can modify the description.

2 **Provide information for the Entity Contact Person Profile attributes.**

Entity Contact Person Profile attributes contain values that define the administrator of the entity.

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 in New Value and click Add.

Telephone Numbers
: Type one or more telephone numbers for the contact person in New Value and click Add.

3 (Optional) Provide information for the Organization Profiles.

The Organization Profiles attributes contain values that define the organizational name of the entity.

Names
: Type the complete legal name of the entity's organization in New Value and click Add. 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 in New Value and click Add. 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 in New Value and click Add. Use the format *locale*|*organization-URL*. For example, `en|http://www.`*organization-name*`.com`.

4 Click Save to complete the configuration, or define additional values for the Identity Provider or Service Provider attributes by choosing from the View menu.

- **To define values for Identity Provider attributes, see Broken Link (Target ID: GDWXJ).**
- **To define values for Service Provider attributes, see Broken Link (Target ID: GDWWI).**

▼ To Configure Hosted or Remote Identity Provider Attributes for a Provider Entity

Before performing this procedure, you must have completed the steps in **Broken Link (Target ID: GDWWJ)**.

Note – Some of the attributes below will only be visible after you have saved the initial provider configuration.

---

1 **Choose Identity Provider from the View menu.**

2 **Select the type of provider that you are configuring:**

- New Hosted Provider

  A *hosted provider* is installed on the same server as Federated Access Manager.
- New Remote Provider

  A *remote provider* is not installed on the same server as Federated Access Manager.

3 **Provide information for the Common Attributes.**

Common Attributes contain values that generally define the identity provider.

Provider Type
: The static value of this attribute is the type of provider being configured: hosted or remote.

Description
: The value of this attribute is a description of the identity provider.

Protocol Support Enumeration
: Choose the Liberty ID-FF release that is supported by this provider.
  - `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 in New Value 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`.

Signing Key: Key Alias
: Type the key alias that is 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 that are used by the web service consumer when interacting with another entity.

Note – If the encryption method is DESede, the key size must be 192. If the encryption method is AES, the key size must be 128, 192 or 256.

Encryption Key: Encryption Method
: Choose the method of encryption:

- None
- AES
- DESede

Name Identifier Encryption
: Select the check box to enable encryption of the name identifier.

4 Provide information for the Communication URLs.

Communication URLs attributes contain locations for redirects and sending requests.

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 a new name identifier to 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 **Provide information for the Communication Profiles.**

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 – Federated Access 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, you can define this attribute later.

Note – If configuring a remote identity provider, skip to step 11. If configuring a hosted identity provider, continue with step 7.

---

7 (Hosted Identity Provider Only) Provide mappings for the Authentication Context classes.

This attribute maps the Liberty-defined authentication context classes to authentication methods available from the identity provider.

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 Federated Access Manager authentication type to which the context is mapped.

Note – See “Authentication Types” on page 62 for more information.

Value
: Type the Federated Access Manager authentication option.

Priority
: Choose a priority level for cases where there are multiple contexts.

8 **(Hosted Identity Provider Only) Select any of the available provider entities to assign as a Trusted Provider and click Add.**

This attribute tallies providers that the identity provider trusts.

9 **(Hosted Identity Provider Only) Provide information for the Federated Access Manager Configuration attributes.**

Federated Access Manager Configuration attributes define general information regarding the instance of Federated Access Manager being used as an 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:

- Password
- Mobile Digital ID
- Smartcard
- Smartcard-PKI
- MobileUnregistered
- Software-PKI
- Previous-Session
- Mobile Contract
- Time-Sync-Token
- Password-ProtectedTransport

Realm
Type a value that 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 with 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 Federated Access 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 the `AutoFedAttribute` element and this common attribute as its value.

Attribute Statement Plug-in
: Specify a pluggable class used for adding attribute statements to an assertion that is generated during the Liberty-based single sign-on process.

Identity Provider Attribute Mapping
: Specify values to define the mappings used by the default attribute mapper plug-in. Mappings should be configured in the format:

  SAML-attribute=local-attribute

  For example, `EmailAddress=mail` or `Address=postaladdress`. Type the mapping as a New Value and click Add.

10 (Hosted Identity Provider Only) 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.

Note – To continue configuring a hosted identity provider, skip to step 12.

---

11 **(Remote Identity Provider Only) Provide information for the Proxy Authentication Configuration attributes.**

Proxy Authentication Configuration attributes define values for dynamic identity provider proxying.

Proxy Authentication
: Select the check box to enable proxy authentication for a service provider.

Proxy Identity Providers List
: Type an identifier for an identity provider(s) that can be used for proxy authentication in New Value and click Add. The value is a URI defined as the provider's identifier.

Maximum Number of Proxies
: Enter the maximum number of identity providers that can be used for proxy authentication.

Use Introduction Cookie for Proxying
: Select the check box if you want introductions to be used to find the proxying identity provider.

12 **(Optional) Provide information for the Organization Profiles.**

The Organization Profiles attributes contain values that define the organizational name of the entity.

Names
: Type the complete legal name of the organization in New Value and click Add. 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 also.

Display Names
: Type a name that is suitable for display to a principal in New Value and click Add. The value is defined in 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 in New Value and click Add. Use the format *locale|organization-URL*, for example, `en|http://www.`*organization-name*`.com`.

13 Click New Contact Person to create a contact person for the provider.

The Contact Person attributes contain information regarding a human contact for the identity provider.

First Name
: Type the given name of the identity provider's contact person.

Last Name
: Type the surname of the identity provider's contact person.

Type
: Choose the contact's role from the drop-down menu:

- Administrative
- Billing
- Technical
- Other

Company
: Type the name of the company that employs the contact person.

Liberty Principal Identifier
: Type the name identifier 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 in New Value and click Add.

Telephone Numbers
: Type one or more telephone numbers for the contact person in New Value and click Add.

14 Click Create to create the contact person.

15 Click Save to complete the configuration, or define values for General or Service Provider attributes by choosing from the View menu:

- **To define values for General attributes, see Broken Link (Target ID: GDWXA).**
- **To define values for Service Provider attributes, see Broken Link (Target ID: GDWWI).**

▼ To Configure Hosted or Remote Service Provider Attributes for a Provider Entity

Before performing this procedure, you must have completed the steps in **Broken Link (Target ID: GDWWJ)**.

Note – Some of the attributes below will only be visible after you have saved the initial provider configuration.

---

1 **Choose Service Provider from the View menu.**

2 **Select the type of provider that you are configuring:**

- New Hosted Provider

  A *hosted provider* is installed on the same server as Federated Access Manager.
- New Remote Provider

  A *remote provider* is not installed on the same server as Federated Access Manager.

3 **Provide information for the Common Attributes.**

Common Attributes contain values that generally define the service provider.

Provider Type
: The static value of this attribute is the type of provider being configured: hosted or remote. This attribute is visible only after saving your configuration.

Description
: The value of this attribute is a description of the service provider.

Protocol Support Enumeration
: Select the Liberty ID-FF release that is supported by this provider.

  - `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 in New Value 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`.

Signing Key: Key Alias
: Type the key alias that is 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 that are used by the web service consumer when interacting with another entity.

Encryption Key: Encryption Method
: Select the method of encryption:

- None
- AES
- DESede

Name Identifier Encryption
: Select the check box to enable encryption of the name identifier.

4 Provide information for the Communication URLs.

Communication URLs attributes contain locations for redirects and sending requests.

SOAP Endpoint
: Type a URI 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 identity providers can send logout requests.

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 identity providers 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 that will be used when communicating with the identity provider to specify a new name identifier for the principal. (Registration can occur only after a federation session is established.)

Name Registration Return
: Type a URL to which the service provider will redirect the principal after HTTP name registration has been completed.

5 Provide information for the Communication Profiles.

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 – Federated Access 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 cannot participate in Liberty-based communications. If no authentication domains have been created, you can define this attribute later.

Note – If configuring a hosted service provider, skip to step 9. If configuring a hosted service provider, continue with step 7.

---

7 (Hosted Service Provider Only) Provide a hierarchy for the Authentication Context classes.

This attribute corresponds to the authentication level defined for an Federated Access Manager authentication module. It will redirect the principal to the authentication type with an authentication level equal to the number defined.

Context Reference
: The Liberty-defined authentication context classes are:

- Password
- Mobile Digital ID
- Smartcard
- Smartcard-PKI
- MobileUnregistered
- Software-PKI
- Previous-Session
- Mobile Contract
- Time-Sync-Token
- Password-ProtectedTransport

Level
: Type a level for each authentication context class. The number can be any positive number.

8 (Hosted Service Provider Only) Select any of the available provider entities to assign as a Trusted Provider and click Add.

This attribute tallies providers that the service provider trusts.

9 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 that defines where a provider will send 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 are:

  - *None* specifies that the identity provider will return the name identifier(s) for the principal corresponding to the federation that exists between the identity provider and the requesting service provider or affiliation group. If no such federation exists, an error will be returned.
  - *One-time* specifies that the identity provider will issue a temporary, one-time-use identifier for the principal after federation.
  - *Federation* specifies that the identity provider may start a new identity federation if one does not already exist for the principal.

Affiliation Federation
: Select the check box to enable affiliation federation.

Note – If configuring a remote service provider, skip to step 11. If configuring a hosted service provider, continue with step 10.

---

10 **(Hosted Service Provider Only) Provide information for the Federated Access Manager Configuration attributes.**

Federated Access Manager Configuration attributes define general information regarding the instance of Federated Access Manager being used as a service provider.

Service Provider Adapter
: Defines the implementation class for the `com.sun.identity.federation.plugins.FederationSPAdapter` interface, used to add application-specific processing during the federation process.

Provider Alias
: Type an alias name for the local service 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

This attribute defines the service provider's default authentication context class (method of authentication). This method will always be called when the service provider sends an authentication 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 (even during a live session) when an authentication request is received. This attribute is enabled by default.

Request Identity Provider to be Passive

Select the check box to specify that the identity provider must not interact with the principal and must interact with the user.

Realm

Type a value that 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 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 with 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 service 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 the `AutoFedAttribute` element and this common attribute as its value.

Attribute Mapper Class
: The class used to map attributes in the SAML assertion to user attributes defined locally by the service provider. The default class is `com.sun.identity.federation.services.FSDefaultAttributeMapper`.

Service Provider Attribute Mapping
: Specify values to define the mappings used by the default attribute mapper plug-in specified above. Mappings should be configured in the format:

  SAML-attribute=local-attribute

  For example, `EmailAddress=mail` or `Address=postaladdress`. Type the mapping as a New Value and click Add.

11 **Provide information for the Proxy Authentication Configuration attributes.**

Proxy Authentication Configuration attributes define values for dynamic identity provider proxying.

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. Type the URI defined as the provider's identifier in New Value and click Add.

Maximum Number of Proxies
: Enter the maximum number of identity providers that can be used for proxy authentication.

Use Introduction Cookie for Proxying
: Select the check box if you want introductions to be used to find the proxying identity provider.

12 **(Optional) Provide information for the Organization Profiles.**

The Organization Profiles attributes contain values that define the organizational name of the entity.

Names
: Type the complete legal name of the entity's organization in New Value and click Add. 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 in New Value and click Add. 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 in New Value and click Add. Use the format *locale*|*organization-URL*, for example, `en|http://www.`*organization-name*`.com`.

13 Click New Contact Person to create a contact person for the provider.

The Contact Person attributes contain information regarding a human contact for the identity provider.

First Name
: Type the given name of the identity provider's contact person.

Last Name
: Type the surname of the identity provider's contact person.

Type
: Choose the contact's role from the drop-down menu:

- Administrative
- Billing
- Technical
- Other

Company
: Type the name of the company that employs the contact person.

Liberty Principal Identifier
: Type the name identifier 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 in New Value and click Add.

Telephone Numbers
: Type one or more telephone numbers for the contact person in New Value and click Add.

14 Click Create to create the contact person.

15 **Click Save to complete the configuration, or define values for General or Identity Provider attributes by choosing from the View menu:**

- **To define values for General attributes, see Broken Link (Target ID: GDWXA).**
- **To define values for Identity Provider attributes, see Broken Link (Target ID: GDWXJ).**

Configuring Affiliate Entities

After you create an affiliate entity, you populate it with affiliation information. This section contains the following procedures:

- **Broken Link (Target ID: ADMDF)**
- **Broken Link (Target ID: ADMDG)**
- **Broken Link (Target ID: ADMDH)**

▼ To Configure an Affiliate Entity

Before performing this procedure, you must have completed the steps in **Broken Link (Target ID: ADMDA)**.

1 **In the Federated Access Manager Console, select the Federation tab.**

2 **Under Federation, select the Entities tab.**

3 **Select the entity that you want to configure.**

Ensure that you select an entity marked as type Affiliate.

4 **Define values for the General or Affiliate attribute groupings by choosing from the View menu:**

- **To define values for General attributes, see Broken Link (Target ID: ADMDG)**
- **To define values for Affiliate attributes, see Broken Link (Target ID: ADMDH)**

▼ To Configure General Attributes for an Affiliate Entity

Before performing this procedure, you must have completed the steps in **Broken Link (Target ID: ADMDF)**.

1 **Choose General from the View menu, and provide information for the Entity Common Attributes.**

Entity Common Attributes contain values that define the entity.

Entity Name
  The static value of this attribute is the name that you provided when creating the entity.

Type
: The static value of this attribute is Affiliate.

Description
: The value of this optional attribute is the description that you provided when creating the entity. You can modify the description.

2 Provide information for the Entity Contact Person Profile attributes.

Entity Contact Person Profile attributes contain values that define the administrator of the entity.

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 in New Value and click Add.

Telephone Numbers
: Type one or more telephone numbers for the contact person in New Value and click Add.

3 (Optional) Provide information for the Organization Profiles.

The Organization Profiles attributes contain values that define the organizational name of the entity.

Names
: Type the complete legal name of the organization in New Value and click Add. 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 also.

Display Names
: Type a name that is suitable for display to a principal in New Value and click Add. The value is defined in 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 in New Value and click Add. Use the format *locale|organization-URL*, for example, `en|http://www.`*organization-name*`.com`.

4 **Click Save to complete the configuration, or choose Affiliate from the View menu to configure the Affiliate attributes.**

To define values for Affiliate attributes, see **Broken Link (Target ID: ADMDH)**.

▼ To Configure Affiliate Attributes for an Affiliate Entity

Before performing this procedure, you must have completed the steps in **Broken Link (Target ID: ADMDF)**.

1 **Select any of the available provider entities to add to the affiliation.**

A provider must be a member of an authentication domain as, without a specified authentication domain, it cannot participate in Liberty-based communications. The provider can belong to one or more affiliations. Also, be sure that the selected provider has the Affiliation Federation attribute enabled and the Protocol Support Enumeration attribute set to `urn:liberty:iff:2003-08` to enable the Liberty ID-FF version 1.2.

2 **Choose Affiliate from the View menu and provide information for the Common Attributes.**

Common Attributes contain values that generally define the affiliation.

Name
: The value of this attribute is the name of the affiliation.

Owner
: The value of this attribute is the owner of the affiliation.

Signing Key: Key Alias
: Type the key alias that is used to sign requests and responses.

Encryption Key: Key Alias
: Type the security certificate alias. Certificates are stored in a JKS 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
: Select the method of encryption:

- None
- AES
- DESede

3 **Click Save to complete the configuration.**

4 **Click OK to complete the configuration, or choose General from the View menu to configure the General attributes.**

To define values for General attributes, see **Broken Link (Target ID: ADMDG)**.

Deleting Entities

If an entity is to be deleted from the console, it first needs to be manually removed from the Trusted Providers list (if the provider is hosted) or the Available Providers list (if part of an affiliation).

▼ To Delete a Provider or Affiliate Entity

1 **In the Federated Access Manager Console, click the Federation tab.**

2 **Under Federation, select the Entities tab.**

3 **Select the check box next to the entity that you want to delete.**

No warning message is displayed when performing a delete.

4 **Click Delete.**

Creating and Configuring Entities using `amadmin`

The previous sections detailed how to create and configure entities using the Federated Access Manager console. But entities can also be created and configured in one step using the `amadmin` command-line interface and prepared XML files. Rather than filling in provider attribute values manually, you would create an XML file containing the provider attributes and corresponding values and import it using `amadmin`. Alternatively, you can modify the sample provider metadata XML files included with Federated Access Manager.

Caution – The format of the XML file used as input is based on the `sms.dtd`, located in `/`*FederatedAccessManager-base*`/SUNWam/dtd`. Alterations to the DTD files may hinder the operation of Federated Access Manager.

There are two types of provider metadata (formatted in XML files) that can be used as input to `amadmin`:

- **Standard metadata** properties are defined in the Liberty ID-FF specification.
- **Extended metadata** properties are proprietary and used by features specific to Federated Access Manager.

Note – `amadmin` uses different options to load the different types of metadata XML files. Information on how to use `amadmin` can be found in “Using amadmin for Federation Management” in *Sun Java System Access Manager 7.1 Administration Reference*. Information regarding the attributes and possible values can be found in the online help of the Federated Access Manager console or in the following sections:

- **Broken Link (Target ID: GASUV)**
- **Broken Link (Target ID: GDWXC)**
- **Broken Link (Target ID: GASVW)**

Following are instructions to load the provider metadata:

- **Broken Link (Target ID: GDWWY)**
- **Broken Link (Target ID: GDWVS)**

Loading Standard Metadata Using `amadmin`

To load metadata compliant with the Liberty ID-FF use the following command:

```
amadmin --runasdn userdn --password password --import metadata_filename
```

This option is usually used to load provider metadata sent from a trusted partner in an XML file compliant with the Liberty ID-FF. Here is an example of a service provider metadata XML file compliant with the Liberty ID-FF.

EXAMPLE 7–1 Service Provider Standard Metadata XML File for `amadmin`

```
<!--
  Copyright (c) 2005 Sun Microsystems, Inc. All rights reserved
  Use is subject to license terms.
-->
```

EXAMPLE 7–1 Service Provider Standard Metadata XML File for `amadmin` *(Continued)*

```
<EntityDescriptor meta:providerID="http://sp10.com" meta:cacheDuration="360"
xmlns:meta="urn:liberty:metadata:2003-08" xmlns="urn:liberty:metadata:2003-08">
  <SPDescriptor cacheDuration="180" xmlns:meta="urn:liberty:metadata:2003-08"
   aaa="aaa" protocolSupportEnumeration="urn:liberty:iff:2003-08">
   <KeyDescriptor use="signing">
    <EncryptionMethod>http://something/encrypt</EncryptionMethod>
     <KeySize>4567</KeySize>
     <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
     <ds:X509Data xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
     <ds:X509Certificate xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      MIIC1DCCApICBD8poYwwCwYHKoZIzjgEAwUAMFAxCzAJBgNVBAYTAlVTMQwwCgYDVQQKEwNTdW4x
      IDAeBgNVBAsTF1NVTiBPTkUgSWRlbnRpdHkgU2VydmVyMREwDwYDVQQDEwhzdW4tdW5peDAeFw0w
      MzA3MzEyMzA5MDBaFw0wNDAxMjcyMzA5MDBaMFAxCzAJBgNVBAYTAlVTMQwwCgYDVQQKEwNTdW4x
      IDAeBgNVBAsTF1NVTiBPTkUgSWRlbnRpdHkgU2VydmVyMREwDwYDVQQDEwhzdW4tdW5peDCCAbcw
      ggEsBgcqhkjOOAQBMIIBHwKBgQD9f1OBHXUSKVLfSpwu7OTn9hG3UjzvRADDHj+AtlEmaUVdQCJR
      +1k9jVj6v8X1ujD2y5tVbNeBO4AdNG/yZmC3a5lQpaSfn+gEexAiwk+7qdf+t8Yb+DtX58aophUP
      BPuD9tPFHsMCNVQTWhaRMvZ1864rYdcq7/IiAxmd0UgBxwIVAJdgUI8VIwvMspK5gqLrhAvwWBz1
      AoGBAPfhoIXWmz3ey7yrXDa4V7l5lK+7+jrqgvlXTAs9B4JnUVlXjrrUWU/mcQcQgYC0SRZxI+hM
      KBYTt88JMozIpuE8FnqLVHyNKOCjrh4rs6Z1kW6jfwv6ITVi8ftiegEkO8yk8b6oUZCJqIPf4Vrl
      nwaSi2ZegHtVJWQBTDv+z0kqA4GEAAKBgCNS1il+RQAQGcQ87GBFde8kf8R6ZVuaDDajFYE4/LNT
      Kr1dhEcPCtvL+iUFi44LzJf8Wxh+eA5K1mjIdxOo/UdwTpNQSqiRrm4Pq0wFG+hPnUTYLTtENkVX
      IIvfeoVDkXnF/2/i1Iu6ttZckimOPHfLzQUL4ldL4QiaYuCQF6NfMAsGByqGSM44BAMFAAMvADAs
      AhQ6yueX7YlD7IlJhJ8D4l6xYqwopwIUHzX82qCzF+VzIUhi0JG7slSpyis=
     </ds:X509Certificate>
     </ds:X509Data>
     </ds:KeyInfo>
   </KeyDescriptor>
   <SingleLogoutServiceURL>http://www.sun.com/slo"</SingleLogoutServiceURL>
   <SingleLogoutServiceReturnURL>http://www.sun.com/sloservice
    </SingleLogoutServiceReturnURL>
   <FederationTerminationServiceURL>http://www.sun.com/fts
    </FederationTerminationServiceURL>
   <FederationTerminationServiceReturnURL>http://www.sun.com/ftsr
    </FederationTerminationServiceReturnURL>
   <FederationTerminationNotificationProtocolProfile>http://projectliberty.org/profiles/
    fedterm-sp-http</FederationTerminationNotificationProtocolProfile>
   <SingleLogoutProtocolProfile>http://projectliberty.org/profiles/slo-sp-http
    </SingleLogoutProtocolProfile>
   <RegisterNameIdentifierProtocolProfile>http://projectliberty.org/profiles/
    rni-sp-http</RegisterNameIdentifierProtocolProfile>
   <RegisterNameIdentifierServiceURL>http://www.sun2.com/risu
    </RegisterNameIdentifierServiceURL>
   <RegisterNameIdentifierServiceReturnURL>http://www.sun2.com/rstu
    </RegisterNameIdentifierServiceReturnURL>
   <RelationshipTerminationNotificationProtocolProfile>http://projectliberty.org/
    profiles/rel-term-soap</RelationshipTerminationNotificationProtocolProfile>
```

EXAMPLE 7–1 Service Provider Standard Metadata XML File for `amadmin` *(Continued)*

```
   <NameIdentifierMappingBinding AuthorityKind="ppp:AuthorizationDecisionQuery"
    Location="http://eng.sun.com" Binding="http://www.sun.com"
    xmlns:ppp="urn:oasis:names:tc:SAML:1.0:protocol"></NameIdentifierMappingBinding>
   <AdditionalMetaLocation namespace="abc">http://www.aol.com</AdditionalMetaLocation>
   <AdditionalMetaLocation namespace="efd">http://www.netscape.com</AdditionalMetaLocation>
   <AssertionConsumerServiceURL id="jh899" isDefault="true">
    http://www.iplanet.com/assertionurl</AssertionConsumerServiceURL>
   <AuthnRequestsSigned>true</AuthnRequestsSigned>
  </SPDescriptor>
  <ContactPerson xmlns:meta="urn:liberty:metadata:2003-08" contactType="technical"
   meta:libertyPrincipalIdentifier="myid">
  <Company>SUn Microsystems</Company>
  <GivenName>Joe</GivenName>
  <SurName>Smith</SurName>
  <EmailAddress>joe@sun.com</EmailAddress>
  <EmailAddress>smith@sun.com</EmailAddress>
  <TelephoneNumber>45859995</TelephoneNumber>
  </ContactPerson>
  <Organization xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <OrganizationName xml:lang="en">sun com</OrganizationName>
  <OrganizationName xml:lang="en">sun micro com</OrganizationName>
  <OrganizationDisplayName xml:lang="en">sun.com</OrganizationDisplayName>
  <OrganizationURL xml:lang="en">http://www.sun.com/liberty</OrganizationURL>
  </Organization>
</EntityDescriptor>
```

Loading Proprietary Metadata Using `amadmin`

Federated Access Manager provides proprietary attributes that are not a specific part of the Liberty ID-FF. To load Federated Access Manager proprietary metadata use the following command:

`amadmin --runasdn` *userdn* `--password` *password* `--data` *proprietary_metadata_filename*

After loading the metadata, the `--export` option can be used to export metadata compliant with the Liberty ID-FF. This file can then be exchanged with trusted partners. Here is an example of an identity provider metadata XML file for proprietary attributes.

EXAMPLE 7–2 Identity Provider Proprietary Metadata XML File for `amadmin`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Requests PUBLIC "-//iPlanet//Sun Access Manager 2005Q4 Admin CLI
DTD//EN"    "jar://com/iplanet/am/admin/cli/amAdmin.dtd">
<Requests>
   <OrganizationRequests DN="dc=companyA,dc=com">
```

EXAMPLE 7–2 Identity Provider Proprietary Metadata XML File for `amadmin` *(Continued)*

```
      <CreateHostedProvider id="http://sp.companyA.com" role="SP"
       defaultUrlPrefix="http://sp.companyA.com:80">
          <AttributeValuePair>
              <Attribute name="iplanet-am-provider-name"/>
              <Value>sp</Value>
          </AttributeValuePair>
          <AttributeValuePair>
              <Attribute name="iplanet-am-provider-alias"/>
              <Value>sp.companyA.com</Value>
          </AttributeValuePair>
          <AttributeValuePair>
              <Attribute name="iplanet-am-list-of-authenticationdomains"/>
              <Value>samplecot</Value>
          </AttributeValuePair>
          <AttributeValuePair>
              <Attribute name="iplanet-am-certificate-alias"/>
              <Value>cert_alias</Value>
          </AttributeValuePair>
          <AttributeValuePair>
              <Attribute name="iplanet-am-trusted-providers"/>
              <Value>http://idp.companyB.com</Value>
              <Value>http://idp.companyC.com</Value>
          </AttributeValuePair>
          <SPAuthContextInfo AuthContext="Password" AuthLevel="1"/>
          <AttributeValuePair>
              <Attribute name="iplanet-am-provider-homepage-url"/>
              <Value>http://sp.companyA.com:80/idff/index.jsp</Value>
          </AttributeValuePair>
      </CreateHostedProvider>
  </OrganizationRequests>
</Requests>
```

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. (The members of the domain must have previously established a *circle of trust* based on the Liberty Alliance Project architecture and operational agreements.)

Note – An authentication domain is *not* a domain in the domain name system (DNS) sense of the word.

The following procedures describe how to create, configure, and delete authentication domains using the Federated Access Manager Console.

- **Broken Link (Target ID: FXIYE)**
- **Broken Link (Target ID: FXIXJ)**
- **Broken Link (Target ID: FXJCG)**

▼ To Create An Authentication Domain

1 **In the Federated Access Manager Console, click the Federation tab.**

2 **Under Federation, select the Authentication Domains tab.**

3 **Select New.**

The New Authentication Domain attributes are displayed.

4 **Type a name for the authentication domain.**

5 **(Optional) Type a description of the authentication domain in the Description field.**

6 **(Optional) Type a value for the Writer Service URL.**

The 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`. For more information about the Common Domain Services, see **Broken Link (Target ID: ADMGF)**.

7 **(Optional) Type a value for the Reader Service URL.**

The 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`. For more information about the Common Domain Services, see **Broken Link (Target ID: ADMGF)**.

8 **Select Active or Inactive.**

The default status is Active. Selecting Inactive disables communication within the authentication domain.

9 **Click OK.**

The new authentication domain is now displayed in the list of configured Authentication Domains.

▼ To Configure or Modify an Authentication Domain

1 **In the Federated Access Manager Console, click the Federation tab.**

2 **Under Federation, select the Authentication Domains tab.**

All created Authentication Domains are displayed.

3 **Click the name of the authentication domain that you want to modify.**

The General and Providers properties for the authentication domain are displayed.

4 **(Optional) Enter or modify a description of the authentication domain in the Description field.**

5 **(Optional) Enter or modify the value for the Writer Service URL.**

The 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`. For more information on the Common Domain Services, see **Broken Link (Target ID: ADMGF)**.

6 **(Optional) Enter or modify the value for the Reader Service URL.**

The 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`. For more information on the Common Domain Services, see **Broken Link (Target ID: ADMGF)**.

7 **Select Active or Inactive.**

The default status is Active. Selecting Inactive disables communication within the authentication domain.

8 **Click Add to populate the authentication domain with providers.**

The Trusted Providers page is displayed.

9 **Choose from the list of Available Providers and click Add.**

10 **Click OK to save the providers to the authentication domain.**

The authentication domain's attribute page is displayed.

11 **Click Save to complete the configuration.**

▼ To Delete an Authentication Domain

Deleting an authentication domain does not delete the providers that belong to it although it will impact the trusted relationship.

1 **In the Federated Access Manager Console, click the Federation tab.**

2 **Under Federation, select the Authentication Domains tab.**

All created Authentication Domains are displayed.

3 **Select the check box next to the authentication domain that you want to delete.**

4 **Click Delete.**

The Pre-login URL

The pre-login process is the entry point for applications participating in Liberty-based single sign-on. The principal would be redirected to the location defined by the pre-login URL if no Federated Access Manager session token is found. This default process, though, can be modified based on the values of query parameters passed to Federated Access Manager by the service provider via a URL.

A *query parameter* is a name/value pair appended to the end of a URL. The parameter starts with a question mark (?) and takes the form *name*=*value*. A number of parameters can be combined in one URL; when more than one parameter exists, they are separated by an ampersand (&). Use the format `http://`*hostname*`:`*port*`/`*deploy-uri*`/preLogin?metaAlias=`*metaAlias*. Additional parameters are appended to the URL as &*param1*=*value1*&*param2*=*value2* and so on. These parameters and their usage and values are described in the following table.

TABLE 7–1 Pre-login URL Parameters for Federation

Parameter	Description
`actionOnNoFedCookie`	The `actionOnNoFedCookie` parameter provides the flexibility to redirect a user when the `fedCookie` is not present in the browser, and when there is only one identity provider. It takes the following values: ■ `commonlogin` will redirect to a common login page. ■ `locallogin` will redirect to the local Federated Access Manager login page. ■ `passive` will issue a request to the identity provider by setting the `isPassive` parameter of the `AuthnRequest` element to `true`. ■ `active` will issue a normal single sign-on request to the identity provider.

TABLE 7–1 Pre-login URL Parameters for Federation *(Continued)*

Parameter	Description
`anonymousOnetime`	The `anonymousOnetime` parameter can be used by service providers that authenticate users with anonymous, one time federation sessions. A value of `true` enables the service provider to issue a one time federation request and generate an anonymous session after successful verification of the authentication assertion from the identity provider. This feature is useful when the service provider doesn't have a user repository (for example, `http://www.weather.com`) but would like to depend on an identity provider for authentication. When the service provider receives a successful authentication assertion from an identity provider, they would generate an anonymous, temporary session.
`authlevel`	The `authlevel` parameter takes as a value a positive number that maps to an authentication level defined in the Federated Access Manager Authentication Framework. The authentication level indicates how much to trust a method of authentication. In this framework, each service provider is configured with a default authentication context (preferred method of authentication). However, the provider might like to change the assigned authentication context to one that is based on the defined authentication level. For example, provider B would like to generate a local session with an authentication level of 3 so it requests the identity provider to authenticate the user with an authentication context assigned that level. The value of this query parameter determines the authentication context to be used by the identity provider.
`goto`	The `goto` parameter takes as a value a URL to which the principal will be redirected after a successful SSO. If the value is not specified, default redirection will occur based on the value of the Provider Home Page URL attribute defined in the service provider configuration. The value of this URL can be configured by changing the `iplanet-am-provider-homepage-url` attribute in the `amProviderConfig.xml` file.
`gotoOnFedCookieNo`	The `gotoOnFedCookieNo` parameter takes as a value a URL to which the principal is redirected if a `fedCookie` with a value of `no` is found. The default behavior is to redirect the user to the Federated Access Manager login page.

In order to modify the pre-login URL, edit the relevant properties in either the `AMConfig.properties` file or the `AMAgent.properties` file, dependant on your deployment. See the following procedures for more information:

- **Broken Link (Target ID: GDEPU)**
- **Broken Link (Target ID: GDEOW)**

▼ To Configure for Pre-login

In a federation setup, Federated Access Manager acts as a service provider and manages an application that runs on a separate instance of Sun Java System Web Server. You must configure the agent that is protecting this application as follows:

1 **Point the `com.sun.am.policy.loginURL` property in the `AMAgent.properties` file to the pre-login service URL running on Federated Access Manager.**

For example: `com.sun.am.policy.loginURL =`
`http://www.sp1.com:58080/amserver/preLogin?metaAlias=www.sp1.com`

2 **Point the `com.sun.am.policy.am.library.loginURL` in the `AMAgent.properties` file to the login URL of the instance of Federated Access Manager acting as the service provider.**

For example: `com.sun.am.policy.am.library.loginURL =`
`http://www.sp1.com:58080/amserver/UI/Login`

▼ To Configure for Global Logout

To implement the logout process for all service providers using the Liberty Logout method, do the following:

1 **Copy the `AMClient.properties` file to the service provider's web container.**

2 **Revise the Logout method, as follows:**

