Sun OpenSSO Enterprise 8.0 Administration Guide

Part No: 820–3885–11
March 2009

100222@23474

Contents

Preface 13

Part I Access Control 19

1 Logging In To The Console 21
Administrator Interface 21
User Interface 22
Legacy Support 22

2 Organizing Data within Realms 23
Understanding Realms 23
Creating and Modifying Realms 24
▼ To Create a New Realm 24
▼ To Modify a Realm's General Properties 25
Managing Configuration Data Within Realms 25