```
ResourceBundle rsbu =ResourceBundle.getBundle("AMClient");
String logouturl = rsbu.getString
("com.sun.identity.federation.client.samples.logoutURL");
response.sendRedirect(logouturl);
```

This revision is equivalent to a redirection to
`http://www.sp1.com:58080/amserver/liberty-logout?metaAlias=www.sp1.com`.

Federated Operations

This section contains procedures illustrating how to use Federated Access Manager to configure interactions based on the Federation protocol. They are:

- **Broken Link (Target ID: GAMXU)**
- **Broken Link (Target ID: GALOQ)**
- **Broken Link (Target ID: GEACO)**
- **Broken Link (Target ID: GDSWN)**
- **Broken Link (Target ID: GAMXG)**

Auto-Federation

The auto-federation feature in Federated Access 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 Federated Access 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 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

Federated Access Manager provides a script for federating user accounts in bulk. It is called `ambulkfed` and is located in `/opt/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, the script 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 Federated Access Manager. `ambulkfed` generates and loads the LDIF data into Federated Access Manager based on its given provider role. For example, it will load `spuserdata.ldif` if Federated Access Manager acts as a service provider and it will load `idpuserdata.ldif` if Federated Access Manager acts as an identity provider. The LDIFs will also be stored locally and can be used with `ldapmodify` to load the data into a remote instance of Federated Access Manager. If the remote provider is not an instance of Federated Access 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.

Configuring Trust Between Providers

In order to complete interactions based on the Liberty ID-FF, trust must exist between all communicating providers. Each provider that wishes to be part of a federated trust model does so after complex business negotiations, the exchange of provider configuration metadata, and the configuration of trust. Using the Federated Access Manager console, trusted providers are configured using the metadata and are then grouped (as *entities*) into an *authentication domain*. To accomplish this, you load the provider metadata, and assign the configured providers to the same authentication domain. The following procedure explains how to configure trust using either the command line interface or the Federated Access Manager console. Additional information can be found in **Broken Link (Target ID: ADMCY)**.

▼ To Configure Trust Between Service Providers and Identity Providers

Before You Begin You must have metadata files specific to each provider you are configuring. Access Manager includes sample metadata XML files that you can modify for your purposes.

1 **Load the hosted and remote provider metadata XML files to Federated Access Manager using the `amadmin` command line interface.**

See **Broken Link (Target ID: GDOHT)** for information.

2 **Login to the Federated Access Manager console as `amadmin`, the default administrator.**

3 **Under Federation, click the Authentication Domains tab.**

4 **Select New.**

The new Authentication Domain attributes are displayed.

5 **Create the authentication domain and click OK.**

See **Broken Link (Target ID: FXIYE)** for information.

6 **Under Federation, click the Entities tab.**

7 **Select the name of a provider.**

The provider was created when the metadata was loaded. The General attributes for the chosen provider are displayed.

8 **Select the appropriate provider type from the View pull down menu.**

9 **Scroll down to Authentication Domains, select the authentication domain just created and click Add.**

The authentication domain will be moved under Selected.

10 **Click Save to store the change.**

Repeat this configuration for all providers (remote and hosted) with which you want to establish trust.

11 **Under Federation, click the Authentication Domains tab.**

12 **Select the name of the authentication domain which was previously created.**

The General attributes are displayed.

13 **Under Providers, click Add.**

The Select Trusted Partner Type and Profile page is displayed.

14 **Select the appropriate provider(s) as trusted members of the authentication domain and click Add.**

The provider(s) will be moved under Selected.

15 Click OK to save the change.

16 Click Save to store the change.

Trust is now established between the appropriate providers.

Signing Liberty ID-FF Requests and Responses

Federation-based communications passing between identity providers and service providers are generally required to be digitally signed and verified. Signing and verifying messages provides data integrity, data origin authentication, and a basis for non-repudiation. To turn on signing for all Liberty ID-FF requests and responses emanating from your instance of Federated Access Manager, set the value of the `com.sun.identity.federation.services.signingOn` property in `AMConfig.properties` to `true` and restart Federated Access Manager and its web container. This allows for signing of Liberty ID-FF requests being sent and verification of signature validity for Liberty ID-FF responses received. If set to `false`, signing is disabled. If set to `optional`, requests and responses will be signed or verified only if required by the federation profile being used. After installation, `AMConfig.properties` is located in the `etc/opt/SUNWam/config` directory.

Note – More information on `com.sun.identity.federation.services.signingOn` and the other identity federation properties in `AMConfig.properties` can be found in the Chapter 6, “amConfig.properties Reference,” in *Sun Java System Access Manager 7.1 Administration Reference*.

Additionally, you can enable the signing of an authentication request from a service provider configured on your instance of Federated Access Manager, use the following procedure.

▼ To Enable Signing of Service Provider Authentication Requests

Before You Begin A keystore must be set up before turning on the signing properties. See **Broken Link (Target ID: GDWQH)** information on how to do this.

1 Log in to the Federated Access Manager console as the top-level administrator, by default, `amadmin`**.**

2 Select the Federation tab.

3 Select the Entities tab.

4 Select the name of the entity that contains the service provider configuration for which you want to enable the signing of an authentication request.

5 Select Service Provider from the View pull-down menu.

6 **Enable the Sign Authentication Request property under the Service Provider configuration and click Save.**

7 **Log out of the Federated Access Manager console.**

Dynamic Identity Provider Proxying

An identity provider that is asked to authenticate a principal that has already been authenticated with another identity provider may proxy the authentication request, on behalf of the requesting service provider, to the authenticating identity provider. This is called *dynamic identity provider proxying*. When the first identity provider receives an authentication request regarding a principal, it prepares a new authentication assertion on its own behalf by referencing the relevant information from the original assertion and sending the assertion to the authenticating identity provider.

Note – The service provider requesting authentication may control this proxy behavior by setting a list of preferred identity providers or by defining the amount of times the identity provider can proxy the request.

▼ To Configure and Test Dynamic Identity Provider Proxying

The following steps describe the procedure to enable three machines for identity provider proxying and test the configuration. The procedure assumes the three machines have Federated Access Manager installed and are configured as follows:

Machine	Authentication Function	Federation Function
Machine 1	Authenticating Identity Provider	Identity Provider
Machine 2	Proxying Identity Provider	Identity Provider and Service Provider
Machine 3	Requesting Service Provider	Service Provider

All of the WAR files and metadata used in the following procedure can be found in /*FederatedAccessManager-base*/`samples/liberty/sample1`.

1 **To configure machine 3, deploy the SP1 WAR files and load** `sp1Metadata.xml`**.**

Ensure that the metadata defines machine 2 as an identity provider and machine 3 as a service provider.

2 **To configure machine 1, deploy the IDP1 WAR files and load** `idp1Metadata.xml`**.**

Ensure that the metadata defines machine 1 as an identity provider and machine 2 as a service provider.

3 **To configure machine 2, do the following:**

a. **To configure machine 2 as a service provider, deploy the SP1 WAR files.**

   Modify `AMClient.properties` to reflect this.

b. **To configure machine 2 as an identity provider, load a second, modified `idp1Metadata.xml`.**

   Ensure that `idp1Metadata.xml` contains *only* data that defines machine 1 as an identity provider. Remove all other metadata.

4 **Log in to machine 2 and modify the following metadata:**

a. **Change the value of the Authentication Type attribute to Local.**

   This attribute can be found in the Federated Access Manager Configuration section of the entity describing machine 2 as a service provider.

b. **Add machine 1 and machine 3 to the list of Trusted Providers configured for machine 2.**

   This attribute can be found in the Trusted Provider section of the entity describing machine 2 as a service provider.

c. **Save the configuration.**

5 **Also on machine 2, modify the following metadata regarding machine 3.**

a. **Select the check box next to Enable Proxy Authentication.**

   This attribute can be found in the Proxy Authentication Configuration section of the entity that defines machine 3 as an identity provider.

b. **Add machine 1 to the list of Proxy Identity Providers List.**

   This attribute can be found in the Proxy Authentication Configuration section of the entity that defines machine 3 as an identity provider. The value is a URI defined as the provider's identifier.

c. **Set Maximum Number of Proxies to 1.**

d. **Save the configuration.**

6 **Federate a user between machine 3 (acting as a service provider) and machine 2 (acting as an identity provider).**

7 **Federate a user between machine 2 (acting as a service provider) and machine 1 (acting as an identity provider).**

8 Close the browser and attempt single sign-on.

You will be redirected to machine 1 rather than machine 2 if you enter the username and password used to federate with machine 1.

Common Domain Services for Federation Management

Service providers need a way to determine which identity provider is used by a principal requesting authentication. Because authentication domains are configured without regard to their location, this function must work across DNS-defined domains. Thus, a common domain is configured for this purpose. The *common domain* is established for use only within the scope of the Common Domain Services for Federation Management. In Federated Access Manager deployments, the Common Domain Services for Federation Management are deployed in a web container installed in a predetermined and pre-configured *common* domain so that the common domain cookie is accessible to all providers in the authentication domain. If the HTTP server in the common domain is operated by the service provider, the service provider will redirect the user agent to the appropriate identity provider.

Configuring the Common Domain Services for Federation Management URLs

In Federated Access Manager, the Common Domain Services for Federation Management are configured using two URLs that point to servlets developed for writing and reading the common domain cookie. They are:

- **Broken Link (Target ID: GAMXW)**
- **Broken Link (Target ID: GAMWS)**

Note – For more information on how to configure these URLs, see **Broken Link (Target ID: FXIYE)** in **Broken Link (Target ID: ADMCS)**.

Writer Service URL

The Writer Service URL is used by the identity provider. After successful authentication, the common domain cookie is appended with the query parameter `_liberty_idp=`*entity-ID-of-identity-provider*. This parameter is used to redirect the principal to the Writer Service URL defined for the identity provider. The URL is configured as the value for the Writer Service URL attribute when an authentication domain is created. Use the format `http://`*common-domain-host:port*`/`*deployment-uri*`/writer` where *common-domain-host:port* refers to the machine on which the Common Domain Services for Federation Management are installed and *deployment-uri* tells the web container where to look for information specific to the application (such as classes or JARs). The default URI is `amcommon`.

Reader Service URL

The Reader Service URL is used by the service provider. The service provider redirects the principal to this URL in order to find the preferred identity provider. Once found, the principal is redirected to the identity provider for single sign-on. The URL is defined as the value for the Reader Service URL attribute when an authentication domain is created. It is formatted as `http://`*common-domain-host:port*`/`*deployment-uri*`/transfer` where *common-domain-host:port* refers to the machine on which the Common Domain Services for Federation Management are installed and *deployment-uri* tells the web container where to look for information specific to the application (such as classes or JARs). The default URI is `amcommon`.

Configuring the Common Domain Services for Federation Management Properties

`FSIntroConfig.properties` is a file that contains properties used to configure the Common Domain Services for Federation Management. It is deployed as part of the web application and located in `/`*FederatedAccessManager-base*`/web-src/common/WEB-INF/classes`. It contains the properties described in the following table (which may be modified).

TABLE 7–2 Common Domain Services for Federation Management Properties in `FSConfig.properties`

Property	Description
`com.sun.identity.federation.services.introduction.cookiedomain`	The value of this property is the name of the common domain.
`com.sun.identity.federation.services.introduction.cookietype`	This property takes a value of either `PERSISTENT` or `SESSION`. `PERSISTENT` defines the cookie as one that will be stored and reused after a web browser is closed and reopened. `SESSION` defines the cookie as one that will not be stored after the web browser has been closed.
`com.iplanet.am.cookie.secure`	This property takes a value of either `false` or `true`. It defines whether the cookie needs to be secured or not.
`com.iplanet.am.cookie.encode`	This property takes a value of either `false` or `true`. It defines whether the cookie will be URL encoded or not. This property is useful if, for example, the web container that reads or writes the cookie decrypts or encrypts it by default.

CHAPTER 8

Web Services

Sun Java^{TM} System Federated Access Manager implements the Liberty Identity Web Services Framework (Liberty ID-WSF) which defines a web services stack that can be used to support the Liberty Alliance Project business model. These web services leverage the Liberty ID-FF for principal authentication, federation, and privacy protections.

Web services are distributed applications developed using open technologies such as eXtensible Markup Language (XML), SOAP, and HyperText Transfer Protocol (HTTP). Enterprises use these technologies as a mechanism for allowing their applications to cross network boundaries and communicate with those of their partners, customers and suppliers. Federated Access Manager implements the Liberty ID-WSF which is designed to operate in concert with a federated identity framework, such as the Liberty Identity Federation Framework (Liberty ID-FF). Federated includes the following Liberty ID-WSF web services:

Authentication Web Service

The Authentication Web Service adds authentication functionality to the SOAP binding. It provides authentication to a WSC, allowing the WSC to obtain security tokens for further interactions with other services at the same provider. These other services may include a discovery service or single sign-on service. Upon successful authentication, the final Simple Authentication and Security Layer (SASL) response contains the resource offering for the Discovery Service.

Caution – Do not confuse the Liberty-based Authentication Web Service with the proprietary Federated Access Manager Authentication Service discussed in the *Sun Java System Access Manager 7.1 Technical Overview*.

Authentication Web Service Attribute

The Authentication Web Service attribute is a *global* attribute. The value of this attribute is carried across theFederated Access Manager configuration and inherited by every organization.

Note – For information about the types of attributes used in Federated Access Manager, see the *Sun Java System Access Manager 7.1 Technical Overview*.

The attribute for the Authentication Web Service is defined in the `amAuthnSvc.xml` service file and is called the Mechanism Handlers List.

Mechanism Handlers List

The Mechanism Handler List attribute stores information about the SASL mechanisms that are supported by the Authentication Web Service.

`key` Parameter

The required `key` defines the SASL mechanism supported by the Authentication Web Service.

`class` Parameter

The required `class` specifies the name of the implemented class for the SASL mechanism. Two authentication mechanisms are supported by the following default implementations:

TABLE 8–1 Default Implementations for Authentication Mechanism

Class	Description
`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.

Note – The Authentication Web Service layer provides an interface that must be implemented for each SASL mechanism to process the requested message and return a response. For more information, see **Broken Link (Target ID: GALGQ)**.

Liberty Personal Profile Service

The Liberty Personal Profile Service is a data service that supports storing and modifying a principal's identity attributes. It maps attributes defined in a user's personal profile to LDAP attributes in a data store. These identity attributes might include the user's first name, last name, home address, or emergency contact information. The Liberty Personal Profile Service is queried or updated by a WSC acting on behalf of the principal. .

Liberty Personal Profile Service Attributes

The Liberty Personal Profile Service attributes are *global* attributes. The values of these attributes are carried across the Federated Access Manager configuration and inherited by each configured organization.

Note – For information about the types of attributes used in Federated Access Manager, see the *Sun Java System Access Manager 7.1 Technical Overview*.

Attributes for the Personal Profile Service are defined in the `amLibertyPersonalProfile.xml` service file. The attributes are:

- **Broken Link (Target ID: ADMHK)**
- **Broken Link (Target ID: ADMHL)**
- **Broken Link (Target ID: ADMHM)**
- **Broken Link (Target ID: ADMHN)**
- **Broken Link (Target ID: ADMHO)**
- **Broken Link (Target ID: ADMHP)**
- **Broken Link (Target ID: ADMHQ)**
- **Broken Link (Target ID: ADMHR)**
- **Broken Link (Target ID: ADMHS)**
- **Broken Link (Target ID: ADMHT)**
- **Broken Link (Target ID: ADMHU)**
- **Broken Link (Target ID: ADMHV)**
- **Broken Link (Target ID: ADMHW)**
- **Broken Link (Target ID: ADMHX)**
- **Broken Link (Target ID: ADMHY)**
- **Broken Link (Target ID: ADMHZ)**

ResourceID Mapper

The value of this attribute specifies the implementation of `com.sun.identity.liberty.ws.interfaces.ResourceIDMapper`. Although a new implementation can be developed, Federated Access Manager provides the default `com.sun.identity.liberty.ws.idpp.plugin.IDPPResourceIDMapper`, which maps a discovery resource identifier to a user identifier.

Authorizer

Before processing a request, the Liberty Personal Profile Service verifies the authorization of the WSC making the request. There are two levels of authorization verification:

- Is the requesting entity authorized to access the requested resource profile information?
- Is the requested resource published to the requestor?

Authorization occurs through 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, Federated Access 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 this:

```
/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, which are also defined in the Liberty Personal Profile Service:

- **Broken Link (Target ID: ADMHS)**
- **Broken Link (Target ID: ADMHT)**

Attribute Mapper

The value of this attribute defines the class for mapping a Liberty Personal Profile Service attribute to an Federated Access Manager user attribute. By default, the class is `com.sun.identity.liberty.ws.idpp.plugin.IDPPAttributeMapper`.

Note – `com.sun.identity.liberty.ws.idpp.plugin.IDPPAttributeMapper` is not a public class.

Provider ID

The value of this attribute defines the unique identifier for this instance of the Liberty Personal Profile Service. Use the format *protocol*`://`*hostname*`:`*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 that is 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.

Note – The term *container* as described in this section is not related to the Federated Access Manager identity-related object that is also called *container*.

For example, Emergency Contact and Common Name are two default containers for the Liberty Personal Profile Service. To add a new container, click Add, enter values in the provided fields and click OK.

Note – This functionality is not yet public.

PPLDAP Attribute Map List

Each identity attribute defined in the Liberty Personal Profile Service maps one-to-one with a Federated Access Manager LDAP attribute. For example, `JobTitle=sunIdentityServerPPEmploymentIdentityJobTitle` maps the Liberty `JobTitle` attribute to the Federated Access Manager `sunIdentityServerPPEmploymentIdentityJobTitle` attribute.

The value of this attribute is a list that specifies the mappings. The list is used by the attribute mapper defined in **Broken Link (Target ID: ADMHM)**, by default, `com.sun.identity.liberty.ws.idpp.plugin.IDPPAttributeMapper`.

Note – When adding new attributes to the Liberty Personal Profile Service or the LDAP data store, ensure that the new attribute mappings are configured as values of this attribute.

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.

```
<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>
          </DefaultValues>
</AttributeSchema>
```

Require Query PolicyEval

If selected, this option requires that a policy evaluation be performed for Liberty Personal Profile Service queries. For more information, see **Broken Link (Target ID: ADMHL)**.

Require Modify PolicyEval

If selected, this option requires that a policy evaluation be performed for Liberty Personal Profile Service modifications. For more information, see **Broken Link (Target ID: ADMHL)**.

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 8–1 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"
```

EXAMPLE 8–1 Extension Query for `creditcard` *(Continued)*

```
  xmlns="http://www.sun.com/identity/liberty/pp"
  targetNamespace="http://www.sun.com/identity/liberty/pp">
  <xs:annotation>
      <xs:documentation>
      </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>
```

Type the new attribute and click Add.

Extension Attributes Namespace Prefix

The value of this attribute specifies the namespace prefix for the extensions defined in the **Broken Link (Target ID: ADMHU)**. This prefix is prepended to the element and helps to distinguish metadata from different XML schema namespaces.

Service Update

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 about this functionality and the mechanisms, see the *Liberty ID-WSF Security Mechanisms* specification.

Discovery Service

The Discovery Service is a framework for describing and discovering identity web services. It allows a requesting entity, such as a service provider, to dynamically determine a principal's registered web services provider (WSP), such as an attribute provider. Typically, a service provider queries the Discovery Service, which responds by providing a resource offering that describes the requested WSP. (A *resource offering* defines associations between a piece of identity data and the service instance that provides access to the data.) The implementation of the Discovery Service includes Java and web-based interfaces. The service is bootstrapped using Liberty ID-FF single sign-on or the Liberty ID-WSF Authentication Web Service. .

Note – By definition, a discoverable service is assigned a service type URI, allowing the service to be registered in Discovery Service instances. The service type URI is typically defined in the Web Service Definition Language (WSDL) file that defines the service.

Discovery Service Attributes

The Discovery Service attributes are global attributes whose values are applied across the Federated Access Manager configuration and inherited by every configured organization. The Discovery Service attributes are:

- **Broken Link (Target ID: ADMIQ)**
- **Broken Link (Target ID: ADMIR)**
- **Broken Link (Target ID: ADMIS)**
- **Broken Link (Target ID: ADMIT)**
- **Broken Link (Target ID: ADMIU)**
- **Broken Link (Target ID: ADMIV)**
- **Broken Link (Target ID: ADMIW)**
- **Broken Link (Target ID: ADMIX)**
- **Broken Link (Target ID: ADMIY)**
- **Broken Link (Target ID: ADMIZ)**
- **Broken Link (Target ID: ADMJA)**
- **Broken Link (Target ID: ADMJB)**
- **Broken Link (Target ID: ADMJC)**

Note – For information about the types of attributes used in Federated Access Manager, see the *Sun Java System Access Manager 7.1 Technical Overview*.

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 only if other relevant attributes values are changed to match the new pointer.

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 are selected. If an authentication method is not selected and a WSC sends a request using that method, the request is rejected. For more information, see the Liberty ID-WSF Security Mechanisms specification.

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 describes the available options. More information can be found in the Liberty ID-WSF Discovery Service Specification.

TABLE 8–2 Policy-Related Directives

Directive	Purpose
`AuthenticateRequester`	The Discovery Service should include a SAML assertion containing an `AuthenticationStatement` in its query 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 query responses that indicate the status of the session.
`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.

TABLE 8–2 Policy-Related Directives (Continued)

Directive	Purpose
`GenerateBearerToken`	For use with Bearer Token Authentication, the Discovery Service should generate a token that grants the bearer permission to access the resource.

Policy Evaluation for Discovery Lookup

If enabled, the service will perform a policy evaluation for the `DiscoveryLookup` operation. By default, the check box is not selected.

Policy Evaluation for Discovery Update

If enabled, the service will perform a policy evaluation for the `DiscoveryUpdate` operation. By default, the check box is not selected.

`Authorizer` Plug-in 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 WSC. The default class is `com.sun.identity.liberty.ws.disco.plugins.DefaultDiscoAuthorizer`. See **Broken Link (Target ID: ADMJI)**.

Entry Handler Plug-in 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. This interface is used to set or retrieve a principal's discovery entries. To handle discovery entries differently, implement the `com.sun.identity.liberty.ws.disco.plugins.DiscoEntryHandler` 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`. See **Broken Link (Target ID: ADMJK)**.

Classes For `ResourceIDMapper` Plug-in

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` assumes the format to be *providerID* + "/" + *the Base64 encoded userIDs*

- `com.sun.identity.liberty.ws.disco.plugins.DefaultHexResourceIDMapper` assumes the format to be *providerID* + "/" + *the hex string of userIDs*

Different implementations may also be developed with the interface and added as a value of this attribute by clicking New and defining the following attributes:

- *Provider ID* takes as a value a URI that points to the Discovery Service. Use the format `http://`*host*`:`*port*`/amserver/Liberty/disco`. See **Broken Link (Target ID: ADMIQ)**.
- *ID Mapper* takes as a value the class name and path of the implementing class.

See **Broken Link (Target ID: ADMJJ)**.

Authenticate Response Message

If enabled, the service authenticates the response message. By default, the function is not enabled.

`SessionContextStatement` for Bootstrapping

If enabled, this attribute specifies whether to generate a `SessionContextStatement` for bootstrapping. A `SessionContextStatement` conveys the session status of an entity. By default, this function is not enabled.

Encrypt `NameIdentifier` in Session Context for Bootstrapping

If enabled, the service encrypts the name identifier in a `SessionContextStatement`. By default, this function is not enabled.

Implied Resource

If enabled, the service does not generate a resource identifier for bootstrapping. By default, this function 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. The value of the Resource Offerings for Bootstrapping attribute is a default value configured during installation. If you want to define a new resource offering, you must first delete the existing resource offering, then click New to define the attributes for a new resource offering. If you want to edit an existing resource offering, click the name of the existing Service Type to modify the attributes. For more information, see **Broken Link (Target ID: ADMJG)**.

Storing Resource Offerings

A *resource offering* defines an association between a type of identity data and a URI to the WSDL file that provides information about obtaining access to the data. In Federated Access Manager, a resource offering can be stored as a user attribute or as a dynamic attribute. Storing resource offerings within a user profile supports both `DiscoveryLookup` and `DiscoveryUpdate` operations. Storing resource offerings within a service and assigning that service to a realm or role supports only the `DiscoveryLookup` operation using the discovery protocol. (Updates can still be done using the Federated Access Manager Console.) More information is provided in the following sections:

- **Broken Link (Target ID: ADMJE)**
- **Broken Link (Target ID: ADMJF)**
- **Broken Link (Target ID: ADMJG)**

Storing Resource Offerings as User Attributes

Resource offerings can be stored as an attribute under a user's profile using the Lightweight Directory Access Protocol (LDAP). Storing resource offerings within a user profile supports both `DiscoveryLookup` and `DiscoveryUpdate` operations. The following procedure explains how to access and create a user's resource offerings.

▼ To Store a Resource Offering as a User Attribute

1. **In the Federated Access Manager Console, click the Access Control tab.**
2. **Select the name of the realm that contains the user profile you want to modify.**
3. **Select Subjects to access user information.**
4. **Select the name of the user profile that you want to modify.**
5. **Select Services to access the user's services.**
6. **Click Add to configure the Discovery Service for this user.**
7. **Select Discovery Service and click Next.**

   The Discovery Service is added to the user's services.
8. **Select General to access the user's User Discovery Resource Offering attribute.**
9. **Click Edit.**

   A User Discovery Resource Offering window opens.

10 Click Add in the User Discovery Resource Offering window.

11 (Optional) Type a value for the Resource ID Attribute.

This field defines an identifier for the resource offering.

12 Type the Resource ID Value.

This field defines the resource identifier. A *resource identifier* is a URI registered with the Discovery Service that point to a particular discovery resource. It is generated by the profile provider. The value of this attribute must not be a relative URI and should contain a domain name that is owned by the provider hosting the resource. If a discovery resource is exposed in multiple Resource Offerings, the Resource ID Value for all of those resource offerings would be the same. An example of a valid Resource ID value is `http://profile-provider.com/profiles/14m0B82k15csaUxs`.

Tip – `urn:libery:isf:implied-resource` can be used as a Resource ID Value when only one resource can be operated upon at the service instance being contacted. The URI only implicitly identifies the resource in question. In some circumstances, the use of this resource identifier can eliminate the need for contacting the discovery service to access the resource.

---

13 (Optional) Enter a description of the resource offering in the Description field.

14 Type a URI for the value of the Service Type attribute.

This URI defines the type of service.

Tip – It is recommended that the value of this attribute be the `targetNamespace` URI defined in the abstract WSDL description for the service. An example of a valid URI is `urn:liberty:id-sis-pp:2003-08`.

---

15 Type a URI for the value of the Provider ID attribute.

This attribute contains the URI of the provider of the service instance. This information is useful for resolving trust metadata needed to invoke the service instance. A single physical provider may have multiple provider IDs. An example of a valid URI is `http://profile-provider.com`.

Note – The provider represented by the URI in the Provider ID attribute must also have a class entry in the `ResourceIDMapper` attribute. For more information, see **Broken Link (Target ID: ADMIX)**.

---

16 Click Add Description to define the Service Description.

For each resource offering, at least one service description must be created.

a. **Select the values for the Security Mechanism ID attribute to define how a web service client can authenticate to a web service provider.**

   This field lists the security mechanisms that the service instance supports. Select the security mechanisms that you want to add and click Add. To prioritize the list, select the mechanism and click Move Up or Move Down.

b. **Type a value for the End Point URL.**

   This value is the URL of the SOAP-over-HTTP endpoint. The URI scheme must be HTTP or HTTPS as in `https://soap.profile-provider.com/soap`.

c. **(Optional) Type a value for the SOAP Action.**

   This value is the equivalent of the `wsdlsoap:soapAction` attribute of the `wsdlsoap:operation` element in the service's concrete WSDL-based description.

d. **Click OK to complete the configuration.**

17 Check the Options box if there are no options or add a URI to specify options for the resource offering.

This field lists the options that are available for the resource offering. Options provide hints to a potential requestor about the availability of certain data or operations to a particular offering. The set of possible URIs are defined by the service type, not the Discovery Service. If no option is specified, the service instance does not display any available options. For a standard set of options, see the *Liberty ID-SIS Personal Profile Service Specification*.

18 Select a directive for the resource offering.

Directives are special entries defined in SOAP headers that can be used to enforce policy-related decisions. You can choose from the following:

- `GenerateBearerToken` specifies that a bearer token be generated.
- `AuthenticateRequester` must be used with any service description that use SAML for message authentication.
- `EncryptResourceID` specifies that the Discovery Service encrypt the resource ID.
- `AuthenticateSessionContext` is specified when a Discovery Service provider includes a SAML assertion containing a `SessionContextStatement` in any future `QueryResponse` messages.
- `AuthorizeRequester` is specified when a Discovery Service provider wants to include a SAML assertion containing a `ResourceAccessStatement` in any future `QueryResponse` messages.

If you want to associate a directive with one or more service descriptions, select the check box for that Description ID. If no service descriptions are selected, the directive is applied to all description elements in the resource offering.

19 **Click OK.**

20 **Click Close to close the User Discovery Resource Offering window.**

21 **Click Save to save the configuration.**

Storing Resource Offerings as Dynamic Attributes

Due to the repetition inherent in storing discovery entries as user attributes, Federated Access Manager has established the option of storing a discovery entry as a dynamic attribute within a role or a realm. The role or realm can then be assigned to an identity-related object, making the entry available to all users within the object. Unlike storing a discovery entry as a user attribute, this scenario only supports the `DiscoveryLookup` operation.

There are two ways in which you can store discovery entries as dynamic attributes. You can store them in a realm or in a role. The following sections describe the procedures:

- **Broken Link (Target ID: FXIZS)**
- **Broken Link (Target ID: GAVTS)**

▼ To Store Resource Offerings as Dynamic Attributes in a Realm

To create a discovery entry as a dynamic attribute in a realm, the Discovery Service must first be added and a template created.

1 **In the Federated Access Manager Console, click the Access Control tab.**

2 **Select the name of the realm you want to modify.**

3 **Select Services to access the realm's services.**

4 **Click Add to add the Discovery Service to the realm.**

A list of available services is displayed.

5 **Select Discovery Service and click Next to add the service.**

A list of added services is displayed including the Discovery Service.

6 **Select Subjects to access user information.**

7 **Select the name of the user you want to modify.**

8 **Select Services to add the Discovery Service to the user.**

9 **Click Add to add the Discovery Service to the user.**

A list of available services is displayed.

10 **Select Discovery Service and click Next to add the service.**

A list of added services is displayed including the Discovery Service.

11 **Using the path displayed on top of the Federated Access Manager Console, click the name of the realm.**

12 **Click Services to access the realm's services.**

13 **Click Add.**

14 **Select Discovery Service and click Next to add the service.**

15 **Click Discovery Service to add a resource offering to the service.**

16 **Click Add to add a resource offering.**

17 **(Optional) Enter a description of the resource offering in the Description field.**

18 **Type a URI for the value of the Service Type attribute.**

This URI defines the type of service. It is *recommended* that the value of this attribute be the `targetNamespace` URI defined in the *abstract* WSDL description for the service. An example of a valid URI is `urn:liberty:id-sis-pp:2003-08`.

19 **Type a URI for the value of the Provider ID attribute.**

The value of this attribute contains the URI of the provider of the service instance. This information is useful for resolving trust metadata needed to invoke the service instance. A single physical provider may have multiple provider IDs. An example of a valid URI is `http://profile-provider.com`.

Note – The provider represented by the URI in the Provider ID attribute must also have an entry in the `ResourceIDMapper` attribute. For more information, see **Broken Link (Target ID: ADMIX)**.

---

20 Click Add Description to define the Service Description.

For each resource offering, at least one service description must be created.

a. **Select the values for the Security Mechanism ID attribute to define how a web service client can authenticate to a web service provider.**

   This field lists the security mechanisms that the service instance supports. Select the security mechanisms that you want to add and click Add. To prioritize the list, select the mechanism and click Move Up or Move Down.

b. **Type a value for the End Point URL.**

   This value is the URL of the SOAP-over-HTTP endpoint. The URI scheme must be HTTP or HTTPS as in `https://soap.profile-provider.com/soap`.

c. **(Optional) Type a value for the SOAP Action.**

   This value is the equivalent of the `wsdlsoap:soapAction` attribute of the `wsdlsoap:operation` element in the service's concrete WSDL-based description.

d. **Click OK to complete the configuration.**

21 Check the Options box if there are no options or add a URI to specify options for the resource offering.

This field lists the options that are available for the resource offering. Options provide hints to a potential requestor about the availability of certain data or operations to a particular offering. The set of possible URIs are defined by the service type, not the Discovery Service. If no option is specified, the service instance does not display any available options. For a standard set of options, see the *Liberty ID-SIS Personal Profile Service Specification*.

22 Select a directive for the resource offering.

Directives are special entries defined in SOAP headers that can be used to enforce policy-related decisions. You can choose from the following:

- `GenerateBearerToken` specifies that a bearer token be generated.
- `AuthenticateRequester` must be used with any service description that use SAML for message authentication.
- `EncryptResourceID` specifies that the Discovery Service encrypt the resource ID.
- `AuthenticateSessionContext` is specified when a Discovery Service provider includes a SAML assertion containing a `SessionContextStatement` in any future `QueryResponse` messages.
- `AuthorizeRequester` is specified when a Discovery Service provider wants to include a SAML assertion containing a `ResourceAccessStatement` in any future `QueryResponse` messages.

If you want to associate a directive with one or more service descriptions, select the check box for that Description ID. If no service descriptions are selected, the directive is applied to all description elements in the resource offering.

23 **Click OK.**

24 **Click Close to close the Discovery Resource Offering window.**

25 **Click Save to save the configuration.**

▼ To Store Resource Offerings as Dynamic Attributes in a Role

To create a discovery entry as a dynamic attribute in a role, the Discovery Service must first be added and a template created.

1 **In the Federated Access Manager Console, click the Access Control tab.**

2 **Select the name of the realm you want to modify.**

3 **Select Subjects to access the realm's identity information.**

4 **Select Role to access the realm's role information.**

5 **Select the name of the role you want to modify.**

Alternately, you can create a new role and then select the name of this new role.

6 **Under Services, click Add to add the Discovery Service to the role.**

A list of available services is displayed.

7 **Select Discovery Service and click Next to add the service.**

A list of added services is displayed including the Discovery Service.

8 **Click Discovery Service to add a resource offering to the service.**

9 **Click Add.**

10 **(Optional) Enter a description of the resource offering in the Description field.**

11 **Type a URI for the value of the Service Type attribute.**

This URI defines the type of service. It is *recommended* that the value of this attribute be the `targetNamespace` URI defined in the *abstract* WSDL description for the service. An example of a valid URI is `urn:liberty:id-sis-pp:2003-08`.

12 Type a URI for the value of the Provider ID attribute.

The value of this attribute contains the URI of the provider of the service instance. This information is useful for resolving trust metadata needed to invoke the service instance. A single physical provider may have multiple provider IDs. An example of a valid URI is `http://profile-provider.com`.

Note – The provider represented by the URI in the Provider ID attribute must also have an entry in the `ResourceIDMapper` attribute. For more information, see **Broken Link (Target ID: ADMIX)**.

---

13 Click Add Description to define the Service Description.

For each resource offering, at least one service description must be created.

a. Select the values for the Security Mechanism ID attribute to define how a web service client can authenticate to a web service provider.

This field lists the security mechanisms that the service instance supports. Select the security mechanisms that you want to add and click Add. To prioritize the list, select the mechanism and click Move Up or Move Down.

b. Type a value for the End Point URL.

This value is the URL of the SOAP-over-HTTP endpoint. The URI scheme must be HTTP or HTTPS as in `https://soap.profile-provider.com/soap`.

c. (Optional) Type a value for the SOAP Action.

This value is the equivalent of the `wsdlsoap:soapAction` attribute of the `wsdlsoap:operation` element in the service's concrete WSDL-based description.

d. Click OK to complete the configuration.

14 Check the Options box if there are no options or add a URI to specify options for the resource offering.

This field lists the options that are available for the resource offering. Options provide hints to a potential requestor about the availability of certain data or operations to a particular offering. The set of possible URIs are defined by the service type, not the Discovery Service. If no option is specified, the service instance does not display any available options. For a standard set of options, see the *Liberty ID-SIS Personal Profile Service Specification*.

15 Select a directive for the resource offering.

Directives are special entries defined in SOAP headers that can be used to enforce policy-related decisions. You can choose from the following:

- `GenerateBearerToken` specifies that a bearer token be generated.

- `AuthenticateRequester` must be used with any service description that use SAML for message authentication.
- `EncryptResourceID` specifies that the Discovery Service encrypt the resource ID.
- `AuthenticateSessionContext` is specified when a Discovery Service provider includes a SAML assertion containing a `SessionContextStatement` in any future `QueryResponse` messages.
- `AuthorizeRequester` is specified when a Discovery Service provider wants to include a SAML assertion containing a `ResourceAccessStatement` in any future `QueryResponse` messages.

If you want to associate a directive with one or more service descriptions, select the check box for that Description ID. If no service descriptions are selected, the directive is applied to all description elements in the resource offering.

16 **Click OK.**

17 **Click Close to close the Discovery Resource Offering window.**

18 **Click Save to save the configuration.**

Storing a Resource Offering for Discovery Service Bootstrapping

Before a WSC can contact the Discovery Service to obtain a resource offering, the WSC needs to discover the Discovery Service. Thus, an initial resource offering for locating the Discovery Service is sent back to the WSC in a SAML assertion generated during authentication. The following procedure describes how to configure a global attribute for bootstrapping the Discovery Service. For more information on bootstrapping the Discovery Service, see **Broken Link (Target ID: ADMJC)**.

▼ To Store a Resource Offering for Discovery Service Bootstrapping

1 **In the Federated Access Manager Console, select the Web Services tab.**

2 **Under Web Services, click the Discovery Service tab.**

3 **Choose New under the Resource Offerings for Bootstrapping Resources attribute.**

By default, the resource offering for bootstrapping the Discovery Service is already configured. In order to create a new resource offering, you must first delete the default resource offering.

4 **(Optional) Type a description of the resource offering.**

5 Enter a URI for the value of the Service Type attribute.

This field defines the type of service. It is recommended that the value of this attribute be the `targetNamespace` URI defined in the *abstract* WSDL description for the service. An example of a valid URI is `urn:liberty:disco:2003-08`.

6 Enter a URI for the value of the Provider ID attribute.

This attribute contains the URI of the provider of the service instance. This is useful for resolving trust metadata needed to invoke the service instance. A single physical provider may have multiple provider IDs. An example of a valid URI is `http://sample_disco.com`.

Note – The provider represented by the URI in the Provider ID attribute must also have an entry in the Classes for ResourceIDMapper Plugin attribute. For more information, see **Broken Link (Target ID: ADMIX)**.

---

7 Click Add Description to define a security mechanism ID.

For each resource offering, at least one service description must be created.

a. Select the values for the Security Mechanism ID attribute to define how a web service client can authenticate to a web service provider.

This field lists the security mechanisms that the service instance supports. Select the security mechanisms you wish to add and click the Add button. To arrange the priority of the list, select the mechanism and use the Move Up or Move Down buttons.

b. Type a value for the End Point URL.

This value is the URL of the SOAP-over-HTTP endpoint. The URI scheme must be HTTP or HTTPS as in `https://soap.profile-provider.com/soap`.

c. (Optional) Type a value for the SOAP action.

This field contains the equivalent of the `wsdlsoap:soapAction` attribute of the `wsdlsoap:operation` element in the service's concrete WSDL-based description.

d. Click OK to save the configuration.

8 Check the Options box if there are no options or add a URI to specify options for the resource offering.

This field lists the options that are available for the resource offering. Options provide hints to a potential requestor about the availability of certain data or operations to a particular offering. The set of possible URIs are defined by the service type, not the Discovery Service. If no option is specified, the service instance does not display any available options. For a standard set of options, see the *Liberty ID-SIS Personal Profile Service Specification*.

9 **Select a directive for the resource offering.**

Directives are special entries defined in SOAP headers that can be used to enforce policy-related decisions. You can choose from the following:

- `GenerateBearerToken` specifies that a bearer token be generated.
- `AuthenticateRequester` must be used with any service description that use SAML for message authentication.
- `EncryptResourceID` specifies that the Discovery Service encrypt the resource ID.
- `AuthenticateSessionContext` is specified when a Discovery Service provider includes a SAML assertion containing a `SessionContextStatement` in any future `QueryResponse` messages.
- `AuthorizeRequester` is specified when a Discovery Service provider wants to include a SAML assertion containing a `ResourceAccessStatement` in any future `QueryResponse` messages.

If you want to associate a directive with one or more service descriptions, select the check box for that Description ID. If no service descriptions are selected, the directive is applied to all description elements in the resource offering.

10 **Click OK to complete the configuration.**

SOAP Binding Service

The SOAP Binding Service is the method of transport used to convey identity data between web services. It includes a set of Java APIs used by the developer of a Liberty-enabled identity service. The APIs are used to send and receive identity-based messages using SOAP, an XML-based messaging protocol. The service invokes the correct request handler class (specified by a service endpoint) to handle the messages.

SOAP Binding Service Attributes

The SOAP Binding Service attributes are *global* attributes. The values of these attributes are carried across the Federated Access Manager configuration and inherited by every organization.

Attributes for the SOAP Binding Service are defined in the `amSOAPBinding.xml` service file. The SOAP Binding Service attributes are as follows:

- **Broken Link (Target ID: ADMJT)**
- **Broken Link (Target ID: ADMJW)**
- **Broken Link (Target ID: ADMJX)**

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. The interface must be implemented on the server side for each Liberty-based web service that uses the SOAP Binding Service.

To add a new implementation, click New and define values for the following parameters.

`Key` Parameter

The `Key` parameter is the last part of the URI path to a SOAP endpoint. The SOAP endpoint in Federated Federated Access Manager is the `SOAPReceiver` servlet. The URI to the `SOAPReceiver` uses the format `protocol://`*host*`:`*port*`/`*deloy-uri*`/Liberty/`*key*. If you define `disco` as the `Key`, the URI path to the `SOAPReceiver` for the corresponding Discovery Service would be *protocol*`://`*host*`:`*port*`/amserver/Liberty/disco`.

Note – Different service clients must use different keys when connecting to the `SOAPReceiver`.

`Class` Parameter

The `Class` parameter specifies the name of the class implemented from `com.sun.identity.liberty.ws.soapbinding.RequestHandler` for the particular web service. For example, `class=com.example.identity.liberty.ws.disco.DiscoveryService`.

`SOAP Action` Parameter

The optional `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.

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 WSC.

Note – This interface is not public. The value of the attribute is configured during installation.

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 a mechanism is not selected and a WSC sends a request using it, the request is rejected. Following is a list of the supported authentication mechanisms:

- `urn:liberty:security:2003-08:ClientTLS:SAML`
- `urn:liberty:security:2003-08:ClientTLS:X509`
- `urn:liberty:security:2003-08:ClientTLS:null`
- `urn:liberty:security:2003-08:TLS:SAML`
- `urn:liberty:security:2003-08:TLS:X509`
- `urn:liberty:security:2003-08:TLS:null`
- `urn:liberty:security:2003-08:null:SAML`
- `urn:liberty:security:2003-08:null:X509`
- `urn:liberty:security:2003-08:null:null`
- `urn:liberty:security:2004-04:ClientTLS:Bearer`
- `urn:liberty:security:2004-04:TLS:Bearer`
- `urn:liberty:security:2004-04:null:Bearer`
- `urn:liberty:security:2005-02:ClientTLS:Bearer`
- `urn:liberty:security:2005-02:ClientTLS:SAML`
- `urn:liberty:security:2005-02:ClientTLS:X509`
- `urn:liberty:security:2005-02:TLS:Bearer`
- `urn:liberty:security:2005-02:TLS:SAML`
- `urn:liberty:security:2005-02:TLS:X509`
- `urn:liberty:security:2005-02:null:Bearer`
- `urn:liberty:security:2005-02:null:SAML`
- `urn:liberty:security:2005-02:null:X509`

Note – For more complete information about authentication mechanisms and their level of security, see the *Liberty ID-WSF Security Mechanisms* specification.

◆ ◆ ◆ CHAPTER 9

SAML Administration

Sun Java™ System Federated Access Manager uses the Security Assertion Markup Language (SAML) as the means for exchanging security information. SAML uses an eXtensible Markup Language (XML) framework to achieve interoperability between vendor platforms that provide SAML assertions. This chapter explains SAML and defines how it is used within Federated Access Manager. It covers the following topics:

SAML Attributes

The SAML module is configured by applying values to its attributes. `amSAML.xml` is the XML service file that defines the attributes. All SAML attributes are *global* in that the values applied to them are carried across the Federated Access Manager configuration and inherited by every organization defined in the instance of Federated Access Manager.

Note – For more information on service files, see *Sun Java System Access Manager 7.1 Administration Guide*.

Most attributes in the SAML module can be configured either through the Federated Access Manager Console or the XML service file. **Broken Link (Target ID: ADMEV)** lists the attributes that can *only* be configured by modifying the `amSAML.xml` file. **Broken Link (Target ID: ADMEW)** lists the attributes that can be configured using the console or the XML service file.

`amSAML.xml` Attributes

The following attributes can *only* be configured through the `amSAML.xml` file using the `amadmin` command-line interface.

- `iplanet-am-saml-cleanup-interval` is used to specify how often the internal thread is run to clean up expired assertions from the internal data store. The default is 180 seconds.

- `iplanet-am-saml-assertion-max-number` is used to specify the maximum number of assertions that the server can hold at one time. No new assertion is created if the maximum number is reached. The default value is 0, which means no limit.

▼ To Modify Attributes in the `amSAML.xml` File

1 Duplicate the `amSAML.xml` service file and make any changes to the attributes.

2 Remove the old `amSAML.xml` service file.

3 Use `amadmin` to reload the newly modified `amSAML.xml` file.

For more information on `amadmin`, see the *Sun Java System Access Manager 7.1 Administration Guide*.

Console Attributes

The following SAML attributes can be configured by using the Access Manager Console or by modifying `amSAML.xml` as described in **Broken Link (Target ID: ADMEV)**. When viewed using the Console, the SAML attributes are separated into the following groups:

- **Broken Link (Target ID: ADMEX)**
- **Broken Link (Target ID: ADMFC)**
- **Broken Link (Target ID: ADMFF)**
- **Broken Link (Target ID: ADMFI)**

Properties Group

The attributes in the Properties group are as follows:

- **Broken Link (Target ID: ADMEY)**
- **Broken Link (Target ID: ADMEZ)**
- **Broken Link (Target ID: ADMFA)**
- **Broken Link (Target ID: ADMFB)**

Target Specifier

This attribute assigns a name to the destination site URL value that is used in the redirects discussed in **Broken Link (Target ID: ADMEO)**. The default is `TARGET`. Only sites configured in the Trusted Partners attribute can be specified as a `TARGET`. For information, see **Broken Link (Target ID: ADMFA)**.

Site Identifiers

This attribute defines any site that is hosted by the server on which Federated Access Manager is installed. A default value is defined for the host during installation (with values retrieved from `AMConfig.properties`), and a Site ID is automatically generated. Multiple entries are possible (for example, load balancing or multiple instances of Federated Access Manager sharing the same Directory Server) although the default site identifier should always remain an entry.

Note – If configuring SAML for SSL (in both the source and destination site), ensure that the protocol defined in the Instance ID attribute is `HTTPS//`.

▼ To Configure a Site Identifier

You may also edit or duplicate entries already listed.

1. **In the Federated Access Manager Console, click the Federation tab.**
2. **Under Federation, click the SAML tab.**
3. **Select New under the Site Identifiers attribute.**
4. **Enter values for the following attributes:**

   Instance ID
   : The value of this property is *protocol*`://`*host*`:`*port*.

   Site ID
   : This identifier is generated for each site, although the value will be the same for multiple servers behind a load balancer. To obtain this identifier manually, type the following at the command line:

   ```
   % #java -classpath AM-classpath \ com.sun.identity.saml.common.SAMLSiteID
   \protocol://host:port
   ```

   For more information, see **Broken Link (Target ID: ADMFR)**.

   Issuer Name
   : The value of this property is *host*`:`*port*.

5. **Click OK.**

Trusted Partners

This attribute defines any trusted partner (remote to the server on which Federated Access Manager is installed) that will be communicating with Federated Access Manager.

Note – The trusted partner site must have a prearranged trust relationship with one or more of the sites configured in **Broken Link (Target ID: ADMEZ)**.

Before configuring a trusted partner, you must 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 receives a single sign-on assertion). Following is the procedure for configuring a trusted partner.

▼ To Configure a Trusted Partner

The Trusted Partners attribute can contain one or more entries. Each entry is configured based on the site's defined role. 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 which profile it uses to receive assertions.

1. **In the Federated Access Manager Console, click the Federation tab.**

2. **Under Federation, click the SAML tab.**

3. **Select New under the Trusted Partners attribute.**

4. **Select the role (Destination or Source) of the partner site that you are configuring by checking the appropriate profiles used to communicate with it and click Next.**

   Select Web Browser Artifact Profile or Web Browser Post Profile for either Destination, Source, or both, or SOAP Query for Source. The choices made dictate which of the attributes in the following steps need to be configured.

5. **Type values for the Common Settings subattributes based on the selected roles.**

   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 **Broken Link (Target ID: ADMEZ)**.

   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 **Broken Link (Target ID: ADMFA)**.

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. For more information, see **Broken Link (Target ID: GANCY)**.

Version

The SAML version used (`1.0` or `1.1`) to send SAML requests. If this parameter is not defined, 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. The default is `com.sun.identity.saml.plugins.DefaultAccountMapper`. An account mapper needs to be implemented from one of the included interfaces:

- `com.sun.identity.saml.plugins.AccountMapper`
- `com.sun.identity.saml.plugins.PartnerAccountMapper`

If no class is defined, no attributes will be included in the assertion. For more information, see **Broken Link (Target ID: GANBL)**.

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 `Certificate` name 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*.

6 **Type values for the Destination subattributes.**

Artifact: SAML URL
The URL that points to the servlet that implements the Web Browser Artifact Profile. See **Broken Link (Target ID: ADMEP)**.

Post: Post URL
The URL that points to the servlet that implements the Web Browser POST Profile. See **Broken Link (Target ID: ADMEQ)**.

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. For more information, see **Broken Link (Target ID: ADMFS)**.

SOAP Query: Action Mapper
The class that is used to get single sign-on information and map partner actions to Federated Access Manager authorization decisions. You need to implement an action mapper from the included interface. If no class is specified, the `DefaultActionMapper` will be used. For more information, see **Broken Link (Target ID: ADMFS)**.

7 Type values for the Source subattributes.

Artifact: SOAP URL
The URL to the SAML SOAP Receiver. See **Broken Link (Target ID: ADMER)**.

Authentication Type
Authentication types that can be used with SAML:

- `NOAUTH`

  Specify `NOAUTH` if the URL to the SAML SOAP receiver is accessed using HTTP, and the SAML SOAP receiver is not protected by HTTP basic authentication.
- `BASICAUTH`

  Specify `BASICAUTH` if the URL to the SAML SOAP receiver is accessed using HTTP, and the SAML SOAP receiver is protected by HTTP basic authentication.
- `SSL`

  Specify `SSL` if the URL to the SAML SOAP receiver is accessed using HTTPS, and the SAML SOAP receiver is not protected by HTTP basic authentication.
- `SSLWITHBASICAUTH`

  Specify `SSLWITHBASICAUTH` if the URL to the SAML SOAP receiver is accessed using HTTPS, and the SAML SOAP receiver is protected by HTTP basic authentication.

Note – If you are protecting the SAML SOAP receiver URL with HTTP basic authentication, you do so in the web container configuration and not in the Federated Access Manager configuration. You do, however, supply the HTTP basic authentication user ID and password in the Federated Access Manager configuration.

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`. For more information, see **Broken Link (Target ID: ADMFA)**.

User
: When Basic Authentication 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 Basic Authentication 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.

8 **Click Finish to complete the configuration.**

Target URLs

If the `TARGET` URL received through either profile is listed as a value of this attribute, the assertions received will be sent to the `TARGET` URL using an HTTP FORM POST.

Caution – Do not use test URLs or any other additional URLs in a POST.

To configure this attribute, type values for the following subattributes:

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
: The port number, such as `58080`.

Path
: The URI, such as `/amserver/console`.

Assertion

The attributes in the Assertion group are as follows:

- **Broken Link (Target ID: ADMFD)**
- **Broken Link (Target ID: ADMFE)**

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

The attributes in the Artifact group are as follows:

- **Broken Link (Target ID: ADMFG)**
- **Broken Link (Target ID: ADMFH)**

For more information about artifacts, see **Broken Link (Target ID: ADMEP)**.

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`.

Signing

The attributes in the Signing group are as follows:

- **Broken Link (Target ID: ADMFJ)**
- **Broken Link (Target ID: ADMFK)**
- **Broken Link (Target ID: ADMFL)**

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 are digitally signed whether or not this feature is enabled.

SAML Operations

This section contains procedures illustrating how to use the Federated Access Manager SAML Service. They are:

- **Broken Link (Target ID: GDWOY)**

Setting Up SAML Single Sign-on

The following procedures explain how to configure and access instances of Federated Access Manager for single sign-on using SAML 1.x assertions. Machine A (`exampleA.com`) is the source site which authenticates the user and creates the SAML authentication assertion. Machine B (`exampleB.com`) is the destination site which consumes the assertion and generates a `SSOToken` for the user.

Note – If both machines are in the same domain, the cookie names must be different. You can change the cookie name by modifying the `com.iplanet.am.cookie.name` property in `AMConfig.properties`, located in `/etc/opt/SUNWam/config/`.

This section contains the following procedures:

- **Broken Link (Target ID: GDWOJ)**
- **Broken Link (Target ID: GDWPI)**

▼ To Set Up SAML Single Sign-on

This procedure assumes the following values:

Deployment URI	`amserver`
Port	58080
Protocol	`http`

1 Write down or copy the value of the Site ID attribute from the destination site (machine B).

a. Login to the Federated Access Manager console running at `exampleB.com` as the default administrator, `amadmin`.

b. Click the Federation tab.

c. Click the SAML tab.

d. Click the sole entry listed under Site Identifiers.

This takes you to the *Edit site identifier* page.

e. Write down or copy the value of the Site ID attribute.

f. Click Cancel.

g. Log out of this instance of Federated Access Manager.

2 Configure the source site (machine A) to trust the destination site (machine B) AND write down or copy the value of the Site ID attribute from the source site.

a. Login to the Federated Access Manager console running at `exampleA.com` as the default administrator, `amadmin`.

b. **Click the Federation tab.**

c. **Click the SAML tab.**

d. **Click New under Trusted Partners.**

   This takes you to the *Select trusted partner type and profile* page.

e. **Check Artifact and Post under Destination and click Next.**

   This takes you to the *Add New Trusted Partner* page.

f. **Set the values of the following attributes to configure machine B as a trusted partner of machine A:**

Source ID	Type the Site ID copied from the destination site, machine B, in the previous step.
Target	The value of this attribute contains the host's domain or domain with port. Do not include the accompanying protocol. For example, `exampleB.com` and `exampleB.com:58080` are valid but, `http://exampleB.com:58080`.
SAML URL	`http://exampleB.com:58080/amserver/SAMLAwareServlet`
HOST LIST	`exampleB.com`
POST URL	`http://exampleB.com:58080/amserver/SAMLPOSTProfileServlet`

g. **Click Finish.**

h. **Click Save.**

i. **Click the sole entry listed under Site Identifiers.**

   This takes you to the *Edit site identifier* page.

j. **Write down or copy the value of the Site ID attribute.**

k. **Click Cancel to go to previous page.**

l. **Log out of Federated Access Manager.**

3 **Configure the destination site (machine B) to trust the source site (machine A).**

a. **Login to the Federated Access Manager console running at** `exampleB.com` **as the default administrator,** `amadmin`**.**

b. **Click the top-level realm under Access Control.**

c. **Click the Authentication tab.**

d. **Click New under Module Instances.**

e. **Type a value in the Name field.**

f. **Select the SAML radio button and click OK.**

g. **Click Save.**

h. **Click Access Control in the upper left corner.**

i. **Click the Federation tab.**

j. **Click the SAML tab.**

k. **Click New under Trusted Partners.**

   This takes you to the *Select trusted partner type and profile* page.

l. **Check Artifact and Post under Source and click Next.**

   This takes you to the *Add New Trusted Partner* page.

m. **Set the values of the following attributes to configure machine A as a trusted partner of machine B:**

Source ID	Type the Site ID you copied from the source site, machine A, in the previous step.
SOAP URL	`http://exampleA.com:58080/amserver/SAMLSOAPReceiver`
Issuer	`exampleA.com:58080`

Note – If machine B uses `https`, check SSL under Authentication Type. Be sure to modify the protocol in the other attributes as necessary.

---

n. **Click Finish.**

o. **Click Save.**

p. **Log out of Federated Access Manager.**

▼ To Verify the SAML Single Sign-on Configurations

1 **Login to the Federated Access Manager console running at** `exampleA.com` **as the default administrator,** `amadmin`**.**

2 **To initialize single sign-on from machine A, do one of the following:**

- **Access the following URL to use the SAML Artifact profile:**

  `http://exampleA.com:58080/amserver/SAMLAwareServlet?`
  `TARGET=`*exampleB.com_Target_URL*

- **Access the following URL to use the SAML POST profile:**

  `http://exampleA.com:58080/amserver/SAMPOSTProfileServlet?`
  `TARGET=`*exampleB.com_Target_URL*

  Note – XML signing must be enabled before running the SAML POST profile. See **Broken Link (Target ID: GDSWN)** for details.

  ---

exampleB.com_Target_URL is any URL on the `exampleB.com` site to which the user will be redirected after a successful single sign-on. For testing purpose, this could be the login page as in `TARGET=http://exampleB.com:58080/amserver/UI/Login`. If the administrator successfully accesses the Federated Access Manager console on the destination site without manual authentication, we know that an `SSOtoken` has been created for the principal on the destination site and single sign-on has been properly established.

CHAPTER 10

SAMLv2 Administration

The SAML v2 Plug-in for Federation Services is an auxiliary program that works with either Sun Federated Access Manager. The plug-in incorporates a subset of features based on the Security Assertion Markup Language (SAML) version 2 specifications and, when installed, allows support for interactions based on those specifications.

Administration

In order to communicate using the SAML v2 profiles you need, at least, two instances of the installed SAML v2 Plug-in for Federation Services. One instance will act for the identity provider and the other will act for the service provider. To prepare your instances of the SAML v2 Plug-in for Federation Services for interactions, you need to exchange configuration information or *metadata* with all participating identity and service providers, import each provider's metadata using an XML-based *metadata configuration file*, and assemble the providers into a *circle of trust*. The SAML v2 Plug-in for Federation Services accomplishes all this administration and configuration using the command-line interface, `saml2meta`. Utility APIs can then be used to communicate with the data store, reading, writing, and managing the relevant properties and property values.

Note – Membership in a circle of trust is transient and might change over the life cycle of the circle as relationships among the partners themselves change.

Provider Metadata and Circles of Trust

In order to communicate using the SAML v2 profiles you need, at the least, two instances of the installed SAML v2 Plug-in for Federation Services configured in one circle of trust. Circles of trust configured for real time interactions must have, at the least, one instance acting as the circle's identity provider and one instance acting as a service provider.

To prepare your instances of the SAML v2 Plug-in for Federation Services for interactions, you need to exchange and import the metadata for all participating identity and service providers, and assemble the providers into a circle of trust. The following steps are an overview of the process. Many of these steps are accomplished using the `saml2meta` interface as described in **Broken Link (Target ID: GBSBB)**.

Note – Before beginning this configuration process, be sure that you have completed the instructions in **Broken Link (Target ID: GBPTK)**.

---

1. Decide whether the instance of the SAML v2 Plug-in for Federation Services you are configuring will act as either an identity provider, a service provider, or both.
2. Create standard and extended metadata configuration files containing the appropriate metadata for your organization as described in **Broken Link (Target ID: GBSUR)**.

   Tip – Alternately, you can use the default identity provider metadata configuration files (`idpMeta.xml` and `idpExtended.xml`) or the default service provider metadata configuration files (`spMeta.xml` and `spExtended.xml`) created during installation. See **Broken Link (Target ID: GBSTH)** and **Broken Link (Target ID: GBSTW)** for more information.

   ---

3. Create a circle of trust as described in **Broken Link (Target ID: GDQLB)**.

   Information about circles of trust can be found in **Broken Link (Target ID: GDQMT)**.
4. Import your organization's provider metadata into the circle of trust as described in **Broken Link (Target ID: GBTDG)**.
5. Determine which organizations will be added to the circle of trust as identity providers and service providers and create a standard and an extended metadata configuration file for each. See **Broken Link (Target ID: GBSUR)**.

   Note – The values in these files will come from the providers themselves.

   ---

6. Import the trusted provider metadata into the circle of trust as described in **Broken Link (Target ID: GBTDG)**.
7. Restart the web container.

Metadata

SAML profiles require that pre-interaction agreements regarding user identifiers, provider (entity) identifiers, binding support, SOAP endpoints, public key information and other similar types of data be made between providers in a circle of trust. This configuration information, or *metadata*, is defined in an XML file and shared amongst all providers who will participate in the interactions. Application programming interfaces (API) are then used to communicate with the data store; reading, writing, and managing the relevant properties and property values. There are two classifications of metadata:

- *Standard metadata* is defined in the Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0 specification. The specification includes information such as the single sign-on service URL and the assertion consumer service URL and is written so as to be extensible. For more information, see **Broken Link (Target ID: GBSTH)**.
- *Extended metadata* are properties used by the SAML v2 Plug-in for Federation Services proprietary features and include information such as the account mapper implementation class, and the local authentication URL. For more information, see **Broken Link (Target ID: GBSTW)**.

Instructions on how to use to the `saml2meta` command-line interface to manage metadata is in **Broken Link (Target ID: GBTDG)**. Instructions on how to generate a dual provider metadata configuration file is in **Broken Link (Target ID: GBVKB)**.

Note – Metadata is sometimes referred to as *entity descriptor* or *entity configuration* where entity generically refers to the `entityID` with which each provider is uniquely identified. For more information on the `entityID`, see **Broken Link (Target ID: GBSTW)**.

Standard Metadata Properties

Standard metadata properties are defined in the Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0 specification and include information such as the single sign-on service URL and the assertion consumer service URL. During installation, two standard metadata configuration files are created for use as input to the `saml2meta` utility. They are located in `/`*FederatedAccessManager-base*`/`*product-directory*`/saml2/meta`.

- `idpMeta.xml` is the default standard metadata configuration file if your instance of the SAML v2 Plug-in for Federation Services will act as an identity provider.
- `spMeta.xml` is the default standard metadata configuration file if your instance of the SAML v2 Plug-in for Federation Services will act as an service provider.

The following sections define both the identity provider and service provider standard metadata properties that have been implemented in the SAML v2 Plug-in for Federation Services.

- **Broken Link (Target ID: GBTLK)**

- **Broken Link (Target ID: GBTKU)**

Note – A complete listing of all the standard metadata properties can be found in the Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0.

Identity Provider Standard Metadata Properties

The identity provider standard metadata properties implemented in the SAML v2 Plug-in for Federation Services are defined in the following table.

`WantAuthnRequestsSigned`	Takes a value of `true` or `false`. If `true`, all authentication requests received by this identity provider must be signed.
`ArtifactResolutionService`	Defines the endpoint(s) that support the Artifact Resolution profile.
`SingleLogoutService`	Defines the endpoint(s) that support the Single Logout profiles.
`ManageNameIDService`	Defines the endpoint(s) that support the Name Identifier Management profiles.
`NameIDFormat`	Defines the name identifier formats supported by the identity provider. Name identifiers are a way for providers to communicate with each other regarding a user. Single sign-on interactions support two types of identifiers: ■ A *persistent identifier* is saved to a particular user's data store entry as the value of two attributes. ■ A *transient identifier* is temporary and no data will be written to the user's persistent data store. More information about name identifiers is in **Broken Link (Target ID: GBUET)**.
`SingleSignOnService`	Defines the endpoint(s) that support the profiles of the Authentication Request protocol. All identity providers must support at least one such endpoint.

Service Provider Standard Metadata Properties

The service provider standard metadata properties implemented in the SAML v2 Plug-in for Federation Services are defined in the following table.

`AuthnRequestsSigned`	Takes a value of `true` or `false`. If `true`, the service provider will sign all outgoing authentication requests.

`WantAssertionsSigned`	Takes a value of `true` or `false`. If `true`, all assertions received by this service provider must be signed.
`SingleLogoutService`	Defines the endpoint(s) that support the Single Logout profiles.
`ManageNameIDService`	Defines the endpoint(s) that support the Name Identifier Management profiles.
`NameIDFormat`	Defines the name identifier formats supported by the service provider. Name identifiers are a way for providers to communicate with each other regarding a user. Single sign-on interactions support two types of identifiers: ■ A *persistent identifier* is saved to a particular user's data store entry as the value of two attributes. ■ A *transient identifier* is temporary and no data will be written to the user's persistent data store. More information about name identifiers is in **Broken Link (Target ID: GBUET)**.
`AssertionConsumerService`	Defines the endpoint(s) that support the profiles of the Authentication Request protocol. All service providers support at least one such endpoint.

Extended Metadata Properties

Extended metadata properties are properties used by our proprietary features and include information such as the account mapper implementation class and the local authentication URL. The properties are specific to whether the provider is an identity provider or a service provider. During installation, two extended metadata configuration files are created for use as input to the `saml2meta` command. They are located in `/`*FederatedAccessManager-base*`/`*product-directory*`/saml2/meta` or `/`*FederationManager-base*`/SUNWam/saml2/meta`.

- `idpExtended.xml` is the default extended metadata configuration file if your instance of the SAML v2 Plug-in for Federation Services will act as an identity provider.
- `spExtended.xml` is the default extended metadata configuration file if your instance of the SAML v2 Plug-in for Federation Services will act as an service provider.

The following sections define properties in the identity provider and service provider extended metadata.

- **Broken Link (Target ID: GBSUO)**
- **Broken Link (Target ID: GBSUE)**

Identity Provider Extended Metadata Properties

The identity provider extended metadata properties are defined in the following table.

`hosted`	Specifies whether the entity is hosted on, or remote to, the server to which this metadata is being applied. A value of `0` or `flase` specifies that the entity is hosted. A value of `1` or `true` specifies that the entity is hosted.
`entityID`	Specifies the `EntityID` of the provider you are configuring. The value of `EntityID` for your local provider is the unique uniform resource identifier (URI) you decide to use to identity yourself to other providers. You will get a remote provider's `EntityID` from the metadata they give to you. **Note –** This `EntityID` is different from the entities configured using the console in Federated Access Manager. It is specific to SAML v2 interactions.
`metaAlias`	Specifies a `metaAlias` for the provider being configured. The `metaAlias` is used to locate the provider's entity identifier and the organization in which it is located. The value is a string equal to the realm or organization name (dependent on whether the SAML v2 Plug-in for Federation Services is installed in Federated Access Manager) coupled with a forward slash and the provider name. For example, `/suncorp/travelprovider`. **Caution –** The names used in the `metaAlias` must not contain a `/`.
`signingCertAlias`	Specifies the provider certificate alias used to find the correct signing certificate in the keystore.
`encryptionCertAlias`	Specifies the provider certificate alias used to find the correct encryption certificate in the keystore.
`basicAuthOn`	Basic authentication can be turned on to protect SOAP endpoints. This property takes a value of `true` or `false`. Any provider accessing these endpoints must have the user and password defined in the following two properties: `basicAuthUser` and `basicAuthPassword`.
`basicAuthUser`	The user associated with the basic authentication.
`basicAuthPassword`	The password associated with the basic authentication.
`autofedEnabled`	Enables auto-federation which automatically federates a user's disparate provider accounts based on a common attribute. This property takes a value of `true` or `false`.
`autofedAttribute`	Specifies the attribute used to match a user's disparate provider accounts when auto-federation is enabled.
`assertionEffectiveTime`	Specifies (in seconds) the amount of time that an assertion is valid counting from the assertion's issue time. The default value is `600` seconds.

`idpAuthncontextMapper`	Specifies the name of the implementation class for the `IDPAuthnContextMapper` interface. This class sets the authentication context in the assertion. The default value is `com.sun.identity.saml2.plugins.DefaultIDPAuthnContextMapper`, the default implementation.
`idpAuthncontextClassrefMapping`	Sets the mappings between the requested authentication context class and the actual authentication mechanism. The value of this attribute is in the format of: *authnContextClassRef* **|** *authnType***=***authnValue* **|** *authnType***=***authnValue* **|** ... where *authnContextClassRef* is the authentication context class reference, *authnType* is the module, level, or service, and *authnValue* is the module name, authentication level, or service name.
`idpAccountMapper`	Specifies the implementation of the `AccountMapper` interface used to map a remote user account to a local user account for purposes of single sign-on. The default value is `com.sun.identity.saml2.plugins.DefaultIDPAccountMapper`, the default implementation.
`idpAttributeMapper`	Specifies the implementation of the `AttributeMapper` interface used to map a remote user account attribute to a local user account attribute for purposes of single sign-on. The default value is `com.sun.identity.saml2.plugins.DefaultIDPAttributeMapper`, the default implementation
`attributeMap`	Specifies the mapping of attributes between providers. The value of this attribute is in the format: *SAML v2-attribute***=***user-attribute* where *SAML v2-attribute* is the attribute name that goes over the wire and *user-attribute* is the attribute name it will map to once it arrives. **Note** – If auto-federation is enabled, the value of *SAML v2-attribute* is equal to the value of `autofedAttribute`.
`wantNameIDEncrypted`	Takes a value of `true` or `false`. If `true`, the service provider must encrypt all `NameID` elements.
`wantArtifactResolveSigned`	Takes a value of `true` or `false`. If `true`, the service provider must sign the `ArtifactResolve` element.
`wantLogoutRequestSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `LogoutRequest` element.

`wantLogoutResponseSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `LogoutResponse` element.
`wantMNIRequestSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `ManageNameIDRequest` element.
`wantMNIResponseSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `ManageNameIDResponse` element.
`cotlist`	Specifies the name of the circle(s) of trust to which this provider belongs. As one provider may be in a number of circles, this attribute might have multiple values.

Service Provider Extended Metadata Properties

The service provider extended metadata properties are defined in the following table.

`hosted`	Specifies whether the entity is hosted on, or remote to, the server to which this metadata is being applied. A value of `0` or `flase` specifies that the entity is hosted. A value of `1` or `true` specifies that the entity is hosted.
`entityID`	Specifies the `EntityID` of the provider you are configuring. The value of `EntityID` for your local provider is the unique uniform resource identifier (URI) you decide to use to identity yourself to other providers. You will get a remote provider's `EntityID` from the metadata they give to you. **Note –** This `EntityID` is different from the entities configured using the console in Federated Access Manager. It is specific to SAML v2 interactions.
`metaAlias`	Specifies a `metaAlias` for the provider being configured. The `metaAlias` is used to locate the provider's entity identifier and the organization in which it is located. The value is a string equal to the realm or organization name (dependent on whether the SAML v2 Plug-in for Federation Services is installed in Federated Access Manager) coupled with a forward slash and the provider name. For example, `/suncorp/travelprovider`. **Caution –** The names used in the `metaAlias` must not contain a `/`.
`signingCertAlias`	Specifies the provider certificate alias used to find the correct signing certificate in the keystore.
`encryptionCertAlias`	Specifies the provider certificate alias used to find the correct encryption certificate in the keystore.

`basicAuthOn`	Basic authentication can be turned on to protect SOAP endpoints. This property takes a value of `true` or `false`. Any provider accessing these endpoints must have the user and password defined in the following two properties: `basicAuthUser` and `basicAuthPassword`.
`basicAuthUser`	The user associated with the basic authentication.
`basicAuthPassword`	The password associated with the basic authentication.
`autofedEnabled`	Auto-federation automatically federates a user's disparate provider accounts based on a common attribute. This property takes a value of `true` or `false`.
`autofedAttribute`	Specifies the attribute used to match a user's disparate provider accounts when auto-federation is enabled.
`spAccountMapper`	Specifies the implementation of the `AccountMapper` interface used to map a remote user account to a local user account for purposes of single sign-on. The default value is `com.sun.identity.saml2.plugins.DefaultSPAccountMapper`, the default implementation.
`spAttributeMapper`	Specifies the implementation of the `AttributeMapper` interface used to map a remote user account attribute to a local user account attribute for purposes of single sign-on. The default value is `com.sun.identity.saml2.plugins.DefaultSPAttributeMapper`, the default implementation
`spAuthncontextMapper`	Specifies the implementation of the `SPAuthnContextMapper` interface used to create the requested authentication context. The default implementation is `com.sun.identity.saml2.plugins.DefaultSPAttributeMapper`.
`spAuthncontextClassrefMapping`	Sets the provider's desired authentication context class and authentication level. Multiple values can be specified. The value of this property is in the format: *authnContextClassRef* **|** *authlevel* **|** *default* For example: **`urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport | 1`** or **`urn:oasis:names:tc:SAML:2.0:ac:classes:Password | 0 | default`**

`spAuthncontextComparisonType`	Specifies what the resulting authentication context must be when compared to the value of this property. Accepted values include: ■ *exact* where the authentication context statement in the assertion must be the exact match of, at least, one of the authentication contexts specified. ■ *minimum* where the authentication context statement in the assertion must be, at least, as strong (as deemed by the identity provider) one of the authentication contexts specified. ■ *maximum* where the authentication context statement in the assertion must be no stronger than any of the authentication contexts specified. ■ *better* where the authentication context statement in the assertion must be stronger than any of the authentication contexts specified. The default value is *exact*.
`attributeMap`	Specifies the mapping of attributes between providers. The value of this attribute is in the format: *SAML v2-attribute*=*user-attribute* where *SAML v2-attribute* is the attribute name that goes over the wire and *user-attribute* is the attribute name it will map to once it arrives. **Note –** If auto-federation is enabled, the value of *SAML v2-attribute* is equal to the value of `autofedAttribute`.
`saml2AuthModuleName`	Specifies the name of the instance of the SAML v2 authentication module. The default value is `SAML2`.
`localAuthURL`	Specifies the URL of the local login page. For more information, see **Broken Link (Target ID: GBTKM)**.
`intermediateUrl`	Specifies a URL to which a user can be directed after authentication and before the original request's URL. An example might be a successful account creation page after the auto-creation of a user account.

`defaultRelayState`	After a successful SAML v2 operation (single sign-on, single logout, or federation termination), a page is displayed. This page, generally the originally requested resource, is specified in the initiating request using the `RelayState` element. If a `RelayState` is not specified, the value of this `defaultRelayState` property is displayed. For more information, see **Broken Link (Target ID: GBTKL)**.
	
	Caution – When `RelayState` or `defaultRelayState` contains special characters (such as &), it must be URL-encoded. For example, if the value of `RelayState` is `http://www.sun.com/apps/myapp.jsp?param1=abc&param2=xyz`, it must be URL-encoded as:
	
	`http%3A%2F%2Fwww.sun.com%2Fapps%2Fmyapp.jsp%3Fparam1%3Dabc%26param2%3Dxyz`
	
	and then appended to the URL. For example, the service provider initiated single sign-on URL would be:
	
	`http://`*host*`:`*port*`/`*deploy-uri*`/saml2/jsp/spSSOInit.jsp?metaAlias=/sp&idpEntityID=http://www.idp.com&RelayState=http%3A%2F%2Fwww.sun.com%2Fapps%2Fmyapp.jsp%3Fparam1%3Dabc%26param2%3Dxyz`

`AssertionTimeSkew`	Assertions are valid for a period of time and not before or after. This property specifies a grace period (in seconds) for the `notBefore` value. The default value is `300`. It has no relevance to the `notAfter` value.
`wantAttributeEncrypted`	Takes a value of `true` or `false`. If `true`, the identity provider must encrypt all `AttributeStatement` elements.
`wantAssertionEncrypted`	Takes a value of `true` or `false`. If `true`, the identity provider must encrypt all `Assertion` elements.
`wantNameIDEncrypted`	Takes a value of `true` or `false`. If `true`, the identity provider must encrypt all `NameID` elements.
`wantArtifactResponseSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `ArtifactResponse` element.
`wantLogoutRequestSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `LogoutRequest` element.
`wantLogoutResponseSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `LogoutResponse` element.
`wantMNIRequestSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `ManageNameIDResponse` element.
`wantMNIResponseSigned`	Takes a value of `true` or `false`. If `true`, the identity provider must sign the `ManageNameIDResponse` element.

`cotlist`	Specifies the name of the circle of trust to which this provider belongs.
`transientUser`	Specifies the identifier of the user to which all identity provider users will be mapped on the service provider side in cases of single sign-on using the transient name identifier.

Dual Purpose Provider Metadata Files

According to the SAML v2 specifications, one metadata file can contain configuration data for one identity provider and one service provider. Thus, it is possible to create one standard metadata configuration file and one extended configuration file which, when imported, will configure one member of a circle of trust to act as both an identity provider and a service provider. Sample files and instructions on how to generate them are found in the following sections.

- **Broken Link (Target ID: GBWDW)**
- **Broken Link (Target ID: GBWDT)**
- **Broken Link (Target ID: GBVJW)**

Dual Purpose Standard Metadata Configuration File

The dual purpose standard metadata file would contain one `<EntityDescriptor>` element containing both `<IDPSSODescriptor>` and `<SPSSODescriptor>` elements. The following sample is a standard metadata configuration file in which the data configures `zosma21.central.sun.com` as both a service provider and an identity provider.

```
<EntityDescriptor
    xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
    entityID="zosma21.central.sun.com/">
    <IDPSSODescriptor
        WantAuthnRequestsSigned="false"
        protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <ArtifactResolutionService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
            Location="http://zosma21.central.sun.com:80/amserver/ArtifactResolver/
             metaAlias/idp"
            index="0"
            isDefault="1"/>
        <SingleLogoutService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
            Location="http://zosma21.central.sun.com:80/amserver/IDPSloRedirect/
             metaAlias/idp"
            ResponseLocation="http://zosma21.central.sun.com:80/amserver/
             IDPSloRedirect/metaAlias/idp"/>
        <SingleLogoutService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
```

```
                Location="http://zosma21.central.sun.com:80/amserver/
                 IDPSloSoap/metaAlias/idp"/>
            <ManageNameIDService
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
                Location="http://zosma21.central.sun.com:80/amserver/IDPMniRedirect/
                 metaAlias/idp"
                ResponseLocation="http://zosma21.central.sun.com:80/amserver/
                 IDPMniRedirect/metaAlias/idp"/>
            <ManageNameIDService
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
                Location="http://zosma21.central.sun.com:80/amserver/IDPMniSoap/
                 metaAlias/idp"/>
            <NameIDFormat>
                urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
            </NameIDFormat>
            <NameIDFormat>
                urn:oasis:names:tc:SAML:2.0:nameid-format:transient
            </NameIDFormat>
            <SingleSignOnService
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
                Location="http://zosma21.central.sun.com:80/amserver/SSORedirect/
                 metaAlias/idp"/>
            <SingleSignOnService
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
                Location="http://zosma21.central.sun.com:80/amserver/SSOSoap/
                 metaAlias/idp"/>
        </IDPSSODescriptor>
        <SPSSODescriptor
            AuthnRequestsSigned="false"
            WantAssertionsSigned="false"
            protocolSupportEnumeration=
                "urn:oasis:names:tc:SAML:2.0:protocol">
            <SingleLogoutService
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
                Location="http://zosma21.central.sun.com:80/amserver/SPSloRedirect/
                 metaAlias/sp"
                ResponseLocation="http://zosma21.central.sun.com:80/amserver/
                 SPSloRedirect/metaAlias/sp"/>
            <SingleLogoutService
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
                Location="http://zosma21.central.sun.com:80/amserver/SPSloSoap/
                 metaAlias/sp"/>
            <ManageNameIDService
                Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
                Location="http://zosma21.central.sun.com:80/amserver/SPMniRedirect/
                 metaAlias/sp"
                ResponseLocation="http://zosma21.central.sun.com:80/amserver/
                 SPMniRedirect/metaAlias/sp"/>
        <ManageNameIDService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
            Location="http://zosma21.central.sun.com:80/amserver/SPMniSoap/
             metaAlias/sp"
            ResponseLocation="http://zosma21.central.sun.com:80/amserver/
             SPMniSoap/metaAlias/sp"/>
        <NameIDFormat>
            urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
        </NameIDFormat>
        <NameIDFormat>
            urn:oasis:names:tc:SAML:2.0:nameid-format:transient
        </NameIDFormat>
        <AssertionConsumerService
            isDefault="true"
            index="0"
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
            Location="http://zosma21.central.sun.com:80/amserver/Consumer/
             metaAlias/sp"/>
        <AssertionConsumerService
            index="1"
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
            Location="http://zosma21.central.sun.com:80/amserver/Consumer/
             metaAlias/sp"/>
    </SPSSODescriptor>
</EntityDescriptor>
```

Dual Purpose Extended Metadata Configuration File

The dual purpose extended metadata file would contain one <EntityConfig> element containing both <IDPSSOConfig> and <SPSSOConfig> elements. The following sample is an extended metadata configuration file in which the data configures zosma21.central.sun.com as both a service provider and an identity provider.

```
<EntityConfig xmlns="urn:sun:fm:SAML:2.0:entityconfig"
    xmlns:fm="urn:sun:fm:SAML:2.0:entityconfig"
    hosted="1"
    entityID="zosma21.central.sun.com/">
    <IDPSSOConfig metaAlias="/idp">
       <Attribute name="signingCertAlias">
           <Value></Value>
       </Attribute>
       <Attribute name="encryptionCertAlias">
           <Value></Value>
       </Attribute>
       <Attribute name="basicAuthOn">
           <Value>false</Value>
       </Attribute>
       <Attribute name="basicAuthUser">
```

```
        <Value></Value>
    </Attribute>
    <Attribute name="basicAuthPassword">
        <Value></Value>
    </Attribute>
    <Attribute name="autofedEnabled">
        <Value>false</Value>
    </Attribute>
    <Attribute name="autofedAttribute">
        <Value></Value>
    </Attribute>
    <Attribute name="assertionEffectiveTime">
        <Value>600</Value>
    </Attribute>
    <Attribute name="idpAuthncontextMapper">
        <Value>com.sun.identity.saml2.plugins.DefaultIDPAuthnContextMapper</Value>
    </Attribute>
    <Attribute name="idpAuthncontextClassrefMapping">
        <Value>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport</Value>
    </Attribute>
    <Attribute name="idpAccountMapper">
        <Value>com.sun.identity.saml2.plugins.DefaultIDPAccountMapper</Value>
    </Attribute>
    <Attribute name="idpAttributeMapper">
        <Value>com.sun.identity.saml2.plugins.DefaultIDPAttributeMapper</Value>
    </Attribute>
    <Attribute name="attributeMap">
        <Value></Value>
    </Attribute>
    <Attribute name="wantNameIDEncrypted">
        <Value></Value>
    </Attribute>
    <Attribute name="wantArtifactResolveSigned">
        <Value></Value>
    </Attribute>
    <Attribute name="wantLogoutRequestSigned">
        <Value></Value>
    </Attribute>
    <Attribute name="wantLogoutResponseSigned ">
        <Value></Value>
    </Attribute>
    <Attribute name="wantMNIRequestSigned">
        <Value></Value>
    </Attribute>
    <Attribute name="wantMNIResponseSigned">
        <Value></Value>
    </Attribute>
    <Attribute name="cotlist">
```

```
        </Attribute>
    </IDPSSOConfig>
    <SPSSOConfig metaAlias="/sp">
        <Attribute name="signingCertAlias">
            <Value></Value>
        </Attribute>
        <Attribute name="encryptionCertAlias">
            <Value></Value>
        </Attribute>
        <Attribute name="basicAuthOn">
            <Value>false</Value>
        </Attribute>
        <Attribute name="basicAuthUser">
            <Value></Value>
        </Attribute>
        <Attribute name="basicAuthPassword">
            <Value></Value>
        </Attribute>
        <Attribute name="autofedEnabled">
            <Value>false</Value>
        </Attribute>
        <Attribute name="autofedAttribute">
            <Value></Value>
        </Attribute>
        <Attribute name="transientUser">
            <Value></Value>
        </Attribute>
        <Attribute name="spAccountMapper">
            <Value>com.sun.identity.saml2.plugins.DefaultSPAccountMapper</Value>
        </Attribute>
        <Attribute name="spAttributeMapper">
            <Value>com.sun.identity.saml2.plugins.DefaultSPAttributeMapper</Value>
        </Attribute>
        <Attribute name="spAuthncontextMapper">
            <Value>com.sun.identity.saml2.plugins.DefaultSPAuthnContextMapper</Value>
        </Attribute>
        <Attribute name="spAuthncontextClassrefMapping">
            <Value>PasswordProtectedTransport|0|default</Value>
        </Attribute>
        <Attribute name="spAuthncontextComparisonType">
            <Value>exact</Value>
        </Attribute>
        <Attribute name="attributeMap">
            <Value></Value>
        </Attribute>
        <Attribute name="saml2AuthModuleName">
            <Value></Value>
        </Attribute>
```

```
       <Attribute name="localAuthURL">
           <Value></Value>
       </Attribute>
       <Attribute name="intermediateUrl">
           <Value></Value>
       </Attribute>
       <Attribute name="defaultRelayState">
           <Value></Value>
       </Attribute>
       <Attribute name="assertionTimeSkew">
           <Value>300</Value>
       </Attribute>
       <Attribute name="wantAttributeEncrypted">
           <Value></Value>
       </Attribute>
       <Attribute name="wantAssertionEncrypted">
           <Value></Value>
       </Attribute>
       <Attribute name="wantNameIDEncrypted">
           <Value></Value>
       </Attribute>
       <Attribute name="wantArtifactResponseSigned">
           <Value></Value>
       </Attribute>
       <Attribute name="wantLogoutRequestSigned">
           <Value></Value>
       </Attribute>
       <Attribute name="wantLogoutResponseSigned ">
           <Value></Value>
       </Attribute>
       <Attribute name="wantMNIRequestSigned">
           <Value></Value>
       </Attribute>
       <Attribute name="wantMNIResponseSigned">
           <Value></Value>
       </Attribute>
       <Attribute name="cotlist">
       </Attribute>
    </SPSSOConfig>
</EntityConfig>
```

▼ To Generate Dual Purpose Metadata Configuration Files

This procedure creates one standard metadata file and one extended metadata file that contains configuration information for one provider that, when imported, will define it as capable of both functions. See **Broken Link (Target ID: GBSBB)** for more information on the `saml2meta` command line interface.

1 **Generate the dual purpose standard and extended metadata configuration files.**

`saml2meta [-i` *staging-directory***`] template -u amadmin -w`** *password* **`-e dual -s /sp1 -d /idp1 -m dualMeta.xml -x dualExtended.xml`**

2 **Import the generated standard and extended metadata configuration files.**

`saml2meta [-i` *staging-directory***`] import -u amadmin -w`** *password* **`-m dualMeta.xml -x dualExtended.xml`**

Circles of Trust

Circles of trust need to be created to define trust relationships among identity providers and service providers. A *circle of trust* is a grouping of service providers (with at least one identity provider) that have, in place, pertinent business agreements regarding how they will do business and interact with identities. Any identity provider or service provider within a circle of trust will honor requests and information that come from other providers in the same circle. Requests and information will be rejected if communicating providers are not part of the same circle of trust. In the SAML v2 Plug-in for Federation Services, circles of trust are created using the `saml2meta` command-line interface, allowing you to configure technologically the participants and rules drawn in the business agreements. Instructions on how to use the `saml2meta` command-line interface to manage circles of trust is in **Broken Link (Target ID: GDQLB)**.

`AMConfig.properties`

`AMConfig.properties` is the main configuration file for Federated Access Manager. `AMConfig.properties` is located in the `/etc/opt/`*product-directory*`/config` directory in Federated Access Manager.

Static Properties in `AMConfig.properties`

This list describes the properties added to `AMConfig.properties` when the SAML v2 Plug-in for Federation Services is installed.

Caution – Do not modify the properties configured during installation.

- **`com.sun.identity.saml2.am_or_fm`** takes a value of `FAM` for Federated Access Manager. It specifies the instance type onto which the SAML v2 Plug-in for Federation Services is installed.

- **`com.sun.identity.saml2.xmlenc.EncProviderImpl= com.sun.identity.saml2.xmlenc.FMEncProvider`** specifies the XML encryption provider implementation class.
- **`com.sun.identity.saml2.xmlenc.SigProviderImpl= com.sun.identity.saml2.xmlsig.FMSigProvider`** specifies the XML signature provider implementation class.
- **`com.sun.identity.common.datastore.provider.default= com.sun.identity.saml2.plugins.IdRepoDataStoreProvider`** specifies the data store provider implementation class. The `IdRepoDataStoreProviderdefault` class provides implementation using the identity repository API.
- The **`com.sun.identity.saml2.nameidinfo.attribute`** and **`com.sun.identity.saml2.nameidinfokey`** properties specify the LDAPv3 attribute to which you want federation information written in a principal's account. For more information, see **Broken Link (Target ID: GBTEJ)**.

Additional Properties in `AMConfig.properties`

This list describes the properties and accompanying values that may be added to `AMConfig.properties` after installation.

- **`com.sun.identity.saml2.cacheCleanUpInterval`** takes a value in seconds. It specifies the amount of time between each cache cleanup.
- **`com.sun.identity.saml2.sdk.mapping.`***interface-name*=*new-class-name* is a mapping property for customized implementation classes. For more information, see **Broken Link (Target ID: GBTZX)**.

The SAML v2 IDP Discovery Service

The SAML v2 IDP Discovery Service is an implementation of the Identity Provider Discovery Profile as described in the Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0 specification. In deployments having more than one identity provider, service providers need to determine which identity provider(s) a principal uses with the Web Browser SSO profile. To allow for this, the SAML v2 IDP Discovery Service relies on a cookie written in a domain that is common to all identity providers and service providers in a circle of trust. This predetermined domain is known as the *common domain*, and the cookie containing the list of identity providers to chose from is known as the *common domain cookie*.

When a user requests access from a service provider and an entity identifier for an identity provider is not received in the request, the service provider redirects the request to the common domain's SAML v2 IDP Discovery Service Reader URL to retrieve the identity provider's entity identifier. If more then one identity provider entity identifier is returned, the last entity identifier in the list is the one to which the request is redirected. Once received, the identity provider redirects to the Discovery Service Writer URL to set the common domain cookie using the value defined in the installation configuration properties file. See **Broken Link (Target ID: GBPUT)** for more information.

Note – The Reader and Writer URLs for the SAML v2 IDP Discovery Service are defined when configuring the circle of trust. If the circle already exists and does not contain values for the Reader and Writer URLs, it must be deleted and recreated.

▼ To Set the Reader and Writer URLs

Before You Begin Instructions on how to install the SAML v2 IDP Discovery Service can be found in **Broken Link (Target ID: GBTFH)**. You should also be familiar with **Broken Link (Target ID: GBSBB)** as well as **Broken Link (Target ID: GBTGZ)**.

1. **Delete the current circle of trust configuration using `saml2meta`, if applicable.**

2. **Create a new circle of trust configuration using `saml2meta` and the `cotcreate` subcommand.**

   ```
   saml2meta [-i staging-directory] cotcreate -u admin-user -w password -t COT-name
    -p idp-discovery-URL-path
   ```

   Make sure to specify the full path to where the SAML v2 Plug-in for Federation Services is deployed using the `-p` option.

3. **Add member providers to the new circle of trust using `saml2meta` and the `cotadd` subcommand.**

   ```
   saml2meta [-i staging-directory] cotadd -u admin-user -w password -t COT-name -e entity-ID
   ```

   Tip – `cotadd` can only add a single provider at a time using the `-e` option. To add a group of providers, you can use the `-l` option with `cotcreate` in the previous step.

   ---

4. **Verify that all member providers have been added to the circle using `saml2meta` and the `cotlist` subcommand.**

   ```
   saml2meta [-i staging-directory] cotlist -u admin-user -w password
   ```

Next Steps Service providers will be redirected to the SAML v2 IDP Discovery Service Reader URL during single sign-on.

The `saml2meta` Command-line Reference

The SAML v2 Plug-in for Federation Services contains the `saml2meta` command-line interface to manage metadata and circles of trust. It is located in /*FederatedAccessManager-base*/*product-directory*/`saml2/meta` or /*FederationManager-base*/`SUNWam/saml2/meta`.

The `saml2meta` syntax is:

```
saml2meta [-i staging-directory] import -u user-DN [-w password | -j password-file] [-r realm]
 [-m XML-file-name] [-x XML-file-name] [-t COT_name]

saml2meta [-i staging-directory] export -u user-DN [-w password | -j password-file] [-r realm]
 -e entityID [-n] [ -m  XML-file-name] [-x XML-file-name]

saml2meta [-i staging-directory] template -u user-DN [-w password | -j password-file]
 [-e entityID] [-s metaAlias [-a certAlias] [-f certAlias]] [-d metaAlias
 [-b certAlias] [-g certAlias]] -m XML-file-name -x XML-file-name

saml2meta [-i staging-directory] delete -u user-DN [-w password | -j password-file]
 [-r realm] [-e entityID] [-c]

saml2meta [-i staging-directory] list -u user-DN [-w password | -j password-file]

saml2meta [-i staging-directory] cotcreate -u user-DN [-w password | -j password-file]
 [-t COT-name] [-p prefix-URL] [-l entity-ID, entity-ID, ...]

saml2meta [-i staging-directory] cotdelete -u user-DN [-w password | -j password-file]
 [-t COT-name]

saml2meta [-i staging-directory] cotadd -u user-DN [-w password | -j password-file]
 [-t COT-name] [-e entityID]

saml2meta [-i staging-directory] cotremove -u user-DN [-w password | -j password-file]
 [-t COT-name] [-e entityID]

saml2meta [-i staging-directory] cotmember -u user-DN [-w password | -j password-file]
 -t COT-name

saml2meta [-i staging-directory] cotlist -u user-DN [-w password | -j password_file]

saml2meta -V

saml2meta -?
```

where:

`-i`	Specifies the directory for the web application staging area. For example, `/var/opt/SUNWam/fm/war-staging` **Note –** This option is specific only to instances of the SAML v2 Plug-in for Federation Services installed on Federated Access Manager.
`-u` or `--runasdn`	Specifies the full distinguished name of the user running the command.
`-w` or `--password`	Specifies the password of the user running the command.
`-j` or `--passwordfile`	Specifies the name of the file that contains the password of the user running the command.
`-r` or `--realm`	Specifies the realm or organization under which the metadata is stored. If not defined, the default value is the root realm or organization.
`-m` or `--metadata`	Specifies a file name for the standard configuration. **Note –** In most subcommands, either `-m` or `-x` must be used. Both can also be used.
`-x` or `--extended`	Specifies a file name for the extended configuration. **Note –** In most subcommands, either `-m` or `-x` must be used. Both can also be used.
`-e` or `--entityid`	Specifies an entity identifier, if applicable.
`-s` or `--serviceprovider`	Specifies a `metaAlias` for the hosted service provider being created. The `metaAlias` is used to locate the provider's entity identifier and the organization in which it is located. The value is a string equal to the realm or organization name (dependent on whether the SAML v2 Plug-in for Federation Services is installed in Federated Access Manager) coupled with a forward slash and the provider name. For example, `/suncorp/travelprovider`. **Caution –** The strings used in the `metaAlias` values must not contain a `/`.
`-a` or `--spcertalias`	Specifies a certificate alias for the hosted service provider to be created.
`-f` or `--specertalias`	Specifies an encrypted certificate alias for the hosted service provider to be created.
`-d` or `--identityprovider`	Specifies a `metaAlias` for the hosted identity provider being created. The `metaAlias` is used to locate the provider's entity identifier and the organization in which it is located. The value is a string equal to the realm or organization name (dependent on whether the SAML v2 Plug-in for Federation Services is installed in Federated Access Manager) coupled with a forward slash and the provider name. For example, `/suncorp/hr`. **Caution –** The strings used in the `metaAlias` values must not contain a `/`.
`-b` or `--idpcertalias`	Specifies a certificate alias for the hosted identity provider to be created.

-g or --idpecertalias	Specifies an encrypted certificate alias for the hosted identity provider to be created.
-n or --sign	Signs the exported XML file.
-c or --extendedonly	Deletes extended configurations only.
-t or --cot	Specifies the name of a circle of trust.
-p or --prefix	Specifies the full path to where the SAML v2 IDP Discovery Service is deployed.
-l or --trustedproviders	Specifies a list of trusted providers in a circle of trust. Input is a comma-separated list of entity identifiers.
-V	Displays version information.
-?	Displays help information.

Note – To access usage information on the command-line, change to /*FederatedAccessManager-base*/*product-directory*/saml2/bin or /*FederationManager-base*/SUNWam/saml2/bin and run saml2meta with no input.

For explanations of the saml2meta subcommands, see the:

- saml2meta Subcommands for Managing Metadata in **Broken Link (Target ID: GBTIC)**
- saml2meta Subcommands for Managing Circles of Trust in **Broken Link (Target ID: GBTGZ)**

Managing Metadata using saml2meta

saml2meta is used to manage the SAML v2 metadata. The following table describes the saml2meta subcommands specific to metadata management.

TABLE 10–1 saml2meta Subcommands for Managing Metadata

Subcommand	Description
import	Loads standard and extended metadata in XML format into a local configuration data store. **Note –** Either -m or -x must be used. Both can also be used.

TABLE 10–1 saml2meta Subcommands for Managing Metadata *(Continued)*

Subcommand	Description
export	Exports standard and extended metadata in XML format from a local configuration data store. **Note –** Either -m or -x must be used. Both can also be used.
template	Generates a metadata configuration file for either type of hosted provider (service or identity) with defined values for default metadata properties. The generated file can be modified for use with import.
delete	Removes standard or extended metadata from a local configuration data store.
list	Generates a list of all the entity identifiers on the system.

Following are some examples on how you might use saml2meta. See **Broken Link (Target ID: GBSBB)** for explanations of the options used.

- The following command example will create both a standard and an extended metadata configuration file for service provider sp.sun.com:

  ```
  # saml2meta template -u amadmin -w password -e sp.sun.com -s /sp
   -m spMeta.xml -x spExtended.xml
  ```

  The standard metadata is defined in spMeta.xml and the extended metadata is defined in spExtended.xml.
- This command example will import the created files into the local configuration data store:

  ```
  # saml2meta import -u amadmin -w password -m spMetadata.xml -x spExtended.xml
  ```

  Note – Remember to delete old metadata before you import modified files.

- See **Broken Link (Target ID: GBVKB)**.

Managing Circles of Trust using saml2meta

The saml2meta command line interface creates and manages the circles of trust used by the SAML v2 Plug-in for Federation Services. The following table describes the saml2meta subcommands specific to circle of trust management.

TABLE 10–2 `saml2meta` Subcommands for Managing Circles of Trust

Subcommand	Description
`cotcreate`	Creates a circle of trust.
`cotdelete`	Removes a circle of trust. **Note –** To delete a circle of trust that contains providers, use `cotremove` to remove each provider first, then use `cotdelete` to delete the circle itself.
`cotadd`	Adds a trusted provider to an existing circle of trust. **Note –** `cotadd` can only add a single entity at a time. Add multiple entities when you first create the circle by using `cotcreate` and the `-l` option.
`cotremove`	Removes a trusted provider from an existing circle of trust.
`cotmember`	Lists the member providers in a particular circle of trust.
`cotlist`	Lists all the circles of trust configured on the system.

The following command example will create a circle of trust:

```
saml2meta [-i staging-directory] cotcreate -u admin-user -w password -t COT-name
 -p idp-discovery-URL-path
```

This second command example will add a trusted provider to an existing circle of trust:

```
saml2meta [-i staging-directory] cotadd -u admin-user -w password -t COT-name -e entity-ID
```

This next command example will remove a trusted provider from an existing circle of trust:

```
saml2meta [-i staging-directory] cotremove -u admin-user -w password -t COT-name -e entity-ID
```

This command example will list all the providers belonging to an existing circle of trust:

```
saml2meta [-i staging-directory] cotmember -u admin-user -w password -t COT-name
```

This last command example will list all the available circles of trust under the instance of the SAML v2 Plug-in for Federation Services:

```
saml2meta [-i staging-directory] cotlist -u admin-user -w password
```

Configuring Specialized SAMLv2 Interactions

The SAML v2 Plug-in for Federation Services can be configured to support many specialized interactions. The use cases described in this chapter give you a general idea of how the SAML v2 Plug-in for Federation Services can be used.

This chapter covers the following topics:

Single Sign-on

Name identifiers are used by the identity provider and the service provider to communicate with each other regarding a user. Single sign-on interactions can support both persistent and transient identifiers. A *persistent identifier* is saved to a particular user entry as the value of two attributes. A *transient identifier* is temporary and no data will be written to the user's data store entry.

In some deployments, there might be no user account on the service provider side of an interaction. In this case, single sign-on with the transient name identifier is used. All users passed from the identity provider to the service provider will be mapped to this one user account. All attributes defined in the `AttributeStatement` will be set as properties of the specific user's single sign-on token. The following procedures describe some interactions using the transient name identifier.

- **Broken Link (Target ID: GBTDJ)**
- **Broken Link (Target ID: GBSZL)**

▼ To Use the Transient Name Identifier

1. **Append the `NameIDFormat=transient` query parameter to the URL that initiates a single sign-on JavaServer Page™ (JSP™).**

   `spSSOInit.jsp` and `idpSSOInit.jsp` both initiate single sign-on interactions.

2. **To test, invoke the URL.**

   For more information, see **Broken Link (Target ID: GBTJX)**.

▼ To Configure Single Sign-on without Service Provider User Account

In some deployments, the service provider side of an interaction might not store user accounts. The single sign-on solution is for all identity provider user accounts to be mapped to one service provider user account. Any attributes inside the `AttributeStatement` will be set as properties of the single sign-on token. The following procedure maps an identity provider user to a service provider `anonymous` user and passes two attributes to the service provider.

1 **Export the identity provider's current extended metadata configuration to a file.**

`saml2meta [-i` *staging-directory*] **`export -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-x`** *IDP-extended-XML-file-name*

2 **Edit `attributeMap` in the exported extended metadata configuration file.**

`attributeMap` defines the mapping between the provider that this metadata is configuring and the remote provider. This attribute takes a value of *autofedAttribute-value*=*remote-provider-attribute*. For example:

```
<Attribute name="attributeMap">
<Value>mail=mail</Value>
<Value>employeeNumber=employeeNumber</Value>
</Attribute>
```

3 **Remove the identity provider's current extended metadata configuration.**

`saml2meta [-i` *staging-directory*] **`delete -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-c`**

4 **Import the identity provider's modified extended metadata configuration file.**

`saml2meta [-i` *staging-directory*] **`import -u amadmin -w`** *password* **`-x`** *IDP-extended-XML-file-name*

5 **Restart the web container.**

6 **Export the service provider's current extended metadata configuration to a file.**

`saml2meta [-i` *staging-directory*] **`export -u amadmin -w`** *password* **`-e`** *SP-entityID* **`-x`** *SP-extended-XML-file-name*

7 **Edit the following attributes in the exported extended metadata configuration file.**

- `transientUser` will take a value of one of the existing transient user identifiers on the service provider side, for example, `anonymous`.
- `attributeMap` defines the mapping between the provider that this metadata is configuring and the remote provider. This attribute takes a value of *autofedAttribute_value*=*remote_provider_attribute*. For example:

  ```
  <Attribute name="attributeMap">
  <Value>mail=mail</Value>
  <Value>employeeNumber=employeeNumber</Value>
  </Attribute>
  ```

8 **Remove the service provider's current extended metadata configuration.**

`saml2meta [-i` *staging-directory*] **`delete -u amadmin -w`** *password* **`-e`** *SP-entityID* **`-c`**

9 **Import the service provider's modified extended metadata configuration file.**

saml2meta [-i *staging-directory*] **import -u amadmin -w** *password* **-x** *SP-extended-XML-file-name*

10 **Restart the web container.**

11 **To test, invoke the single sign-on URL with the** `NameIDFormat=transient` **query parameter appended to it.**

All identity provider users will be mapped to `anonymous` on the service provider side. `mail` and `employeeNumber` will be set as properties in the identity provider user's single sign-on token. For more information on the single sign-on URL, see **Broken Link (Target ID: GBTJX)**.

Auto-Federation

The auto-federation feature will automatically federate a user's disparate provider accounts based on a common attribute defined in the interacting provider's metadata. (It is also referred to as *attribute federation*.) This common attribute, when exchanged in a single sign-on assertion, would identify the user at both provider sites and automatically create the appropriate federations. The following sections describe procedures for auto-federation.

- **Broken Link (Target ID: GBTCQ)**
- **Broken Link (Target ID: GBTEK)**

Note – Auto-federation with the transient name identifier can also be configured as described in **Broken Link (Target ID: GBSZL)**.

▼ To Enable Auto-Federation

Before You Begin You must configure the attribute mapper on the identity provider side to include the common attribute as part of the `AttributeStatement`. You must also configure the attribute mapper on the service provider side to use the common attribute to find the user.

Note – You can also configure the account mapper on the service provider side to map all users to a single user (such as `anonymous`).

---

1 **Export the identity provider's current extended metadata configuration to a file.**

saml2meta [-i *staging-directory*] **export -u amadmin -w** *password* **-e** *IDP-entityID* **-x** *IDP-extended-XML-file-name*

2 **Edit the following attributes in the exported extended metadata configuration file.**

- `autofedEnabled` takes a value of `true`.

- `autofedAttribute` defines the common attribute. For example, `<Value>employeeNumber</Value>`
- `attributeMap` defines the mapping between the provider that this metadata is configuring and the remote provider. This attribute takes a value of *autofedAttribute-value***=***remote-provider-attribute*. For example:

  ```
  <Attribute name="attributeMap">
  <Value>employeeNumber=employeeNumber</Value>
  </Attribute>
  ```

3 **Remove the identity provider's current extended metadata configuration.**

`saml2meta [-i` *staging-directory***`] delete -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-c`**

4 **Import the identity provider's modified extended metadata configuration file.**

`saml2meta [-i` *staging-directory***`] import -u amadmin -w`** *password* **`-x`** *IDP-extended-XML-file-name*

5 **Restart the web container.**

6 **Repeat the above steps to modify the service provider's extended metadata.**

7 **To test, invoke single sign-on from the service provider.**

Following the auto-federation, two SAML v2 attributes and corresponding values are written to the user's data store entry.

▼ To Configure Single Sign-on Without Data Store Writes

This interaction uses auto-federation with the transient name identifier. There is one-to-one mapping between user accounts configured with the identity provider and the service provider based on the value of one attribute. The following procedure describes how to configure single sign-on without writing to the user's data store entry.

1 **Export the identity provider's current extended metadata configuration to a file.**

`saml2meta [-i` *staging-directory***`] export -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-x`** *IDP-extended-XML-file-name*

2 **Edit the following attributes in the exported extended metadata configuration file.**

- `autofedEnabled` takes a value of `true`.
- `autofedAttribute` defines the common attribute on the identity provider side. For example, `mail`.

- `attributeMap` defines the mapping between the identity provider's attribute and the remote provider's attribute. It takes a value of *autofedAttribute-value*=*remote-provider-attribute*. For example:

  ```
  <Attribute name="attributeMap">
  <Value>mail=mail</Value>
  </Attribute>
  ```

3. **Remove the identity provider's current extended metadata configuration.**

   `saml2meta [-i` *staging-directory*] **`delete -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-c`**

4. **Import the identity provider's modified extended metadata configuration file.**

   `saml2meta [-i` *staging-directory*] **`import -u amadmin -w`** *password* **`-x`** *IDP-extended-XML-file-name*

5. **Restart the web container.**

6. **Export the service provider's current extended metadata configuration to a file.**

   `saml2meta [-i` *staging-directory*] **`export -u amadmin -w`** *password* **`-e`** *SP-entityID* **`-x`** *SP-extended-XML-file-name*

7. **Edit the following attributes in the exported extended metadata configuration file.**

   - `transientUser` takes a null value.
   - `autofedEnabled` takes a value of `true`.
   - `autofedAttribute` defines the common attribute. For example, `mail`.
   - `attributeMap` defines the mapping between the provider that this metadata is configuring and the remote provider. This attribute takes a value of *autofedAttribute-value*=*remote-provider-attribute*. For example:

     ```
     <Attribute name="attributeMap">
     <Value>mail=mail</Value>
     </Attribute>
     ```

8. **Remove the service provider's current extended metadata configuration.**

   `saml2meta [-i` *staging-directory*] **`delete -u amadmin -w`** *password* **`-e`** *SP-entityID* **`-c`**

9. **Import the service provider's modified extended metadata configuration file.**

   `saml2meta [-i` *staging-directory*] **`import -u amadmin -w`** *password* **`-x`** *SP-extended-XML-file-name*

10. **Restart the web container.**

11 **To test, invoke the single sign-on URL with the `NameIDFormat=transient` query parameter appended to it.**

All identity provider users will be mapped to the corresponding user on the service provider side based on the `mail` attribute but the auto-federation attributes will not be written to the user entry.

Auto-creation of User Accounts

Auto-creation of user accounts can be enabled on the service provider side. An account would be created when there is none corresponding to the identity provider user account requesting access. This might be necessary, for example, when a new service provider has joined an existing circle of trust.

Note – Auto-creation is supported only when the service provider is running on an instance of Federated Access Manager as it extends that product's Dynamic Profile Creation functionality.

▼ To Enable Auto-creation

Before You Begin You must configure the attribute mapper on the identity provider side to include an `AttributeStatement` from the user. The account mapper on the service provider side will perform user mapping based on the `AttributeStatement`.

1 **Export the identity provider's current extended metadata configuration to a file.**

`saml2meta [-i` *staging-directory*] **`export -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-x`** *IDP-extended-XML-file-name*

2 **Edit the following attributes in the exported extended metadata configuration file.**

- `autofedEnabled` takes a value of `true`.
- `autofedAttribute` defines the common attribute. For example, `<Value>employeeNumber</Value>`
- `attributeMap` defines the mapping between the provider that this metadata is configuring and the remote provider. This attribute takes a value of *autofedAttribute-value***=***remote-provider-attribute*. For example:

```
<Attribute name="attributeMap">
<Value>employeeNumber=employeeID</Value>
</Attribute>
```

3 **Remove the identity provider's current extended metadata configuration.**

`saml2meta [-i` *staging-directory*] **`delete -u amadmin -w`** *password* **`-e`** *IDP-entityID* **`-c`**

4 **Import the identity provider's modified extended metadata configuration file.**

saml2meta [-i *staging-directory***] import -u amadmin -w** *password* **-x** *IDP-extended-XML-file-name*

5 **Restart the web container.**

6 **Repeat the above steps to modify the service provider's extended metadata.**

7 **Enable Dynamic Profile Creation using the Federated Access Manager console.**

a. **Log in to the Federated Access Manager console as the top-level administrator, by default,** `amadmin`**.**

b. **Under the Access Control tab, select the appropriate realm.**

c. **Select the Authentication tab.**

d. **Select Advanced Properties.**

e. **Set User Profile to Dynamic or Dynamic with User Alias and click Save.**

f. **Log out of Federated Access Manager.**

8 **To test, invoke single sign-on from the service provider.**

For more information, see the *Sun Java System Access Manager 7 2005Q4 Administration Guide*.

Using Non-Default Federation Attributes

If Federated Access Manager is retrieving data from an LDAPv3–compliant directory, the object class `sunFMSAML2NameIdentifier` (containing two allowed attributes, `sunfm-saml2-nameid-info` and `sun-fm-saml2-nameid-infokey`) needs to be loaded into the entries of all existing users. When the directory contains a large user database the process is time-intensive. The following procedure can be used to modify your SAML v2 Plug-in for Federation Services installation to use existing LDAP attributes to store user federation information. In this case, there is no need to change the schema.

▼ To Store Federation Information in Existing Attributes

1 **Modify the values of the following properties in** `AMConfig.properties` **to reflect the existing attributes to which you want federation information written:**

- `com.sun.identity.saml2.nameidinfo.attribute`
- `com.sun.identity.saml2.nameidinfokey.attribute`

Note – `AMConfig.properties` is located in the `/etc/opt/`*product-directory*`/config` directory in Federated Access Manager.

2 **Restart the web container.**

Federation information will now be written to the specified attributes.

Enabling XML Signing and Encryption

This section contains the procedure to enable XML signing and encryption.

Caution – If you are modifying your organization's metadata remember to contact all providers in the circle of trust with the new metadata.

To Enable XML Signing and Encryption

1 **In** `AMConfig.properties`**, set** `com.sun.identity.jss.donotInstallAtHighestPriority` **equal to** `true`**.**

Note – `AMConfig.properties` is located in the `/etc/opt/`*product-directory*`/config` directory in Federated Access Manager.

2 **Follow the instructions in the XMLSIG sample to setup a keystore and import the signing and encryption certificates to the keystore.**

In Federated Access Manager, the sample is located in the `/`*FederatedAccessManager-base*`/`*product-directory*`/samples/saml/xmlsig` directory.

Note – The certificate alias assigned during this process will be used in the following steps to identify the certificate.

3 **Regenerate the metadata files so that they include the signing and encryption key information.**

- For identity provider metadata, run the following command:

  `saml2meta template [-i` *war-staging*`] -u` *admin* `-w` *admin-password* `-d` *idp-metaAlias* `-b` *idp-signing-key-alias* `-g` *idp-encryption-key-alias* `-e` *idp-entityID* `-m` *standard-XML-file-name* `-x` *extended-XML-file-name*

  For example:

  ```
  saml2meta template -u amadmin -w 11111111 -d /idp -b test -g test -e
  idp.sun.com -m idpMeta.xml -x idpExt.xml
  ```

- For service provider metadata, run the following command:

  `saml2meta template [-i` *war-staging*`] -u` *admin* `-w` *admin-password* `-s` *sp-metaAlias* `-a` *sp-signing-key-alias* `-f` *sp-encryption-key-alias* `-e` *sp-entityID* `-m` *standard-XML-file-name* `-x` *extended-XML-file-name*

  For example:

  ```
  saml2meta template -u amadmin -w 11111111 -s /idp -a test -f test -e
  sp.sun.com -m spMeta.xml -x spExt.xml
  ```

4 **Enable the appropriate XML signing and encryption features by modifying the generated metadata files.**

Note – XML signing is required for the Web Browser POST Profile.

You can turn on XML signing and encryption features by changing the value of the following attributes to `true`:

- Identity Provider Standard Metadata Configuration File Attribute
  - `wantAuthnRequestsSigned`
- Service Provider Standard Metadata Configuration File Attributes
  - `AuthnRequestsSigned`
  - `WantAssertionsSigned`
- Identity Provider Extended Metadata Configuration File Attributes
  - `wantNameIDEncrypted`
  - `wantArtifactResolveSigned`
  - `WantLogoutRequestSigned`
  - `WantLogoutResponseSigned`
  - `WantMNIRequestSigned`
  - `WantMNIResponseSigned`
- Service Provider Extended Metadata Configuration File Attributes
  - `wantAttributeEncrypted`
  - `wantAssertionEncrypted`
  - `wantNameIDEncrypted`
  - `wantArtifactResponseSigned`
  - `WantLogoutRequestSigned`
  - `WantLogoutResponseSigned`
  - `WantMNIRequestSigned`
  - `WantMNIResponseSigned`

5 **Remove the hosted identity provider metadata by running the following command:**

`saml2meta delete -u amadmin -w` *admin-password* `-e` *idp-entityID*

6 **Import the new hosted identity provider metadata by running the following command:**

`saml2meta import -u amadmin -w` *admin-password* `-m` *standard-XML-file-name* `-x` *extended-XML-file-name* `-t` *COT-name*

7 **Remove the remote service provider metadata by running the following command:**

`saml2meta delete -u amadmin -w` *admin-password* `-e` *sp-entityID*

8 **Get the new remote service provider metadata.**

The instructions in this step assume a testing environment where you are in control of both the identity provider server and the service provider server.

a. **Copy** `spMeta.xml` **and** `spExtended.xml` **from the service provider machine.**

b. **Change hosted="1" to** `hosted="0"` **in** `spExtended.xml`**.**

9 **Import the new remote service provider metadata by running the following command:**

`saml2meta import -u amadmin -w` *admin-password* `-m` *standard-XML-file-name* `-x` *extended-XML-file-name* `-t` *COT-name*

10 **Remove the remote identity provider metadata by running the following command:**

`saml2meta delete -u amadmin -w` *admin-password* `-e` *idp-entityID*

11 **Get the new remote identity provider metadata.**

The instructions in this step assume a testing environment where you are in control of both the identity provider server and the service provider server.

a. **Copy** `idpMeta.xml` **and** `idpExtended.xml` **from the identity provider machine.**

b. **Change hosted="1" to** `hosted="0"` **in** `idpExtended.xml`**.**

12 **Import the new remote identity provider metadata by running the following command:**

`saml2meta import -u amadmin -w` *admin-password* `-m` *standard-XML-file-name* `-x` *extended-XML-file-name* `-t` *COT-name*

13 **Remove the hosted service provider metadata by running the following command:**

`saml2meta delete -u amadmin -w` *admin-password* `-e` *sp-entityID*

14 **Import the new hosted service provider metadata by running the following command:**

`saml2meta import -u amadmin -w` *admin-password* `-m` *standard-XML-file-name* `-x` *extended-XML-file-name* `-t` *COT-name*

15 **Restart your web container.**

Securing SOAP Binding

SOAP binding supports the following authentication methods to protect interactions between SAML v2 entities:

- **Broken Link (Target ID: GBTLN)**
- **Broken Link (Target ID: GBTKP)**

Basic Authentication

Once basic authentication is set up to protect a SAML v2 SOAP endpoint, all entities communicating with this endpoint must configure three basic authentication-related attributes in the extended metadata as described in the following table.

TABLE 10–3 Securing SOAP Endpoint with Basic Authentication

Attribute	Description
`basicAuthOn`	Establishes that the SOAP endpoint is using basic authentication. Takes a value of `true` or `false`.
`basicAuthUser`	Defines the user allowed access to the protected SOAP endpoint in the original SAML v2 entity.
`basicAuthPassword`	Defines an encrypted password for the user. The password is encrypted using `ampassword` on the partner side. For information on `ampassword`, see *Sun Java System Access Manager 7 2005Q4 Administration Guide*.

To modify the metadata, you must first export it to a file. Once you've modified the values of the applicable attributes, the metadata must be reloaded using the `saml2meta` command and the web container must be restarted. For more information, see **Broken Link (Target ID: GBSBB)**.

Secure Socket Layer/Transport Layer Security

Secure Socket Layer/Transport Layer Security (SSL/TLS) can also be enabled to protect SOAP endpoints and secure communications between SAML v2 entities. When one SAML v2 entity initiates communication with a SAML v2 entity deployed in an SSL/TLS-enabled web container, the initiating entity is referred to as the SSL/TLS client and the replying entity is referred to as the SSL/TLS server.

Server Certificate Authentication

For SSL/TLS server authentication (the server needs to present a certificate to the client), the following properties need to be set in the Virtual Machine for the Java^{TM} platform (JVM^{TM}) running the SSL/TLS client:

`-Djavax.net.ssl.trustStore`	Defines the full path to the file containing the server's CA certificate(s).
`-Djavax.net.ssl.trustStoreType`	Takes a value of `JKS` (Java Key Store).

In addition, the client's CA certificate needs to be imported into the certificate store/database of the server's web container and marked as a trusted issuer of client certificates.

Client Certificate Authentication

For SSL/TLS client authentication (the client needs to present a certificate to the server), the following properties need to be set in the JVM software running the SSL/TLS client:

`-Djavax.net.ssl.keyStore`	Defines the full path to the keystore containing the client certificate and private key. This may be the same as that defined in **Broken Link (Target ID: GBTKZ)**.
`-Djavax.net.ssl.keyStoreType`	Takes a value of `JKS`.
`-Djavax.net.ssl.keyStorePassword`	Specifies the password to the keystore.

On the SSL/TLS server side, the client's CA certificate needs to be imported into the web container's keystore and marked as a trusted issuer of client certificates.

Load Balancing

In cases of large deployments, a load balancer can be put in front of multiple instances of Access Manager that have the SAML v2 Plug-in for Federation Services installed. The following procedure describes how to enable load balancer support.

▼ To Enable Load Balancer Support

1 **Install Federated Access Manager and follow the documentation to set up a load balancer.**

Load balancing information for Federated Access Manager can be found in *Sun Java System Access Manager 7 2005Q4 Deployment Planning Guide*.

2 **Install the SAML v2 Plug-in for Federation Services on any machines acting as an identity provider or a service provider.**

3 **On any service provider machines, copy the metadata configuration files into the same directory and rename as follows:**

- `spMeta.xml` to `spMeta.xml.lb`

- `spExtended.xml` to `spExtended.xml.lb`

4 **Edit the new service provider load balancer metadata configuration files as follows:**

- Change the host name of the service provider to that of the load balancer on the service provider side.
- Change the port of the service provider to that of the load balancer on the service provider side.
- Change the `metaAlias` of the service provider to any new `metaAlias`, for example, `/splb`.

5 **On any identity provider machines, copy the metadata configuration files into the same directory and rename as follows:**

- `idpMeta.xml` to `idpMeta.xml.lb`
- `idpExtended.xml` to `idpExtended.xml.lb`

6 **Edit the new identity provider load balancer metadata configuration files as follows:**

- Change the host name of the identity provider to that of the load balancer on the identity provider side.
- Change the port of the identity provider to that of the load balancer on the identity provider side.
- Change the `metaAlias` of the identity provider to any new `metaAlias`, for example, `/idplb`.

7 **Import the new hosted metadata onto the service provider machines.**

8 **Import the new remote identity provider metadata onto the service provider machines.**

9 **Import the new hosted metadata onto the identity provider machines.**

10 **Import the new remote service provider metadata onto the identity provider machines.**

11 **Restart the web containers.**

Access Control

The following procedure will allow user access on the service provider side based on the user's configured roles on the identity provider side. This information is passed to the service provider in an assertion. No matching user entry is necessary on the service provider side.

▼ To Enable Access Control Using Agents and Roles

- You must configure the agent to enforce policy. For example, setting `com.sun.am.policy.agents.config.do_sso_only=false`.

- You must configure the attribute mapper on the identity provider side to include the required attributes in the `AttributeStatment`.
- You must configure the service provider to set the received attributes as key/value pairs in the user's single sign-on token. The agent will set those attributes in the HTTP header, passing them down to the application.

1 **Install the SAML v2 Plug-in for Federation Services on the identity provider.**

2 **Install the SAML v2 Plug-in for Federation Services on the service provider.**

3 **Install the Sun Policy Agents 2.2 to protect the service provider configured on the instance of Federated Access Manager.**

For more information, see the *Sun Java System Access Manager Policy Agent 2.2 User's Guide*.

4 **Modify `com.sun.am.policy.am.login.url` in `AMAgent.properties` so that its value is a URL (appended with the `NameIDFormat=transient` query parameter) that points to a single sign-on JSP on the service provider side.**

```
com.sun.am.policy.am.login.url=SP-protocol://SP-host:SP-port/service-deploy-uri/
saml2/jsp/spSSOInit.jsp?NameIDFormat=transient&metaAlias=SP-metaAlias&
idpEntityID=IDP_EntityID
```

For example:

```
com.sun.am.policy.am.login.url=http://moonriver.red.sun.com:58080/
amserver/saml2/jsp/spSSOInit.jsp?NameIDFormat=transient&metaAlias=/sp&
idpEntityID=sunriver.central.sun.com
```

5 **(Required only if using Web Agent 2.1) Set the value of the `com.sun.am.policy.am.library.loginURL` property to the service provider's login URL so the agent can authenticate itself.**

If the login URL is a URL that initiates a SAML v2 single sign-on interaction, the value of this property will be used to authenticate the agent itself to your instances of Federated Access Manager. An example value might be **`http://`*`host`*`:`*`port`*`/amserver/UI/Login`**.

6 **Modify `spSSOInit.jsp` on the service provider side to use `goto` parameter as the value for `RelayState`.**

The differences are as follows:

```
***************
*** 143,148 ****
--- 143,154 ----
}
idpEntityID = request.getParameter("idpEntityID");
paramsMap = SAML2Utils.getParamsMap(request);
+ String gotoURL = (String) request.getParameter("goto");
+ if (gotoURL != null) {
+ List list = new ArrayList();
+ list.add(gotoURL);
+ paramsMap.put(SAML2Constants.RELAY_STATE, list);
+ }
if ((idpEntityID == null) || (idpEntityID.length() == 0)) {
// get reader url
```

7. **Set up single sign-on without requiring writes to the data store by following the procedure described in Broken Link (Target ID: GBTEK).**

   To test, we assume the `employeenumber` attribute stores the user's role. In addition, the identity provider should have the following configured users:

   - User 1 has `employeenumber` set to `manager` (the manager's role).
   - User 2 has `employeenumber` set to `employee` (the employee's role).

8. **Create a policy with the `SessionProperty` condition on the service provider instance of Federated Access Manager.**

   a. **Log in to the Federated Access Manager console as the top-level administrator, by default, `amadmin`.**

   b. **Under the Access Control tab, select the appropriate realm.**

   c. **Select the Policies tab.**

   d. **Click New Policy.**

   e. **Enter a name for the policy.**

   f. **Click New under Rules.**

   g. **Select URL Policy Agent (with resource name) and click Next.**

   h. **Enter a name for the rule.**

   i. **Enter the application's URL as the value for Resource Name.**

   j. **Select Allow under both GET and POST and click Finish.**

   k. **Click New under Conditions.**

   l. **Select SessionProperty and click Next.**

   m. **Enter a name for the condition.**

n. **Click Add under Values.**

o. **Enter the single sign-on token property name as the value for Property Name.**

   To test, we will use `employeenumber`.

p. **Add the match value to the Values field and click Add.**

   To test, we will use `manager`.

q. **Click Add to return to the New Condition page.**

r. **Click Finish to save the condition.**

s. **Click Create to create the policy.**

For more information on creating policy, see the *Sun Java System Access Manager 7 2005Q4 Administration Guide*.

9 **Access the application using a web browser.**

You will be redirected to the service provider single sign-on JSP defined in the previous step. From there, you will be redirected to the identity provider to login. Single sign-on with the service provider will be accomplished using SAML v2 and, finally, you will be redirected back to the application for policy enforcement. If you logged in as User 1, you will be allowed to access the application as a manager which is allowed by the policy. If you logged in as User 2, an employee, you will be denied access to the application.

Certificate Revocation List Checking

The certificate revocation list (CRL) is a list of revoked certificates that contains the reason(s) for the certificate's revocation, the date of it's issuance, and the entity that issued it. When a potential user attempts to access the Federated Access Manager server, first access is allowed or denied based on the CRL entry for the root certificate included with the request. When the SAML v2 Service receives the incoming XML request, it parses the issuer Distinguished Name (DN) from the root certificate and retrieves the value defined by the `com.sun.identity.crl.cache.directory.searchattr` attribute in `AMConfig.properties`. If the attribute value is `CN` and the issuer DN is, for example, `CN="Entrust.net Client Certification Authority", OU=...`, the SAML v2 Service uses *Entrust.net Client Certification Authority* to retrieve the CRL from the LDAP directory which acts as the CRL repository.

Note – The LDAP directory which acts as the CRL repository is also configured in `AMConfig.properties`.

With this action, one of the following will occur:

1. If the LDAP directory returns a CRL that is not valid, the SAML v2 Service retrieves the value of the `IssuingDistributionPointExtension` attribute (usually an HTTP or LDAP URI) from the CRL and uses it to get new CRL from the certificate authority. If the certificate authority returns a valid CRL, it is saved to the LDAP directory and to memory and used for certificate validation.
2. If the LDAP directory returns no CRL but the certificate that is being validated has a defined `CRL Distribution Point Extension`, the SAML v2 Service retrieves it's value (usually an HTTP or LDAP URI) and uses the value to get a new CRL from the certificate authority. If the certificate authority returns a valid CRL, it is saved to the LDAP directory and to memory and used for certificate validation.
3. If the certificate authority returns a valid CRL, it is saved to the LDAP directory and to memory and used for certificate validation.

Note – Currently, Certificate Revocation List Checking works only with an instance of Sun Directory Server.

After the CRL is loaded into memory and the root certificate validation is successful, the single sign-on process continues with validation of the signed XML message. The following are procedures to set up the SAML v2 Service for CRL checking.

- **Broken Link (Target ID: GEZZG)**
- **Broken Link (Target ID: GFABI)**

Caution – CRL checking currently only works in the case of XML-based signature validation; for example, service provider side POST Artifact profile, or SOAP based logout. CRL checking does not work in the case of URL string based signature validation, XML signing, XML encryption or decryption.

▼ To Set Up for Certificate Revocation List Checking

Before You Begin A local instance of Directory Server must be designated as the CRL repository. It can be the same directory in which the Federated Access Manager schema is stored or it can be standalone. The Java Development Kit (JDK) must be version 1.5 or higher.

1 Create one entry in Directory Server for each certificate authority.

For example, if the certificate authority's `subjectDN` is `CN="Entrust.net Client Certification Authority",OU="www.entrust.net/GCCA_CPS incorp. by ref. (limits lib.)",O=Entrust.net` and the base DN for Directory Server is `dc=sun,dc=com`, create an entry with the DN `cn="Entrust.net Client Certification Authority",ou=people,dc=sun,dc=com`.

Note – If the certificate authority's `subjectDN` does not contain `uid` or `cn` attributes, do the following:

a. **Create a new object class.**

   For example, `sun-am-managed-ca-container`.

b. **Populate the new object class with the following attributes:**

   - `objectclass`
   - `ou`
   - `authorityRevocationList`
   - `caCertificate`
   - `certificateRevocationList`
   - `crossCertificatePair`

c. **Add the following entry (modified per your deployment) to Directory Server.**

```
dn: ou=1CA-AC1,dc=sun,dc=com
objectClass: top
objectClass: organizationalunit
objectClass: iplanet-am-managed-ca-container
ou: 1CA-AC1
```

You will publish the appropriate CRL to the entry created in the last step.

2 **Publish the appropriate CRL to the corresponding LDAP entry.**

This part can be done automatically by Federated Access Manager or manually. If the certificate being validated has a `CRL Distribution Point Extension` value, the publishing of the CRL is done automatically. If the certificate being validated has an `IssuingDistributionPointExtension` value, the initial publishing of the CRL must be done manually but future updates are done in runtime. If the certificate being validated has neither of these values, updates must be done manually at all time. See **Broken Link (Target ID: GFABI)** for information on manual population.

3 **Configure the following properties in** `AMConfig.properties` **to point to the instance of Directory Server designated as the CRL repository.**

- `com.sun.identity.saml2.crl.cache.directory.host` defines the LDAP directory's host name.
- `com.sun.identity.saml2.crl.cache.directory.port` defines the LDAP directory's port number.
- `com.sun.identity.saml2.crl.cache.directory.ssl` takes a vale of TRUE or FALSE.
- `com.sun.identity.saml2.crl.cache.directory.user` defines the DN of the user with permission to bind to the LDAP directory.

- `com.sun.identity.saml2.crl.cache.directory.password` defines the encrypted password for the bind user. Use `ampassword` for the encryption. SeeChapter 2, “The ampassword Command Line Tool,” in *Sun Java System Access Manager 7.1 Administration Reference* for more information.
- `com.sun.identity.saml2.crl.cache.directory.searchloc` defines the base DN from where the search will begin.
- `com.sun.identity.saml2.crl.cache.directory.searchattr` defines the component of the root certificate's `subjectDN` (issuer) that will be used to retrieve the CRL from LDAP directory. The value is a single string as in `cn`.

Note – All root certificate authorities must use the same search attribute.

com.sun.identity.saml2.crl.cache.directory.password defines the password for the bind user. This actually need to be the encrypted password of the bind user, customer need to use ampassword to encrypt the password before putting values here.

4 **Import all the certificate authority certificates into the `cacerts` keystore under the `java.home/jre/lib/secure` directory using the `keytool` utility.**

Certificates must be imported as `trustedcacert`. More information on `keytool` can be found at `http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/keytool.html`.

▼ To Manually Populate a Directory Server with a Certificate Revocation List

1 **Use your browser to get the initial CRL from the certificate authority manually.**

2 **Save the initial CRL file in the binary DER format to your local machine.**

3 **Convert the DER file to the text-based PEM format and finally LDAP Data Interchange Format (LDIF) using the following command:**

`ldif -b` *certificaterevocationlist;binary* **`< famouseCA.crl > crl.ldif`**

Note – The `ldif` command is available in your Directory Server installation.

The `crl.ldif` file contains text similar to the following:

```
certificaterevocationlist;binary:: MIH7MIGmMA0GCSqGSIb3DQEBBQUAMGExCzAJBgNVBA
  YTAlVTMRgwFgYDVQQKEw9VLlMuIEdvdmVybm1lbnQxDDAKBgNVBAsTA0RvRDEMMAoGA1UECxMDUE
  tJMRwwGgYDVQQDExNEb0QgQ2xhc3MgMyBSb290IENBFw0wNzA1MDExNDMzMDNaFw0wNzA1MDExNz
  UzMDNaMBQwEgIBTxcNMDcwNDI3MTY1NzMzWjANBgkqhkiG9w0BAQUFAANBADUd7lBe7JeQKQnKCK
  GddnsCXqii7EitbPuYT55M4Nn3qBgPFSG8bX9H5XBGTB4iofb3h0Y9DCqe10vc8dBM0
```

4 **Do one of the following to define the LDAP entry in which the CRL will be stored.**

- **For an existing entry, specify the DN in the LDIF file.**

  ```
  # entry-id: famouseCA dn: CN=famouseCA,ou=People,dc=sun,dc=com
  certificaterevocationlist;binary:: MIH7MIGmMA0GCSqGSIb3DQEBBQUAMGExCzAJBgNVBA
    YTAlVTMRgwFgYDVQQKEw9VLlMuIEdvdmVybm1lbnQxDDAKBgNVBAsTA0RvRDEMMAoGA1UECxMDUE
    tJMRwwGgYDVQQDExNEb0QgQ2xhc3MgMyBSb290IENBFw0wNzA1MDExNDMzMDNaFw0wNzA1MDExNz
    UzMDNaMBQwEgIBTxcNMDcwNDI3MTY1NzMzWjANBgkqhkiG9w0BAQUFAANBADUd7lBe7JeQKQnKCK
    GddnsCXqii7EitbPuYT55M4Nn3qBgPFSG8bX9H5XBGTB4iofb3h0Y9DCqe10vc8dBM0
  ```

- **For a new entry, specify the DN and object classes in the LDIF file.**

  ```
  # entry-id: tester200
  dn: CN=famouseCA,ou=People,dc=sun,dc=com
  sn: famouseCA
  cn: famouseCA
  employeeNumber: 1001
  telephoneNumber: 555-555-5555
  postalAddress: 555 Test Drive
  iplanet-am-modifiable-by: cn=Top-level Admin Role,dc=iplanet,dc=com
  mail: famouseCA@test.com
  givenName: Test
  inetUserStatus: Active
  uid: tester200
  objectClass: iplanet-am-user-service
  objectClass: inetAdmin
  objectClass: iPlanetPreferences
  objectClass: inetOrgPerson
  objectClass: organizationalPerson
  objectClass: person
  objectClass: iplanet-am-managed-person
  objectClass: inetuser
  objectClass: top
  userPassword: {SSHA}E3TJ4DT7IoOLETVny1ktxUGWNTpBYq8tj3C1Sg==
  creatorsName: cn=puser,ou=dsame users,dc=iplanet,dc=com
  modifiersName: cn=puser,ou=dsame users,dc=iplanet,dc=com
  createTimestamp: 20031125043253Z
  modifyTimestamp: 20031125043253Z
  certificaterevocationlist;binary:: MIH7MIGmMA0GCSqGSIb3DQEBBQUAMGExCzAJBgNVBA
    YTAlVTMRgwFgYDVQQKEw9VLlMuIEdvdmVybm1lbnQxDDAKBgNVBAsTA0RvRDEMMAoGA1UECxMDUE
    tJMRwwGgYDVQQDExNEb0QgQ2xhc3MgMyBSb290IENBFw0wNzA1MDExNDMzMDNaFw0wNzA1MDExNz
    UzMDNaMBQwEgIBTxcNMDcwNDI3MTY1NzMzWjANBgkqhkiG9w0BAQUFAANBADUd7lBe7JeQKQnKCK
    GddnsCXqii7EitbPuYT55M4Nn3qBgPFSG8bX9H5XBGTB4iofb3h0Y9DCqe10vc8dBM0G8=
  ```

5 **Run one of the following `ldapmodify` commands based on whether you are adding the LDIF file to an existing entry or creating a new entry.**

- **To add a CRL to an existing LDAP entry (using an LDIF file with a specified DN), use the following command:**

  ```
  ldapmodify -r -h Directory Server_host -p Directory Server_port
  -f ldif-file -D cn=Directory Manager -w password
  ```

- **To add a CRL to a new LDAP entry (using an LDIF file with a specified DN and object classes), use the following command:**

  ```
  ldapmodify -a -h Directory Server_host -p Directory Server_port
  -f ldif-file -D cn=Directory Manager -w password
  ```

Bootstrapping the Liberty ID-WSF with SAML v2

SAML v2 can be used to bootstrap into the Liberty Alliance Project Identity Web Services Framework (Liberty ID-WSF) version 1.1. For example, a service provider communicating with the SAML v2 specifications might want to communicate with web services based on the Liberty ID-WSF regarding a principal. To do this, the SAML v2 Assertion returned to the service provider must contain a Discovery Service endpoint. The service provider than acts as a web services consumer, using the value included within the Endpoint tag to bootstrap the Discovery Service. This then allows access to other Liberty ID-WSF services.

A sample SAML v2 assertion is reproduced below. Note the SAML v2 security token stored in the Discovery Service resource offering: `urn:liberty:security:2003-08:null:SAML`. Both are stored within the attribute statement.

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" Version="2.0"
ID="s21bdfd298f332ef2ada1d4fd00bab21c0f64cc90a"
IssueInstant="2007-03-27T08:25:26Z">
<saml:Issuer>http://hengming.red.iplanet.com</saml:Issuer>
<saml:Subject>
<saml:NameID NameQualifier="http://hengming.red.iplanet.com"
  SPNameQualifier="http://isdev-2.red.iplanet.com" Format=
  "urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">
  HuCJIy9v5MdrjJQOgsuT4NWmVUl3</saml:NameID>
<saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
<saml:SubjectConfirmationData NotOnOrAfter="2007-03-27T08:35:26Z"
  InResponseTo="s20711ed113989a9bff544f61c700d0bd0a08b78fd"
  Recipient="http://isdev-2.red.iplanet.com:58080/
  amserver/Consumer/metaAlias/sp"  >
  </saml:SubjectConfirmationData>
  </saml:SubjectConfirmation>
  </saml:Subject>
<saml:Conditions NotBefore="2007-03-27T08:25:26Z"
```

```
  NotOnOrAfter="2007-03-27T08:35:26Z">
<saml:AudienceRestriction>
<saml:Audience>http://isdev-2.red.iplanet.com</saml:Audience>
  </saml:AudienceRestriction>
  </saml:Conditions>
<saml:AuthnStatement AuthnInstant="2007-03-27T08:19:24Z"
  SessionIndex="s234f01958bf364aff26829d9d9846ba51afc2b201">
  <saml:AuthnContext>
  <saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:
  2.0:ac:classes:PasswordProtectedTransport
  </saml:AuthnContextClassRef>
  </saml:AuthnContext>
  </saml:AuthnStatement>
<saml:AttributeStatement>
<saml:Attribute Name="offerings" NameFormat="urn:liberty:disco:2003-08">
<saml:AttributeValue xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
<ResourceOffering xmlns="urn:liberty:disco:2003-08">
<ResourceID xmlns="urn:liberty:disco:2003-08">http://hengming.red.iplanet.com
  /aWQ9aWRwLG91PXVzZXIsZGM9aXBsYW5ldCxkYz1jb20sYW1zZGtkbj11aWQ9aWRwLG91PXBlb3BsZ
  SxkYz1pcGxhbmV0LGRjPWNvbQ%3D%3D</ResourceID>
<ServiceInstance xmlns="urn:liberty:disco:2003-08">
<ServiceType>urn:liberty:disco:2003-08</ServiceType>
<ProviderID>http://hengming.red.iplanet.com</ProviderID>
<Description xmlns="urn:liberty:disco:2003-08"
  id="sf6a6d3dcc16e729eea0d7e5587a5ff27f234f991">
<SecurityMechID>urn:liberty:security:2003-08:null:SAML
  </SecurityMechID>
<CredentialRef>s5dc88819de075e4e9c8db3deb8b46d4d8758b4b901
  </CredentialRef>
<Endpoint>http://hengming.red.iplanet.com:58080/amserver/Liberty/disco
  </Endpoint></Description>
  </ServiceInstance></ResourceOffering></saml:AttributeValue></saml:Attribute>
<saml:Attribute Name="credentials" NameFormat="urn:liberty:disco:2003-08">
<saml:AttributeValue xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
<saml:Assertion  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  MajorVersion="1" MinorVersion="1"
  AssertionID="s5dc88819de075e4e9c8db3deb8b46d4d8758b4b901" Issuer=
  "http://hengming.red.iplanet.com" IssueInstant="2007-03-27T08:25:26Z" >
<sec:ResourceAccessStatement xmlns:sec="urn:liberty:sec:2003-08">
<saml:Subject xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
<saml:NameIdentifier NameQualifier="http://isdev-2.red.iplanet.com"
  Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">
  HuCJIy9v5MdrjJQOgsuT4NWmVUl3</saml:NameIdentifier>
<saml:SubjectConfirmation>
<saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:sendervouches
  </saml:ConfirmationMethod>
  </saml:SubjectConfirmation>
  </saml:Subject>
```

```
<ResourceID xmlns="urn:liberty:disco:2003-08">http://hengming.red.iplanet.com/
aWQ9aWRwLG91PXVzZXIsZGM9aXBsYW5ldCxkYz1jb20sYW1zZG
tkbj11aWQ9aWRwLG91PXBlb3BsZSxkYz1pcGxhbmV
0LGRjPWNvbQ%3D%3D</ResourceID>
<sec:ProxySubject xmlns:sec="urn:liberty:sec:2003-08">
<saml:NameIdentifier xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
Format="urn:liberty:iff:nameid:entityID">http://isdev-2.red.iplanet.com
  </saml:NameIdentifier>
<saml:SubjectConfirmation xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
<saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
  </saml:ConfirmationMethod>
<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#"><KeyName>CN=sun-unix, OU=Sun
  Access Manager, O=Sun, C=US</KeyName><KeyValue><RSAKeyValue><Modulus>AOA/2kpfKFWvRXOMbrmTlKe102ibw/
  aTd3HBVgI8cHsywww8M1J0X+vJvvk6eabTNWY5jBfTo9i1bC4AXXoRlxgsE/
  6Uq5+6NGrd+iwfvj25x8HzHX8LrJ+7EzlGVsKO
  M+A3vTP0tCkmYE4jatZbWlRoto0wyInP2wMFdKPrmYWL</Modulus>
<Exponent>AQAB</Exponent></RSAKeyValue>
  </KeyValue></KeyInfo></saml:SubjectConfirmation>
  </sec:ProxySubject><sec:SessionContext xmlns:sec="urn:liberty:sec:2003-08" AuthenticationInstant=
  "2007-03-27T08:25:26Z" AssertionIssueInstant="2007-03-27T08:25:26Z">
<sec:SessionSubject xmlns:sec="urn:liberty:sec:2003-08">
<saml:NameIdentifier xmlns:saml="urn:oasis:names:tc:SAML:1.0:
  assertion" NameQualifier="http://isdev-2.red.iplanet.com"
  Format="urn:oasis:names:tc:SAML:
  2.0:nameid-format:persistent">HuCJIy9v5MdrjJQOgsuT4NWmVUl3
  </saml:NameIdentifier>
<saml:SubjectConfirmation xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
<saml:ConfirmationMethod>urn:oasis:names:tc:SAML:2.0:cm:bearer</saml:ConfirmationMethod>
</saml:SubjectConfirmation>
<lib:IDPProvidedNameIdentifier  xmlns:lib="http://projectliberty.org/
  schemas/core/2002/12"
  NameQualifier="http://hengming.red.iplanet.com" Format="urn:oasis:names:tc:SAML:2.0:
  nameid-format:persistent"  >HuCJIy9v5MdrjJQOgsuT4NWmVUl3
  </lib:IDPProvidedNameIdentifier>
  </sec:SessionSubject>
<sec:ProviderID>http://hengming.red.iplanet.com</sec:ProviderID>
  <lib:AuthnContext xmlns:lib="urn:liberty:iff:2003-08"><lib:AuthnContextClassRef>urn:oasis:
  names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport</lib:AuthnContextClassRef>
  <lib:AuthnContextStatementRef>http://www.projectliberty.org/schemas/authctx/classes/
  Password</lib:AuthnContextStatementRef></lib:AuthnContext></sec:SessionContext>
  </sec:ResourceAccessStatement>
<saml:AuthenticationStatement xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
  AuthenticationInstant="2007-03-27T08:19:24Z">
<saml:Subject>
<saml:NameIdentifier Format="urn:liberty:iff:nameid:entityID">
  http://isdev-2.red.iplanet.com</saml:NameIdentifier>
<saml:SubjectConfirmation>
```

```
<saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
  </saml:ConfirmationMethod>
<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#"><KeyName>CN=sun-unix, OU=Sun
  Access Manager, O=Sun, C=US</KeyName><KeyValue><RSAKeyValue><Modulus>AOA/2kpfKFWvRXOMbrmTlKe102ibw/
  aTd3HBVgI8cHsywww8M1J0X+vJvvk6eabTNWY5jBfTo9i1bC4AXXoRlxgsE/6Uq
  5+6NGrd+iwfvj25x8HzHX8LrJ+7EzlGVsKOM+
  A3vTP0tCkmYE4jatZbWlRoto0wyInP2wMFdKPrmYWL</Modulus>
<Exponent>AQAB</Exponent>
  </RSAKeyValue>
  </KeyValue></KeyInfo></saml:SubjectConfirmation>
</saml:Subject>
</saml:AuthenticationStatement>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<Reference URI="#s5dc88819de075e4e9c8db3deb8b46d4d8758b4b901">
<Transforms>
<Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</Transforms>
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>td1CqmbWC5eMXCK6IFhzZxn3GJg=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>
YJ4g+jV5KIQRpkI9jlsZMbKx9lBhEB5ngB8NrH5nPh8+XFTK2gPZNzovOYOzxlznuxxbvC3A4rpg
UoSeE3N+oE4sl5KnY1GewFgjckAdeWafcLhGd9O68A+9nqMnRW/5fR9mnbk9eqZO8zx2bO8toiWi
pQCTU5XcDYkCNb8LgFs=
</SignatureValue>
<KeyInfo>
<X509Data>
<X509Certificate>
MIICOTCCAeOgAwIBAgIBEjANBgkqhkiG9w0BAQQFADBFMQswCQYDVQQGEwJVUzEYMBYGA1UEChMP
cmVkLmlwbGFuZXQuY29tMRwwGgYDVQQDExNDZXJ0aWZpY2F0ZSBNYW5hZ2VyMB4XDTA1MDYwMjE3
NTkxOFoXDTA2MDYwMjE3NTkxOFowVzELMAkGA1UEBhMCVVMxDDAKBgNVBAoTA1N1bjEnMCUGA1UE
CxMeU1VOIEphdmEgU3lzdGVtIEFjY2VzcyBNYW5hZ2VyMREwDwYDVQQDEwhzdW4tdW5peDCBnzAN
BgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA4D/aSl8oVa9Fc4xuuZOUp7XTaJvD9pN3ccFWAjxwezLD
DDwzUnRf68m++Tp5ptM1ZjmMF9Oj2LVsLgBdehGXGCwT/pSrn7o0at36LB++PbnHwfMdfwusn7sT
OUZWwo4z4De9M/S0KSZgTiNq1ltaVGi2jTDIic/bAwV0o+uZhYsCAwEAAaNoMGYwEQYJYIZIAYb4
QgEBBAQDAgZAMA4GA1UdDwEB/wQEAwIE8DAfBgNVHSMEGDAWgBQqOASyzZ41LergF+cSQN1Gokpa
XjAgBgNVHREEGTAXgRVoZW5nLW1pbmcuaHN1QHN1bi5jb20wDQYJKoZIhvcNAQEEBQADQQDKxdPy
821aQRVZ0wLqa6LBYZCUcZD5AMvzl3EylwtniHmzPtOeZe4NmFj7qQziSb1H57NSkiwKaLZ7Mt6F
jaUU
</X509Certificate>
</X509Data>
</KeyInfo>
</Signature></saml:Assertion>
</saml:AttributeValue></saml:Attribute></saml:AttributeStatement></saml:Assertion>
```

Following are the procedures to enable bootstrapping of the Liberty ID-WSF Discovery Service using SAML v2.

- **Broken Link (Target ID: GFADH)**
- **Broken Link (Target ID: GFAHQ)**

▼ To Enable an Identity Provider for SAML v2 Bootstrapping of Liberty ID-WSF

Before You Begin See **Broken Link (Target ID: GBSBB)** for more information on the command line interface used in this procedure.

1 Choose one of the following options to get metadata for the appropriate identity provider.

The option you choose is dependent on where you are in the process of configuring the identity provider.

- **If metadata for the identity provider you are configuring has not yet been imported, or signing and encryption certificate aliases have not been configured in the existing identity provider metadata, generate standard and extended metadata templates for the identity provider using the `saml2meta` command line interface.**

  ```
  saml2meta template -u amadmin -w amadmin_pw -d /idp
  -b certificate_alias -g enc_certificate_alias -e http://host_machine
  -m standard_meta_filename -x extended_meta_filename
  ```

- **If the identity provider metadata has been imported, and signing and encryption keys have all been configured, export the existing extended entity configuration metadata of the identity provider using the `saml2meta` command line interface.**

  ```
  saml2meta export -u amadmin -w amadmin_pw -d /idp
   -e http://host_machine -x extended_meta_filename
  ```

2 Edit the identity provider's extended entity configuration template by changing the value of `discoveryBootstrappingEnabled` to `true`.

The extended entity configuration template is *extended_meta_filename* created in the previous step. If the attribute doesn't exist in the metadata, add the following lines above the ending tag `</IDPSSOConfig>`.

```
<Attribute name="discoveryBootstrappingEnabled">
<Value>true</Value>
</Attribute>
```

3 **(Optional) Delete the current metadata for the identity provider using the `saml2meta` command line interface.**

The option you choose in this step is dependent on the option chosen in the first step of this procedure.

- **If you choose the first option in this procedure's first step, delete the current standard and extended metadata using the `saml2meta` command line interface.**

  ```
  saml2meta delete -u amadmin -w amadmin_pw -e http://host_machine
  ```

- **If you choose the second option in this procedure's first step, delete the current extended metadata only using the `saml2meta` command line interface.**

  ```
  saml2meta delete -u amadmin -w amadmin_pw -e http://host_machine -c
  ```

4 **Import the new identity provider metadata.**

The option you choose in this step is dependent on the option chosen in the first step of this procedure. *circle_of_trust* is the name of the circle of trust into which you are importing these files.

- **If you choose the first option in this procedure's first step, import the standard metadata and the modified extended metadata files using the `saml2meta` command line interface.**

  ```
  saml2meta import -u amadmin -w amadmin_pw
  -m standard_meta_filename -x extended_meta_filename -t circle_of_trust
  ```

- **If you choose the second option in this procedure's first step, import the modified extended metadata using the `saml2meta` command line interface.**

  ```
  saml2meta import -u amadmin -w amadmin_pw
   -x extended_meta_filename -t circle_of_trust
  ```

5 **Add the following line to the end of the `AMConfig.properties` file to enable Liberty ID-WSF to work with SAML v2 on the identity provider.**

```
com.sun.identity.liberty.ws.uti.providerManagerClass=com.sun.identity.saml2.plugins.S
```

6 **Restart the web container.**

▼ To Enable a Service Provider for SAML v2 Bootstrapping of Liberty ID-WSF

1 **Add the following line to the end of the `AMConfig.properties` file to enable Liberty ID-WSF to work with SAML v2 on the identity provider.**

```
com.sun.identity.liberty.ws.uti.providerManagerClass=com.sun.identity.saml2.plugins.S
```

2 **(Optional) Add the following to the class path of the web application.**

```
/opt/SUNWam/saml2/lib/saml2.jar
```

This step is necessary only if the web application you are protecting is using the same Java Virtual Machine (JVM) as the instance of Federated Access Manager. In this situation, you can use the following API to retrieve the Discovery Service bootstrap resource offering and included security token

```
ResourceOffering SAML2SDKUtil.getDiscoveryBootStrapResourceOffering(
       HttpServletRequest request)
List SAML2SDKUtil.getDiscoveryBootStrapCredentials(
       HttpServletRequest request)
```

3 **Restart the web container.**

PART III

Directory Management and Default Services

This is part three of the *Sun Federated Access Manager 8.0 Administration Guide*. The Directory Management chapter describes how to manage Directory objects when Federated Access Manager is deployed in Legacy Mode. The other chapters describe how to configure and use some of Federated Access Manager's default services. This part contains the following chapters:

◆ ◆ ◆ CHAPTER 11

Directory Management

The Directory Management tab is only displayed when you install Federated Access Manager in Legacy mode. This directory management feature provides an identity management solution for Sun Directory Server-enabled Federated Access Manager deployments.

For more information on the Legacy Mode installation option, see the *Sun Java Enterprise System 5 Installation Guide for UNIX*

Managing Directory Objects

The Directory Management tab contains all the components needed to view and manage the Directory Server objects. This section explains the object types and details how to configure them. User, role, group, organization, sub organization and container objects can be defined, modified or deleted using either the Federated Access Manager console or the command line interface. The console has default administrators with varying degrees of privileges used to create and manage the directory objects. (Additional administrators can be created based on roles.) The administrators are defined within the Directory Server when installed with Federated Access Manager. The Directory Server objects you can manage are:

- "Organizations" on page 275
- "Containers" on page 278
- "Group Containers" on page 279
- "Groups" on page 280
- "People Containers" on page 283
- "Users" on page 284
- "Roles" on page 287

Organizations

An *Organization* represents the top-level of a hierarchical structure used by an enterprise to manage its departments and resources. Upon installation, Federated Access Manager dynamically creates a top-level organization (defined during installation) to manage the Federated Access Manager enterprise configurations. Additional organizations can be created after installation to manage separate enterprises. All created organizations fall beneath the top-level organization.

▼ To Create an Organization

1. **Click the Directory Management tab.**

2. **In the Organizations list, click New.**

3. **Enter the values for the fields. Only Name is required. The fields are:**

Name	Enter a value for the name of the Organization.
Domain Name	Enter the full Domain Name System (DNS) name for the organization, if it has one.
Organization Status	Choose a status of `active` or `inactive` . The default is `active`. This can be changed at any time during the life of the organization by selecting the Properties icon. Choosing `inactive` disables user access when logging in to the organization.

Organization Aliases	This field defines alias names for the organization, allowing you to use the aliases for authentication with a URL login. For example, if you have an organization named `exampleorg`, and define `123` and `abc` as aliases, you can log into the organization using any of the following URLs:
	
	`http://machine.example.com/amserver/UI/Login?org=exampleorg`
	
	`http://machine.example.com/amserver/UI/Login?org=abc`
	
	`http://machine.example.com/amserver/UI/Login?org=123`
	
	Organization alias names must be unique throughout the organization. You can use the Unique Attribute List to enforce uniqueness.

DNS Alias Names	Allows you to add alias names for the DNS name for the organization. This attribute only accepts “real” domain aliases (random strings are not allowed). For example, if you have a DNS named `example.com`, and define `example1.com` and `example2.com` as aliases for an organization named `exampleorg`, you can log into the organization using any of the following URLs:

	`http://machine.example.com/amserver/UI/`
	`Login?org=exampleorg`
	`http://machine.example1.com/amserver/`
	`UI/Login?org=exampleorg`
	`http://machine.example2.com/amserver/`
	`UI/Login?org=exampleorg`

Unique Attribute List	Allows you to add a list of unique attribute names for users in the organization. For example, if you add a unique attribute name specifying an email address, you would not be able to create two users with the same email address. This field also accepts a comma-separated list. Any one of the attribute names in the list defines uniqueness. For example, if the field contains the following list of attribute names: `PreferredDomain, AssociatedDomain` and `PreferredDomain` is defined as `http://www.example.com` for a particular user, then the entire comma-separated list is defined as unique for that URL. Adding the naming attribute 'ou' to the Unique Attribute List will not enforce uniqueness for the default groups, people containers. (ou=Groups,ou=People). Uniqueness is enforced for all sub organizations. **Note –** Unique attributes can not be set in Realm mode. They also cannot be set in the 7.0-8.0–based consoles for Legacy mode. In order to create the unique attribute list, you must login into the 6.3–based console. See “Legacy Mode 6.3 Console” on page 21 for more information.

4 Click OK.

The new organization displays in the Organization list. To edit any of the properties that you defined during creation of the organization, click the name of the organization you wish to edit, change the properties and click Save.

▼ To Delete an Organization

1. **Select the checkbox next to the name of the organization to be deleted.**

2. **Click Delete.**

Note – There is no warning message when performing a delete. All entries within the organization will be deleted and you can not perform an undo.

To Add an Organization to a Policy

Federated Access Manager objects are added to a policy through the policy's subject definition. When a policy is created or modified, organizations, roles, groups, and users can be defined as the subject. Once the subject is defined, the policy will be applied to the object. For more information, see "Managing Policies" on page 121.

Containers

The *container* entry is used when, due to object class and attribute differences, it is not possible to use an organization entry. It is important to remember that the Federated Access Manager container entry and the Federated Access Manager organization entry are not necessarily equivalent to the LDAP object classes `organizationalUnit` and `organization`. They are abstract identity entries. Ideally, the organization entry will be used instead of the container entry.

Note – The display of containers is optional. To view containers you must select Show Containers in the Administration service under Configuration>Console Properties.

▼ To Create a Container

1. **Select the location link of the organization or container where the new container will be created.**

2. **Click the Containers tab.**

3. **Click New in the Containers list.**

4. **Enter the name of the container to be created.**

5. **Click OK.**

▼ To Delete a Container

1. **Click the Containers tab.**

2. **Select the checkbox next to the name of the container to be deleted.**

3. **Click Delete.**

Note – Deleting a container will delete all objects that exist in that Container. This includes all objects and sub containers.

Group Containers

A *group container* is used to manage groups. It can contain only groups and other group containers. The group container Groups is dynamically assigned as the parent entry for all managed groups. Additional group containers can be added, if desired.

Note – The display of group containers is optional. To view group containers you must select Enable Group Containers in the Administration service under Configuration>Console Properties.

▼ To Create a Group Container

1. **Select the location link of the organization or the group container which will contain the new group container.**

2. **Select the Group Containers tab.**

3. **Click New in the Group Containers list.**

4. **Enter a value in the Name field and click OK. The new group container displays in the Group Containers list.**

▼ To Delete a Group Container

1. **Navigate to the organization which contains the group container to be deleted.**

2. **Choose the Group Containers tab.**

3. **Select the checkbox next to the group container to be deleted.**

4 Click Delete.

Groups

A *group* represents a collection of users with a common function, feature or interest. Typically, this grouping has no privileges associated with it. Groups can exist at two levels; within an organization and within other managed groups. Groups that exist within other groups are called *sub-groups*. Sub groups are child nodes that "physically" exist within a parent group.

Federated Access Manager also supports *nested groups,* which are "representations" of existing groups contained in a single group. As opposed to sub groups, nested groups can exist anywhere in the DIT. They allow you to quickly set up access permissions for a large number of users.

There are two types of groups you can create; static groups and dynamic groups. Users can only be manually added to static groups, while dynamic groups control the addition of users through a filter. Nested or sub groups can be added to both types.

Static Group

A static group is created based on the Managed Group Type you specify. Group members are added to a group entry using the `groupOfNames` or `groupOfUniqueNames` object class.

Note – By default, the managed group type is dynamic. You can change this default in the Administration service configuration.

Dynamic Group

A dynamic group is created through the use of an LDAP filter. All entries are funneled through the filter and dynamically assigned to the group. The filter would look for any attribute in an entry and return those that contain the attribute. For example, if you were to create a group based on a building number, you can use the filter to return a list all users containing the building number attribute.

Note – Federated Access Manager should be configured with Directory Server to use the referential integrity plug-in. When the referential integrity plug-in is enabled, it performs integrity updates on specified attributes immediately after a delete or rename operation. This ensures that relationships between related entries are maintained throughout the database. Database indexes enhance the search performance in Directory Server. For more information on enabling the plug-in, see the Sun Java Federated Access Manager 6 Migration Guide.

▼ To Create a Static Group

1. **Navigate to the organization, group, or group container where the new group will be created.**
2. **From the Groups list, click New Static.**
3. **Enter a name for the group in the Name field. Click Next.**
4. **Select the Users Can Subscribe to this Group attribute to allow users to subscribe to the group themselves.**
5. **Click OK.**

   Once the group is created, you can edit the Users Can Subscribe to this Group attribute by selecting the name of the group and clicking the General tab.

▼ To Add or Remove Members to a Static Group

1. **From the Groups list, select the group to which you will add members.**
2. **Choose an action to perform in the Select Action menu. The actions you can perform are as follows:**

New User	This action creates a new user and adds the user to the group when the user information is saved.
Add User	This action adds an existing user to the group. When you select this action, you create a search criteria which will specify users you wish to add. The fields used to construct the criteria use either an `ANY` or `ALL` operator. `ALL` returns users for all specified fields. `ANY` returns users for any one of the specified fields. If a field is left blank, it will match all possible entries for that particular attribute. Once you have constructed the search criteria, click Next. From the returned list of users, select the users you wish to add and click Finish.
Add Group	This action adds a nested group to the current group. When you select this action, you create a search criteria, including search scope, the name of the group (the "*" wildcard is accepted), and you can specify whether users can subscribe to the group themselves. Once you have entered the information, click Next. From the returned list of groups, select the group you wish to add and click Finish.
Remove Members	This action will remove members (which includes users and groups) from the group, but will not delete them. Select the member(s) you wish to remove and choose Remove Members from the Select Actions menu.

Delete Members	This action will permanently delete the member you select. Select the member(s) you wish to delete and choose Delete Members.

To Create a Dynamic Group

1. **Navigate to the organization or group where the new group will be created.**

2. **Click the Groups tab.**

3. **Click New Dynamic.**

4. **Enter a name for the group in the Name field.**

5. **Construct the LDAP search filter.**

   By default, Federated Access Manager displays the Basic search filter interface. The Basic fields used to construct the filter use either an `ANY` or `ALL` operator. `ALL` returns users for all specified fields. `ANY` returns users for any one of the specified fields. If a field is left blank it will match all possible entries for that particular attribute.

6. **When you click OK all users matching the search criteria are automatically added to the group.**

To Add or Remove Members to a Dynamic Group

1. **Form the Groups list, click the name of the group to which you will add members.**

2. **Choose an action to perform in the Select Action menu. The actions you can perform are as follows:**

Add Group	This action adds a nested group to the current group. When you select this action, you create a search criteria, including search scope, the name of the group (the "*" wildcard is accepted), and you can specify whether users can subscribe to the group themselves. Once you have entered the information, click Next. From the returned list of groups, select the group you wish to add and click Finish.
Remove Members	This action will remove members (which includes groups) from the group, but will not delete them. Select the member(s) you wish to remove and choose Remove Members
Delete Members	This action will permanently delete the member you select. Select the member(s) you wish to delete and choose Delete Members.

To Add a Group to a Policy

Federated Access Manager objects are added to a policy through the policy's subject definition. When a policy is created or modified, organizations, roles, groups, and users can be defined as the subject in the policy's Subject page. Once the subject is defined, the policy will be applied to the object. For more information, see "Managing Policies" on page 121.

People Containers

A *people container* is the default LDAP organizational unit to which all users are assigned when they are created within an organization. People containers can be found at the organization level and at the people container level as a sub People Container. They can contain only other people containers and users. Additional people containers can be added into the organization, if desired.

Note – The display of people containers is optional. To view People Containers you must select Enable People Containers in the Administration Service.

▼ Create a People Container

1. **Navigate to the organization or people container where the new people container will be created.**
2. **Click New from the People Container list.**
3. **Enter the name of the people container to be created.**
4. **Click OK.**

▼ To Delete a People Container

1. **Navigate to the organization or people container which contains the people container to be deleted.**
2. **Select the checkbox next to the name of the people container to be deleted.**
3. **Click Delete.**

Note – Deleting a people container will delete all objects that exist in that people container. This includes all users and sub people containers.

Users

A *user* represents an individual's identity. Through the Federated Access Manager Identity Management module, users can be created and deleted in organizations, containers and groups and can be added or removed from roles and/or groups. You can also assign services to the user.

Note – If a user in a sub organization is created with the same user ID as `amadmin`, the login will fail for `amadmin`. If this problem occurs, the administrator should change the user's ID through the Directory Server console. This enables the administrator to login to the default organization. Additionally, the DN to Start User Search in the authentication service can be set to the people container DN to ensure that a unique match is returned during the login process.

▼ To Create a User

1. **Navigate to the organization, container or people container where the user is to be created.**

2. **Click the user tab.**

3. **Click New from the user list.**

4. **Enter data for the following values:**

User ID	This field takes the name of the user with which he or she will log into Federated Access Manager. This property may be a non-DN value.
First Name	This field takes the first name of the user. The First Name value and the Last Name value identify the user in the Currently Logged In field. This is not a required value.
Last Name	This field takes the last name of the user. The First Name value and the Last Name value identify the user.
Full Name	This field takes the full name of the user.
Password	This field takes the password for the name specified in the User Id field.
Password (Confirm)	Confirm the password.
User Status	This option indicates whether the user is allowed to authenticate through Federated Access Manager. Only active users can authenticate. The default value is `Active.`

5. **Click OK.**

▼ To Edit the User Profile

When a user who has not been assigned an administrative role authenticates to Federated Access Manager, the default view is their own User Profile. Additionally, administrators with the proper privileges can edit user profiles. In this view the user can modify the values of the attributes particular to their personal profile. The attributes displayed in the User Profile view can be extended. For more information on adding customized attributes for objects and identities, see the Federated Access Manager Developer's Guide.

1 **Select the user who's profile is to be edited. By default, the General view is displayed.**

2 **Edit the following fields:**

First Name	This field takes the first name of the user.
Last Name	This field takes the last name of the user.
Full Name	This field takes the full name of the user.
Password	Click the Edit link to add and confirm the user password.
Email Address	This field takes the email address of the user.
Employee Number	This field takes the employee number of the user.
Telephone Number	This field takes the telephone number of the user.
Home Address	This field can take the home address of the user.
User Status	This option indicates whether the user is allowed to authenticate through Federated Access Manager. Only active users can authenticate through Federated Access Manager. The default value is Active. Either of the following can be selected from the pull-down menu: . ▪ Active — The user can authenticate through Federated Access Manager. ▪ Inactive — The user cannot authenticate through Federated Access Manager, but the user profile remains stored in the directory.

Note – Changing the user status to Inactive only affects authentication through Federated Access Manager. The Directory Server uses the *nsAccountLock* attribute to determine user account status. User accounts inactivated for Federated Access Manager authentication can still perform tasks that do not require Federated Access Manager. To inactivate a user account in the directory, and not just for Federated Access Manager authentication, set the value of *nsAccountLock* to false. If delegated administrators at your site will be inactivating users on a regular basis, consider adding the *nsAccountLock* attribute to the Federated Access Manager User Profile page. See the *Sun Java System Access Manager 7.1 Developer's Guide* for details.

Account Expiration Date	If this attribute is present, the authentication service will disallow login if the current date and time has passed the specified Account Expiration Date. The format for this attribute is *mm/dd/yyyy hh:mm*.
User Authentication Configuration	This attribute sets the authentication chain for the user.
User Alias List	The field defines a list of aliases that may be applied to the user. In order to use any aliases configured in this attribute, the LDAP service has to be modified by adding the `iplanet-am-user-alias-list` attribute to the User Entry Search Attributes field in the LDAP service.
Preferred Locale	This field specifies the locale for the user.
Success URL	This attribute specifies the URL that the user will be redirected to upon successful authentication.
Failure URL.	This attribute specifies the URL that the user will be redirected to upon unsuccessful authentication.
Password Reset Options	This is used to select the questions on the forgotten password page, which is used to recover a forgotten password.
User Discovery Resource Offering	Sets the User Discovery service's resource offering for the user.

MSIDSN Number	Defines the user's MSISDN number if using MSISDN authentication.

▼ To Add a User to Roles and Groups

1. **Click the Users tab.**
2. **Click the name of the user you wish to modify.**
3. **Select either the Roles or Groups tab.**
4. **Select the role or group to which you wish to add the user and click Add.**
5. **Click Save.**

Note – To remove a user from Roles or groups, Select roles or groups and click Remove and then Save.

To Add a User to a Policy

Federated Access Manager objects are added to a policy through the policy's subject definition. When a policy is created or modified, organizations, roles, groups, and users can be defined as the subject in the policy's Subject page. Once the subject is defined, the policy will be applied to the object. For more information, see "Managing Policies" on page 121.

Roles

Roles are a Directory Server entry mechanism similar to the concept of a *group*. A group has members; a role has members. A role's members are LDAP entries that possess the role. The criteria of the role itself is defined as an LDAP entry with attributes, identified by the Distinguished Name (DN) attribute of the entry. Directory Server has a number of different types of roles but Federated Access Manager can manage only one of them: the managed role.

Note – The other Directory Server role types can still be used in a directory deployment; they just can not be managed by the Federated Access Manager console. Other Directory Server types can be used in a policy's subject definition. For more information on policy subjects, see "Creating Policies" on page 113.

Users can possess one or more roles. For example, a contractor role which has attributes from the Session Service and the Password Reset Service might be created. When new contractor employees join the company, the administrator can assign them this role rather than setting separate attributes in the contractor entry. If the contractor is working in the Engineering department and requires services and access rights applicable to an engineering employee, the administrator could assign the contractor to the engineering role as well as the contractor role.

Federated Access Manager uses roles to apply access control instructions. When first installed, Federated Access Manager configures access control instructions (ACIs) that define administrator permissions. These ACIs are then designated in roles (such as Organization Admin Role and Organization Help Desk Admin Role) which, when assigned to a user, define the user's access permissions.

Users can view their assigned roles only if the Show Roles on User Profile Page attribute is enabled in the Administration Service.

Note – Federated Access Manager should be configured with Directory Server to use the referential integrity plug-in. When the referential integrity plug-in is enabled, it performs integrity updates on specified attributes immediately after a delete or rename operation. This ensures that relationships between related entries are maintained throughout the database. Database indexes enhance the search performance in Directory Server.

There are two types of roles:

- Static — Static roles are created without adding users at the point of the role's creation. Once the role is created, you can then add specific users to it. This gives you more control when adding users to a given role.
- Dynamic – Dynamic roles are created through the use of an LDAP filter. All users are funneled through the filter and assigned to the role at the time of the role's creation. The filter looks for any attribute value pair (for example, `ca=user*`) in an entry and automatically assign the users that contain the attribute to the role.

▼ To Create a Static Role

1. **Go to the organization where the Role will be created.**

2. **Click the Roles tab.**

   A set of default roles are created when an organization is configured, and are displayed in the Roles list. The default roles are:

   Container Help Desk Admin. The Container Help Desk Admin role has read access to all entries in an organizational unit and write access to the `userPassword` attribute in user entries only in this container unit.

Organization Help Desk Admin. The Organization Help Desk Administrator has read access to all entries in an organization and write access to the `userPassword` attribute.

Note – When a sub organization is created, remember that the administration roles are created in the sub organization, not in the parent organization.

Container Admin. The Container Admin role has read and write access to all entries in an LDAP organizational unit. In Federated Access Manager, the LDAP organizational unit is often referred to as a container.

Organization Policy Admin. The Organization Policy Administrator has read and write access to all policies, and can create, assign, modify, and delete all policies within that organization.

People Admin. By default, any user entry in an newly created organization is a member of that organization. The People Administrator has read and write access to all user entries in the organization. Keep in mind that this role DOES NOT have read and write access to the attributes that contain role and group DNs therefore, they cannot modify the attributes of, or remove a user from, a role or a group.

Note – Other containers can be configured with Federated Access Manager to hold user entries, group entries or even other containers. To apply an Administrator role to a container created after the organization has already been configured, the Container Admin Role or Container Help Desk Admin defaults would be used.

Group Admin. The Group Administrator created when a group is created has read and write access to all members of a specific group, and can create new users, assign users to the groups they manage, and delete the users the that they have created.

When a group is created, the Group Administrator role is automatically generated with the necessary privileges to manage the group. The role is not automatically assigned to a group member. It must be assigned by the group's creator, or anyone that has access to the Group Administrator Role.

Top-level Admin. The Top-level Administrator has read and write access to all entries in the top-level organization. In other words, this Top-level Admin role has privileges for every configuration principal within the Federated Access Manager application.

Organization Admin. The Organization Administrator has read and write access to all entries in an organization. When an organization is created, the Organization Admin role is automatically generated with the necessary privileges to manage the organization.

3 **Click the New Static button.**

4 **Enter a name for the role.**

5 **Enter a description of the role.**

6 **Choose the role type from the Type menu.**

The role can be either an Administrative role or a Service role. The role type is used by the console to determine and here to start the user in the Federated Access Manager console. An administrative role notifies the console that the possessor of the role has administrative privileges; the service role notifies the console that the possessor is an end user.

7 **Choose a default set of permissions to apply to the role from the Access Permission menu. The permissions provide access to entries within the organization. The default permissions shown are in no particular order. The permissions are:**

No permissions	No permissions are to be set on the role.
Organization Admin	The Organization Administrator has read and write access to all entries in the configured organization.
Organization Help Desk Admin	The Organization Help Desk Administrator has read access to all entries in the configured organization and write access to the `userPassword` attribute.
Organization Policy Admin	The Organization Policy Administrator has read and write access to all policies in the organization. The Organization Policy Administrator can not create a referral policy to a peer organization. Generally, the No Permissions ACI is assigned to Service roles, while Administrative roles are assigned any of the default ACIs.

▼ To Add Users to a Static Role

1 **Click the name of the role to which you wish to add users.**

2 **In the Members list, select Add User from the Select Action menu.**

3 **Enter the information for the search criteria. You can choose to search for users based on one or more the displayed fields The fields are:**

Match	Allows you to select the fields you wish to include for the filter. `ALL` returns users for all specified fields. `ANY` returns users for any one of the specified fields.
First Name	Search for users by their first name.
User ID	Search for a user by User ID.
Last Name	Search for users by their last name.

Full Name	Search for users by their full name.
User Status	Search for users by their status (active or inactive)

4 **Click Next to begin the search. The results of the search are displayed.**

5 **Choose the users from the names returned by selecting the checkbox next to the user name.**

6 **Click Finish.**

The Users are now assigned to the role.

▼ To Create a Dynamic Role

1 **Go to the organization where the Role will be created.**

2 **Click the Roles tab.**

A set of default roles are created when an organization is configured, and are displayed in the Roles list. The default roles are:

Container Help Desk Admin. The Container Help Desk Admin role has read access to all entries in an organizational unit and write access to the `userPassword` attribute in user entries only in this container unit.

Organization Help Desk Admin. The Organization Help Desk Administrator has read access to all entries in an organization and write access to the `userPassword` attribute.

Note – When a sub organization is created, remember that the administration roles are created in the sub organization, not in the parent organization.

Container Admin. The Container Admin role has read and write access to all entries in an LDAP organizational unit. In Federated Access Manager, the LDAP organizational unit is often referred to as a container.

Organization Policy Admin. The Organization Policy Administrator has read and write access to all policies, and can create, assign, modify, and delete all policies within that organization.

People Admin. By default, any user entry in an newly created organization is a member of that organization. The People Administrator has read and write access to all user entries in the organization. Keep in mind that this role DOES NOT have read and write access to the attributes that contain role and group DNs therefore, they cannot modify the attributes of, or remove a user from, a role or a group.

Note – Other containers can be configured with Federated Access Manager to hold user entries, group entries or even other containers. To apply an Administrator role to a container created after the organization has already been configured, the Container Admin Role or Container Help Desk Admin defaults would be used.

Group Admin. The Group Administrator created when a group is created has read and write access to all members of a specific group, and can create new users, assign users to the groups they manage, and delete the users the that they have created.

When a group is created, the Group Administrator role is automatically generated with the necessary privileges to manage the group. The role is not automatically assigned to a group member. It must be assigned by the group's creator, or anyone that has access to the Group Administrator Role.

Top-level Admin. The Top-level Administrator has read and write access to all entries in the top-level organization. In other words, this Top-level Admin role has privileges for every configuration principal within the Federated Access Manager application.

Organization Admin. The Organization Administrator has read and write access to all entries in an organization. When an organization is created, the Organization Admin role is automatically generated with the necessary privileges to manage the organization.

3 **Click the New Dynamic button.**

4 **Enter a name for the role.**

5 **Enter a description for the role.**

6 **Choose the role type from the Type menu.**

The role can be either an Administrative role or a Service role. The role type is used by the console to determine and where to start the user in the Federated Access Manager console. An administrative role notifies the console that the possessor of the role has administrative privileges; the service role notifies the console that the possessor is an end user.

7 **Choose a default set of permissions to apply to the role from the Access Permission menu. The permissions provide access to entries within the organization. The default permissions shown are in no particular order. The permissions are:**

No permissions	No permissions are to be set on the role.
Organization Admin	The Organization Administrator has read and write access to all entries in the configured organization.
Organization Help Desk Admin	The Organization Help Desk Administrator has read access to all entries in the configured organization and write access to the `userPassword` attribute.

Organization Policy Admin	The Organization Policy Administrator has read and write access to all policies in the organization. The Organization Policy Administrator can not create a referral policy to a peer organization.

Generally, the No Permissions ACI is assigned to Service roles, while Administrative roles are assigned any of the default ACIs.

8 **Enter the information for the search criteria. The fields are:**

Match	Allows you to include an operator for any the fields you wish to include for the filter. `ALL` returns users for all specified fields. `ANY` returns users for any one of the specified fields.
First Name	Search for users by their first name.
User ID	Search for a user by User ID.
Last Name	Search for users by their last name.
Full Name	Search for users by their full name.
User Status	Search for users by their status (active or inactive)

9 **Click OK to initiate the search based on the filter criteria. The users defined by the filter criteria are automatically assigned to the role.**

▼ To Remove Users from a Role

1 **Navigate to the Organization that contains the role to modify.**

Choose Organizations from the View menu in the Identity Management module and select the Roles tab.

2 **Select the role to modify.**

3 **Choose Users from the View menu.**

4 **Select the checkbox next to each user to be removed.**

5 **Click Remove user from the Select Action menu.**

The users are now removed from the role.

To Add a Role to a Policy

Federated Access Manager objects are added to a policy through the policy's subject definition. When a policy is created or modified, organizations, roles, groups, and users can be defined as the subject in the policy's Subject page. Once the subject is defined, the policy will be applied to the object. For more information, see "Managing Policies" on page 121.

◆ ◆ ◆ CHAPTER 12

Current Sessions

This chapter describes the session management features of Federated Access Manager. The Session Management 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 terminate a session. System administrators should ignore the Load Balancer servers listed in the Platform Server list.

The Current Sessions Interface

The Current Sessions module interface allows an administrator, with the appropriate permissions, to view the session information for any user who is currently logged in to Federated Access Manager.

Session Management

The Session Management frame displays the name of the Federated Access Manager that is currently being managed.

Session Information

The Session Information window displays all of the users who are currently logged into Federated Access Manager, and displays the session time for each user. The display fields are:

User ID. Displays the user ID of the user who is currently logged in.

Time Left. Displays the amount of time (in minutes) remaining that the user has for that session before having to re-authenticate.

Max Session Time. Displays the maximum time (in minutes) that the user can be logged in before the session expires and must re-authenticate to regain access.

Idle Time. Displays the time (in minutes) that the user has been idle.

Max Idle Time. Displays the maximum time (in minutes) that a user can remain idle before having to re-authenticate.

The time limits are defined by the administrator in the Session Management Service.

You can display a specific user session, or a specific range of user sessions, by entering a string in the User ID field and clicking Filter. Wildcards are permitted.

Clicking the Refresh button will update the user session display.

Terminating a Session

Administrators with appropriate permissions can terminate a user session at any time.

▼ To Terminate a Session

1. **Select the user session that you wish to terminate.**

2. **Click Terminate.**

◆ ◆ ◆ CHAPTER 13

Password Reset Service

Federated Access Manager provides a Password Reset service to allow users to reset their password for access to a given service or application protected by Federated Access Manager. The Password Reset service attributes, defined by the top-level administrator, control user validation credentials (in the form of secret questions), control the mechanism for new or existing password notification, and sets possible lockout intervals for incorrect user validation.

This chapter contains the following sections:

Registering the Password Reset Service

The Password Reset service does not need to be registered for the realm in which the user resides. If the Password Reset service does not exist in the organization in which the user resides, it will inherit the values defined for the service in Service Configuration.

▼ To Register Password Reset for Users in a Different Realm

1 **Navigate to the realm to which you will register the password for the user.**

2 **Click the realm name and click the Services tab.**

If it has not been added to the realm, click the Add button.

3 **Select the r Password Reset and click Next**

The Password Reset service attributes will be displayed. For attribute definitions, see the online help.

4 **Click Finish.**

Configuring the Password Reset Service

Once the Password Reset service has been registered, the service must be configured by a user with administrator privileges.

▼ To Configure the Service

1 **Select the realm for which the Password Reset service is registered.**

2 **Click the Services tab.**

3 **Click Password Reset from the services list.**

4 **The Password Reset attributes appear, allowing you to define requirements for the Password Reset service. Make sure that the Password Reset service is enabled (it is by default). At a minimum, the following attributes must be defined:**

- User Validation
  - Secret Question
  - Bind DN
  - Bind Password

    The Bind DN attribute must contain a user with privileges for resetting the password (for example, Help Desk Administrator). Due a limitation in Directory Server, Password Reset does not work when the bind DN is `cn=Directory Manager`.

    The remaining attributes are optional. See the online help for a description of the service attributes.

Note – Federated Access Manager automatically installs the Password Reset web application for random password generation. However, you can write your own plug-in classes for password generation and password notification. See the following Readme.html files in the following locations for samples for these plug-in classes.

PasswordGenerator:

```
FederatedAccessManager-base/SUNWam/samples/console/PasswordGenerator
```

NotifyPassword:

```
FederatedAccessManager-base/SUNWam/samples/console/NotifyPassword
```

---

5 **Select the Personal Question Enabled attribute if the user is to define his/her unique personal questions. Once the attributes are defined, click Save.**

▼ To Localize the Secret Question

If you are running a localized version of the Federated Access Manager, and wish to display the secret question in a character set specific to you locale, perform the following:

1 **Add the secret question key to the Current Values list under the Secret Question attribute in the Password Reset service. For example,** `favorite-color`**.**

2 **Add the key to the** `amPasswordReset.properties` **file with the question that you want to displays the value of this key. For example:**

```
favorite-color=What is your favorite color?
```

3 **Add the same key with the localized question to** `AMPasswordReset_`*locale*`.properties` **located in** `/opt/SUNWam/locale`**. When the user attempts to changes his or her password, the localized question is displayed.**

Password Reset Lockout

The Password Reset service contains a lockout feature that will restrict users to a certain number of attempts to correctly answer their secret questions. The lockout feature is configured through the Password Reset service attributes. See the online help for a description of the service attributes. Password Reset supports two types of lockout, memory lockout and physical lockout.

Memory Lockout

This is a temporary lockout and is in effect only when the value in the Password Reset Failure Lockout Duration attribute is greater than zero and the Enable Password Reset Failure Lockout attribute is enabled. This lockout will prevent users from resetting their password through the Password Reset web application. The lockout lasts for the duration specified in Password Reset Failure Lockout Duration, or until the server is restarted. See the online help for a description of the service attributes.

Physical Lockout

This is a more permanent lockout. If the value set in the Password Reset Failure Lockout Count attribute is set to 0 and the Enable Password Reset Failure Lockout attribute is enabled, the users' account status is changed to inactive when he or she incorrectly answers the secret questions. See the online help for a description of the service attributes.

Password Reset for End Users

The following sections describe the user experience for the Password Reset service.

Customizing Password Reset

Once the Password Reset service has been enabled and the attributes defined by the administrator, users are able to log into the Federated Access Manager console in order to customize their secret questions.

▼ To Customize Password Reset

1. **The user logs into the Federated Access Manager console, providing Username and Password and is successfully authenticated.**

2. **In the User Profile page, the user selects Password Reset Options. This displays the Available Questions Answer Screen.**

3. **The user is presented with the available questions that the administrator defined for the service, such as:**

   - What is your pet's name?
     - What is your favorite TV show?
     - What is your mother's maiden name?
     - What is your favorite restaurant?

4 **The user selects the secret questions, up to the maximum number of questions that the administrator defined for the realm (the maximum amount is defined the Password Reset Service). The user then provides answers to the selected questions. These questions and answers will be the basis for resetting the user's password (see the following section). If the administrator has selected the Personal Question Enabled attribute, text fields are provided, allowing the user to enter a unique secret question and provide an answer.**

5 **The user clicks Save.**

Resetting Forgotten Passwords

In the case where users forget their password, Federated Access Manager uses the Password Reset web application to randomly generate new passwords and notify the user of the new password. A typical forgotten password scenario follows:

▼ To Reset Forgotten Passwords

1 **The user logs into the Password Reset web application from a URL given to them by the administrator. For example:**

`http://`hostname:port `/ampassword` (for the default realm

or

`http://`hostname: port/*deploy_uri* `/UI/PWResetUserValidation?realm=realmname`, where realmname is the name of the realm.

Note – If the Password Reset service is not enabled for a parent realm but is enabled for a sub-realm, users must use the following syntax to access the service:

`http://hostname: port/`*deploy_uri*`/UI/PWResetUserValidation?realm=realmname`

---

2 **The user enters the user id.**

3 **The user is presented with the personal questions that were defined in the Password Reset service and select by the user during customization. If the user has not previously logged into the User Profile page and customized the personal questions, the password will not be generated.**

Once the user answers the questions correctly, the new password is generated and emailed to the user. Attempt notification is sent to the user whether the questions are answered correctly or not. Users must have their email address entered in the User Profile page in order for the new password and attempt notification to be received.

Password Policies

A password policy is a set of rules to govern how passwords are used in a given directory. Password policies are defined in the Directory Server, typically through the Directory Server console. A secure password policy minimizes the risks associated with easily-guessed passwords by enforcing the following:

- Users must change their passwords according to a schedule.
- Users must provide non-trivial passwords.
- Accounts may be locked after a number of binds with the wrong password.

Directory Server provides several ways to set password policy at any node in a tree and there are several ways to set the policy. For details refer to

Directory Server Password Policy in the Directory Server Enterprise Edition 6.0 Administration Guide.

Note – In Directory Server, the password policy contains an attribute, `passwordExp`, that defines whether user passwords will expire after a given number of seconds. If the administrator sets the `passwordExp` attribute to `on`, this sets the expiration for the end-user's password as well as the Federated Access Manager's administration accounts, such as `amldap`, `dsame`, and `puser`. When the Federated Access Manager administrator's account password expires, and an end-user is logged in, the user will receive the password change screen. However, Federated Access Manager does not specify the user to which the password change screen pertains. In this case, the screen is intended for the administrator and the end-user will be unable to change the password.

To resolve this, the administrator must log in to the Directory Server and change the `amldap`, `dsame`, and `puser` passwords, or change the `passwordExpirationTime` attribute for some time in the future.

CHAPTER 14

Logging Service

Sun Java™ System Federated Access Manager provides a Logging Service to record information such as user activity, traffic patterns, and authorization violations. In addition, the debug files allow administrators to troubleshoot their installation.

Log Files

The log files record a number of events for each of the services it monitors. These files should be checked by the administrator on a regular basis. The default directory for the log files is `/var/opt/SUNWam/logs` for SPARC systems, `/var/opt/sun/identity` for Linux systems, `/var/opt/sun/identity` for HP-UX, and *jes-install-dir*`\identity` for Windows. The log file directory can be configured in the Logging Service by using the Federated Access Manager console.

See "Logging Overview" in *Sun Java System Access Manager 7.1 Technical Overview* in the Sun Federated Access Manager Technical Overview for a detailed list of the default log file types, the information that is recorded, and log file formats.

For attribute definitions for the Logging Service, see the online help by clicking the Help button in the Federated Access Manager Console.

Federated Access Manager Service Logs

There are two different types of service log files: access and error. Access log files may contain records of action attempts and successful results. Error log files record errors that occur within the Federated Access Manager services. Flat log files are appended with the `.error` or `.access` extension. Database column names end with `_ERROR` or `_ACCESS` for Oracle databases, or `_error` or `_access` for MySQL databases. For example, a flat file logging console events is named `amConsole.access`, while a database column logging the same events is named `AMCONSOLE_ACCESS.` The following sections describe the log files recorded by the Logging Service.

Session Logs

The Logging Service records the following events for the Session Service:

- Login
- Logout
- Session Idle TimeOut
- Session Max TimeOut
- Failed To Login
- Session Reactivation
- Session Destroy

The session logs are prefixed with `amSSO.`

Console Logs

The Federated Access Manager console logs record the creation, deletion and modification of identity-related objects, policies and services including, among others, organizations, organizational units, users, roles, policies and groups. It also records modifications of user attributes including passwords and the addition or removal of users to or from roles and groups. Additionally, the console logs write delegation and data store activities. The console logs are prefixed with `amConsole.`

Authentication Logs

Authentication component logs user logins and logouts. The authentication logs are prefixed with `amAuthentication.`

Federation Logs

The Federation component logs federation-related events including, but not limited to, the creation of an Authentication Domain and the creation of a Hosted Provider. The federation logs are prefixed with `amFederation.`

Policy Logs

The Policy component records policy-related events including, but not limited to, policy administration (policy creation, deletion and modification) and policy evaluation. The policy logs are prefixed with `amPolicy`.

Agent Logs

The policy agent logs are responsible for logging exceptions regarding log resources that were either allowed or denied to a user. The agent logs are prefixed with `amAgent.` `amAgent` logs reside on the agent server only. Agent events are logged on the Federated Access Manager server in the Authentication Logs. For more information on this function, see the documentation for the policy agent in question.

SAML Logs

The SAML component records SAML-related events including, but not limited to, assertion and artifact creation or removal, response and request details, and SOAP errors. The session logs are prefixed with `amSAML`.

amadmin Logs

The command line logs record event errors that occur during operations using the command line tools. These include, but are not limited to, loading a service schema, creating policy and deleting users. The command line logs are prefixed with `amAdmin.` the `amadmin.access` and `amadmin.error` log files reside in a subdirectory of the main logging directory. By default, the `amadmin` command line tool log files reside in `/var/opt/SUNWam/logs`.

Logging Features

The Logging Service has a number of special features which can be enabled for additional functionality. They include To Enable Secure Logging, Command Line Logging and Remote Logging.

Secure Logging

This optional feature adds additional security to the logging function. Secure Logging enables detection of unauthorized changes to, or tampering of, the security logs. No special coding is required to leverage this feature. Secure Logging is accomplished by using a pre-registered certificate configured by the system administrator. This Manifest Analysis and Certification (MAC) is generated and stored for every log record. A special "signature" log record is periodically inserted that represents the signature for the contents of the log written to that point. The combination of the two records ensures that the logs have not been tampered with. There are to methods to enable secure logging; through a Java Security Server (JSS) provider and through a Java Cryptography Extension (JCE) provider.

▼ To Enable Secure Logging through a JSS Provider

1. **Create a certificate with the name `Logger` and install it in the deployment container running Federated Access Manager.**

   For instructions for Application Server, see "Working with Certificates and SSL" in *Sun Java System Application Server Enterprise Edition 8.2 Administration Guide* in the *Sun Java System Application Server Enterprise Edition 8.2 Administration Guide*.

   For instructions for Web Server, see "Managing Certificates" in *Sun Java System Web Server 7.0 Administrator's Guide* in the *Sun Java System Web Server 7.0 Administration Guide*.

2. **Turn on Secure Logging in the Logging Service configuration using the Federated Access Manager console and save the change. The administrator can also modify the default values for the other attributes in the Logging Service.**

   If the logging directory is changed from the default (`/var/opt/SUNWam/logs`), make sure that the permissions are set to 0700. The logging service will create the directory, if it does not exist, but it will create the directory with permissions set to 0755.

   Additionally, if you specify a different directory from the default, you must change the following parameter to the new directory in the web container's `server.policy` file:

   ```
   permission java.io.FilePermission "/var/opt/SUNWam/logs/*","delete,write"
   ```

3. **Create a file in the *FederatedAccessManager-base*/`SUNWam/config` directory that contains the certificate database password and name it `.wtpass`.**

   Note – The file name and the path to it is configurable in the `AMConfig.properties` file. For more information see the "Certificate Database" in AMConfig.properties file reference chapter in the *Federated Access Manager Administration Reference*.

   Ensure that the deployment container user is the only administrator with read permissions to this file for security reasons.

   ---

4. **Restart the server.**

   The secure log directory should be cleared, as some misleading verification errors may be written to the `/var/opt/SUNWam/debug/amLog` file when the secure logging was started.

   To detect unauthorized changes or tampering of the security logs, look for error messages that are written by the verification process to `/var/opt/SUNWam/debug/amLog`. To manually check for tampering, run the `VerifyArchive` utility. See The VerifyArchive command line chapter in the *Federated Access Manager Administration Reference* for more information.

▼ To Enable Secure Logging Through a JCE Provider

1. **Create a certificate named Logger with Java `keytool` command and install it in JKS keystore. For example:**

   `JAVA-HOME/jre/lib/security/Logger.jks`

   For instructions for Application Server, see "Working with Certificates and SSL" in *Sun Java System Application Server Enterprise Edition 8.2 Administration Guide* in the *Sun Java System Application Server Enterprise Edition 8.2 Administration Guide*.

   For instructions for Web Server, see "Managing Certificates" in *Sun Java System Web Server 7.0 Administrator's Guide* in the *Sun Java System Web Server 7.0 Administration Guide*.

2. **Turn on Secure Logging in the Logging Service configuration using the Federated Access Manager console and save the change. The administrator can also modify the default values for the other attributes in the Logging Service.**

   If the logging directory is changed from the default (`/var/opt/SUNWam/logs`), make sure that the permissions are set to 0700. The logging service will create the directory, if it does not exist, but it will create the directory with permissions set to 0755.

   Additionally, if you specify a different directory from the default, you must change the following parameter to the new directory in the web container's `server.policy` file:

   ```
   permission java.io.FilePermission "/var/opt/SUNWam/logs/*","delete,write"
   ```

3. **Create a file in the *FederatedAccessManager-base*/`SUNWam/config` directory that contains the JKS keystore password and name it `.wtpass`.**

   Note – The file name and the path to it is configurable in the `AMConfig.properties` file. For more information see the "Certificate Database" in the AMConfig.properties file reference chapter in the *Federated Access Manager Administration Reference*.

   Ensure that the deployment container user is the only administrator with read permissions to this file for security reasons.

   ---

4. **Edit the following entries in the `amLogging.xml`, located in the *FederatedAccessManager-base*/`config/xml` directory:**

```
sun-am-logging-secure-log-helper

                  <AttributeSchema name="iplanet-am-logging-secure-log-helper"
                      type="single"
                      syntax="string"
                      i18nKey="">
                      <DefaultValues>
                          <Value>com.sun.identity.log.secure.impl.SecureLogHelperJCEImpl</Value>
                      </DefaultValues>
```

```
              </AttributeSchema>
```

```
sun-am-logging-secure-certificate-store
```

```
              <AttributeSchema name="iplanet-am-logging-secure-certificate-store"
                  type="single"
                  syntax="string"
                  i18nKey="">
                  <DefaultValues>
                      <Value>/dir-to-signing-cert-store/Logger.jks</Value>
                  </DefaultValues>
              </AttributeSchema>
```

5 **Delete the existing service schema,** `iPlanetAMLoggingService`**. For example:**

```
./amadmin -u amadmin -w netscape -r iPlanetAMLoggingService
```

6 **Load the edited amLoging.xml to Federated Access Manager using the amadmin command line tool. For example:**

```
./amadmin -u amadmin -w netscape -s /etc/opt/SUNWam/config/xml/amLogging.xml
```

7 **Restart the server.**

To detect unauthorized changes or tampering of the security logs, look for error messages that are written by the verification process to `/var/opt/SUNWam/debug/amLog`. To manually check for tampering, run the `VerifyArchive` utility. See The `VerifyArchive` command line chapter in the *Federated Access Manager Administration Reference* for more information.

Command Line Logging

The `amadmin` command line tool has the ability to create, modify and delete identity objects (organizations, users, and roles, for example) in Directory Server. This tool can also load, create, and register service templates. The Logging Service can record these actions by invoking the `-t` option. If the `com.iplanet.am.logstatus` property in `AMConfig.properties` is enabled (ACTIVE) then a log record will be created. (This property is enabled by default.) The command line logs are prefixed with `amAdmin.` See “The amadmin Command Line Tool” in the *Access Manager Administration Reference* for more information.

Logging Properties

There are properties in the `AMConfig.properties` file that affect logging output:

`com.iplanet.am.logstatus=ACTIVE`	This property will enable or disable logging. The default is ACTIVE.

`iplanet-am-logging.`*service*`.level=` *level*	*service* is the service's normal log file name. For example, to specify a logging level for `amSAML.access`, use the property `iplanet-am-logging.amSAML.access.level`.*level* is one of the `java.util.logging.Level` values and denotes the level of detail recorded in the log file. The levels are OFF, SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, and ALL. Most services do not record log levels with higher detail than INFO.

Remote Logging

Federated Access Manager supports remote logging. This allows a client application using a host where the Federated Access Manager server is installed to create log records on an instance of Federated Access Manager deployed on a remote machine. Remote logging can be initiated in any of the following scenarios:

1. When the logging URL in the Naming Service of one Federated Access Manager instance points to a remote instance and there is a trust relationship configured between the two, logs will be written to the remote Federated Access Manager instance.
2. When the Federated Access Manager SDK is installed against a remote Federated Access Manager instance and a client (or a simple Java class) running on the SDK server uses the logging APIs, the logs will be written to the remote Federated Access Manager machine.
3. When logging APIs are used by Federated Access Manager agents.

▼ To Enable Remote Logging with Web Containers

1 Log into the either the Application Server or Web Server's administration console and add the following JVM options:

- `java.util.logging.manager=com.sun.identity.log.LogManager`
- `java.util.logging.config.file=/`*FederatedAccessManager-base*`/SUNwam/lib/LogConfig.properties`

  For more information on the Application Server administration console, see *Sun Java System Application Server Enterprise Edition 8.2 Administration Guide*.

  For more information on the Web Server administration console, see *Sun Java System Web Server 7.0 Administrator's Guide*.
- If the Java™ 2 Platform, Standard Edition being used is 1.4 or later, this is accomplished by invoking the following at the command line:

```
java -cp /FederatedAccessManager-base
/SUNWam/lib/am_logging.jar:/FederatedAccessManager-base
/SUNWam/lib/xercesImpl.jar:/FederatedAccessManager-base
/SUNWam/lib/xmlParserAPIs.jar:/FederatedAccessManager-base
/SUNWam/lib/jaas.jar:/FederatedAccessManager-base
/SUNWam/lib/xmlParserAPIs.jar:/FederatedAccessManager-base
/SUNWam/lib/servlet.jar:/FederatedAccessManager-base
/SUNWam/locale:/FederatedAccessManager-base/SUNWam/lib/am_services.jar:/
FederatedAccessManager-base/SUNWam/lib/am_sdk.jar:/
FederatedAccessManager-base/SUNWam/lib/jss311.jar:/
FederatedAccessManager-base/SUNWam/lib:.
```

```
-Djava.util.logging.manager=com.sun.identity.log.LogManager
```

```
-Djava.util.logging.config.file=/FederatedAccessManager-base
/SUNwam/lib/LogConfig.properties
```

- If the Java 2 Platform, Standard Edition being used is earlier than 1.4, this is accomplished by invoking the following at the command line:

```
java -Xbootclasspath/a:/FederatedAccessManager-base
/SUNWam/lib/jdk_logging.jar -cp /FederatedAccessManager-base
/SUNWam/lib/am_logging.jar:/FederatedAccessManager-base
/SUNWam/lib/xercesImpl.jar:/FederatedAccessManager-base
/SUNWam/lib/xmlParserAPIs.jar:/FederatedAccessManager-base
/SUNWam/lib/jaas.jar:/FederatedAccessManager-base
/SUNWam/lib/xmlParserAPIs.jar:/FederatedAccessManager-base
/SUNWam/lib/servlet.jar:/FederatedAccessManager-base
/SUNWam/locale:/FederatedAccessManager-base/SUNWam/lib/am_services.jar:/
FederatedAccessManager-base/SUNWam/lib/am_sdk.jar:/
FederatedAccessManager-base/SUNWam/lib/jss311.jar:/
FederatedAccessManager-base/SUNWam/lib:.
```

```
-Djava.util.logging.manager=com.sun.identity.log.LogManager
```

```
-Djava.util.logging.config.file=/FederatedAccessManager-base
/SUNwam/lib/LogConfig.properties
```

2 **Ensure that the following parameters are configured in** `LogConfig.properties` **located in** *FederatedAccessManager-base*`/SUNWam/lib`**:**

- `iplanet-am-logging-remote-handler=com.sun.identity.`

  `log.handlers.RemoteHandler`
- `iplanet-am-logging-remote-formatter=com.sun.`

  `identity.log.handlers.RemoteFormatter`
- `iplanet-am-logging-remote-buffer-size=1`

Remote logging supports buffering on the basis of the number of log records. This value defines the log buffer size by the number of records. Once the buffer is full, all buffered records will be flushed to the server.

- `iplanet-am-logging-buffer-time-in-seconds=3600`

  This value defines the time-out period in which to invoke the log buffer-cleaner thread.

- `iplanet-am-logging-time-buffering-status=OFF`

  This value defines whether log buffering (and the buffer-cleaner thread) is enabled. By default this feature is turned off.

  If timer-based buffering is enabled (`iplanet-am-logging-time-buffering-status=ON`), then the buffer of log records is flushed (to the AM server providing the logging service) when the number of log records reaches the value specified in `iplanet-am-logging-remote-buffer-size`, or when the timer expires (timeout specified in `iplanet-am-logging-buffer-time-in-seconds`). If the timer expires before the buffer size is reached, then the records contained in the buffer are sent. If timer-base buffering of remote logging is disabled, then the buffer size determines when the buffer gets flushed. For example, if the buffer size is 10, and the application only sends 7 records, the buffer will not get flushed, nor the log records written. If the application terminates, then the records in the buffer will get flushed.

Note – Whenever a log file is empty, secure logging may show "verification failure." This is because when the number of created files is equal to the archive size, secure logging will archive from this set and start again. It most instances, you can ignore this error. Once the number of records is equal to the archive size, the error will not be displayed.

---

3 **If using a program with the Client SDK, the following properties in the AMConfig.properties file need to be set accordingly:**

- com.iplanet.am.naming.url
- com.sun.identityagents.app.username
- com.iplanet.am.service.password
- com.iplanet.am.server.protocol
- com.iplanet.am.server.host
- com.iplanet.am.server.port

Refer to the Client SDK samples `README.clientsdk` in the `/opt/SUNWam/war` directory. It details how the `AMConfig.properties` and the make files are generated for the `/opt/SUNWam/war/clientsdk-samples` directory. In turn, those files are used by the samples' makefiles' compile and run entries.

Error and Access Logs

Two types of Federated Access Manager log files exist: access log files and error log files.

Access log files record general auditing information concerning the Federated Access Manager deployment. A log may contain a single record for an event such as a successful authentication. A log may contain multiple records for the same event. For example, when an administrator uses the console to change an attribute value, the Logging Service logs the attempt to change in one record. Logging Service also logs the results of the execution of the change in a second record.

Error log files record errors that occur within the application. While an operation error is recorded in the error log, the operation attempt is recorded in the access log file.

Flat log files are appended with the `.error` or `.access` extension. Database table names end with `_ERROR` or `_ACCESS`. For example, a flat file logging console events is named `amConsole.access` while a database table logging the same events is named `AMCONSOLE_ACCESS` or `amConsole_access`.

The following table provides a brief description of the log file produced by each Federated Access Manager component.

TABLE 14–1 Federated Access Manager Component Logs

Component	Log Filename Prefix	Information Logged
Session	`amSSO`	Session management attributes values such as login time, logout time, timeout limits.
Administration Console	`amConsole`	User actions performed through the administration console such as creation, deletion and modification of identity-related objects, realms, and policies.
Authentication	`amAuthentication`	User logins and logouts.
Identity Federation	`amFederation`	Federation-related events such as the creation of an Authentication Domain and the creation of a Hosted Provider. The federation logs are prefixed with `amFederation`.
Authorization (Policy)	`amPolicy`	Policy-related events such as policy creation, deletion, or modification, and policy evaluation.
Policy Agent	`amAgent`	Exceptions regarding resources that were either accessed by a user or denied access to a user. `amAgent` logs reside on the server where the policy agent is installed. Agent events are logged on the Federated Access Manager machine in the Authentication logs.

TABLE 14–1 Federated Access Manager Component Logs *(Continued)*

Component	Log Filename Prefix	Information Logged
SAML	`amSAML`	SAML-related events such as assertion and artifact creation or removal, response and request details, and SOAP errors.
Command-line	`amAdmin`	Event errors that occur during operations using the amadmin command line tool. When flat file logging is specified, the amAdmin log files are placed in the `amadmincli` subdirectory under the main logging directory (default `/var/opt/SUNWam/logs`). Examples are: loading a service schema, creating policy, and deleting users.

See Access Manager Log File Reference in the *Federated Access Manager Administration Reference*for list and description of the Federated Access Manager log files.

Debug Files

The debug files are not a feature of the Logging Service. They are written using different APIs which are independent of the logging APIs. Debug files are stored in `/var/opt/SUNWam/debug`. This location, along with the level of the debug information, is configurable in the `AMConfig.properties` file, located in the *FederatedAccessManager-base*`/SUNWam/lib/` directory. For more information on the debug properties, see the AMConfig.properties file reference chapter in the *Federated Access Manager Administration Reference*..

Debug Levels

There are several levels of information that can be recorded to the debug files. The debug level is set using the `com.iplanet.services.debug.level` property in `AMConfig.properties`.

1. Off—No debug information is recorded.
2. Error—This level is used for production. During production, there should be no errors in the debug files.
3. Warning—Currently, using this level is not recommended.
4. Message—This level alerts to possible issues using code tracing. Most Federated Access Manager modules use this level to send debug messages.

Note – Warning and Message levels should not be used in production. They cause severe performance degradation and an abundance of debug messages.

Debug Output Files

A debug file does not get created until a module writes to it. Therefore, in the default `error` mode no debug files may be generated. The debug files that get created on a basic login with the debug level set to `message` include:

- amAuth
- amAuthConfig
- amAuthContextLocal
- amAuthLDAP
- amCallback
- amClientDetection
- amConsole
- amFileLookup
- amJSS
- amLog
- amLoginModule
- amLoginViewBean
- amNaming
- amProfile
- amSDK
- amSSOProvider
- amSessionEncodeURL
- amThreadManager

The most often used files are the `amSDK, amProfile` and all files pertaining to authentication. The information captured includes the date, time and message type (Error, Warning, Message).

Using Debug Files

The debug level, by default, is set to `error`. The debug files might be useful to an administrator when they are:

- Writing a custom authentication module.
- Writing a custom application using the Federated Access Manager SDKs. The `amProfile` and `amSDK` debug files capture this information.
- Troubleshooting access permissions while using the console or SDK. The `amProfile` and `amSDK` debug files also capture this information.
- Troubleshooting SSL.
- Troubleshooting the LDAP authentication module. The `amAuthLDAP` debug file captures this information.

The debug files should go hand in hand with any troubleshooting guide we might have in the future. For example when SSL fails, someone might turn on debug to message and look in the `amJSS` debug file for any specific certificate errors.

◆ ◆ ◆ CHAPTER 15

Notification Service

Sun Federated Access Manager 8.0 Notification Service allows for session notifications to be sent to remote web containers. It is necessary to enable this service for use by SDK applications running remotely from the Federated Access Manager server itself. This chapter explains how to enable a remote web container to receive the notifications. It contains the following sections:

Overview

The Notification Service allows for session notifications to be sent to web containers that are running the Federated Access Manager SDK remotely. The notifications apply to the Session, Policy and Naming Services only. In addition, the remote application must be running in a web container. The purpose of the notifications would be:

- To sync up the client side cache of the respective services.
- To enable more real time updates on the clients. (Polling is used in absence of notifications.)
- No client application changes are required to support notifications.

Note that the notifications can be received only if the remote SDK is installed on a web container.

Enabling The Notification Service

Following are the steps to configure the remote SSO SDK to receive session notifications.

▼ To Receive Session Notifications

1. **Install Federated Access Manager on Machine 1.**

2. **Install Sun Java System Web Server on Machine 2.**

3. **Install the `SUNWamsdk` on the same machine as the Web Server.**

   For instructions on installing the Federated Access Manager SDK remotely, see the *Sun Java Enterprise System 5 Installation Guide*.

4. **Ensure that the following are true concerning the machine where the SDK is installed.**

   a. **Ensure that the right access permissions are set for the /** ***remote_SDK_server*/** `SUNWam/lib` **and /** ***remote_SDK_server*** **/** `SUNWam/locale` **directories on the server where the SDK is installed.**

   These directories contains the files and jars on the remote server.

   b. **Ensure that the following permissions are set in the Grant section of the** `server.policy` **file of the Web Server.**

   `server.policy` is in the `config` directory of the Web Server installation. These permissions can be copied and pasted, if necessary:

   ```
   permission java.security.SecurityPermission
   "putProviderProperty.Mozilla-JSS"
   ```

   ```
   permission java.security.SecurityPermission "insertProvider.Mozilla-JSS";
   ```

   c. **Ensure that the correct classpath is set in** `server.xml`**.**

   `server.xml` is also in the `config` directory of the Web Server installation. A typical classpath would be:

   ```
   <JAVA javahome="/export/home/ws61/bin/https/jdk"
   serverclasspath="/export/home/ws61/bin/https/jar/webserv-rt.jar:
   ${java.home}/lib/tools.jar:/export/home/ws61/bin/https/jar/webserv-ext.jar:
   /export/home/ws61/bin/https/jar/webserv-jstl.jar:/export/home/ws61/
       bin/https/jar/nova.jar"
   classpathsuffix="::/IS_CLASSPATH_BEGIN_DELIM:
                   //usr/share/lib/xalan.jar:
                   //export/SUNWam/lib/xmlsec.jar:
                   //usr/share/lib/xercesImpl.jar:
                   //usr/share/lib/sax.jar:
                   //usr/share/lib/dom.jar:
                   //export/SUNWam/lib/dom4j.jar:
                   //export/SUNWam/lib/jakarta-log4j-1.2.6.jar:
                   //usr/share/lib/jaxm-api.jar:
                   //usr/share/lib/saaj-api.jar:
   ```

```
                    //usr/share/lib/jaxrpc-api.jar:
                    //usr/share/lib/jaxrpc-impl.jar:
                    //export/SUNWam/lib/jaxm-runtime.jar:
                    //usr/share/lib/saaj-impl.jar:/export/SUNWam
                    //lib:/export/SUNWam/locale:
                    //usr/share/lib/mps/jss3.jar:
                    //export/SUNWam/lib/    am_sdk.jar:
                    //export/SUNWam/lib/am_services.jar:
                    //export/SUNWam/lib/am_sso_provider.jar:
                    //export/SUNWam/lib/swec.jar:
                    //export/SUNWam/lib/acmecrypt.jar:
                    //export/SUNWam/lib/iaik_ssl.jar:
                    //usr/share/lib/jaxp-api.jar:
                    //usr/share/lib/mail.jar:
                    //usr/share/lib/activation.jar:
                    //export/SUNWam/lib/servlet.jar:
                    //export/SUNWam/lib/am_logging.jar:
                    //usr/share/lib/commons-logging.jar:
                    //IS_CLASSPATH_END_DELIM:"
    envclasspathignored="true" debug="false"
    debugoptions="-Xdebug -Xrunjdwp:
    transport=dt_socket,
    server=y,suspend=n"
    javacoptions="-g"
    dynamicreloadinterval="2">
```

5 **Use the SSO samples installed on the remote SDK server for configuration purposes.**

a. **Change to the** / *remote_SDK_server* /`SUNWam/samples/sso` **directory.**

b. **Run** `gmake`**.**

c. **Copy the generated class files from** / *remote_SDK_server* /`SUNWam/samples/sso` **to** / *remote_SDK_server* /`SUNWam/lib/`**.**

6 **Copy the encryption value of** `am.encryption.pwd` **from the** `AMConfig.properties` **file installed with Federated Access Manager to the** `AMConfig.properties` **file on the remote server to which the SDK was installed.**

The value of `am.encryption.pwd` is used for encrypting and decrypting passwords.

7 **Login into Federated Access Manager as** `amadmin`**.**

`http://`*AcceessManager-HostName*`:3000/amconsole`

8 **Execute the servlet by entering `http://` *remote_SDK_host*`:58080/servlet/SSOTokenSampleServlet` into the browser location field and validating the `SSOToken`.**

`SSOTokenSampleServlet` is used for validating a session token and adding a listener. Executing the servlet will print out the following message:

```
SSOToken host name: 192.18.149.33 SSOToken Principal name:
uid=amAdmin,ou=People,dc=red,dc=iplanet,dc=com Authentication type used: LDAP
IPAddress of the host: 192.18.149.33 The token id is
AQIC5wM2LY4SfcyURnObg7vEgdkb+32T43+RZN30Req/BGE= Property: Company is - Sun
Microsystems Property: Country is - USA SSO Token Validation test Succeeded
```

9 **Set the property `com.iplanet.am.notification.url=` in `AMConfig.properties` of the machine where the Client SDK is installed:**

```
com.iplanet.am.notification.url=http://clientSDK_host.domain:port
/servlet
    com.iplanet.services.comm.client.PLLNotificationServlet
```

10 **Restart the Web Server.**

11 **Login into Federated Access Manager as `amadmin`.**

`http://`*AcceessManager-HostName*`:3000/amconsole`

12 **Execute the servlet by entering `http://` *remote_SDK_host*`:58080/servlet/SSOTokenSampleServlet` into the browser location field and validating the `SSOToken` again.**

When the machine on which the remote SDK is running receives the notification, it will call the respective listener when the session state is changed. Note that the notifications can be received only if the remote SDK is installed on a web container.

▼ To Enable the Notification Service with a Portal-only Installation

This section describes the steps to enable notification with WebLogic 8.1 in a Portal-only installation, which by default runs in `polling` mode. For Portal instances that also contain the `amserver` component, these procedures are not needed. `amserver` components are automatically configured to perform notification.

1 **Register the `PLLNotificationServlet` in WebLogic.**

WebLogic 8.1 requires that a web application be deployed. Also, the servlet URL must be valid so that when accessed from a browser, the following message is returned:

```
Webtop 2.5 Platform Low Level notification servlet
```

2 **Enter the registered URL into `AMConfig.properties` as follows:**

com.iplanet.am.notifaction.url=http://*weblogic_instance-host.domain:port*/notification/PLL

3 **Disable polling in AMConfig.properties. This automatically enables notification:**

```
com.iplanet.am.session.client.polling.enable=false
```

4 **Restart WebLogic and test the configuration.**

If you have set the debug mode to `message`, you should see session notification arriving at the portal when triggered. For example, a action such as the termination of a user from the Federated Access Manager console will cause a notification event.

Index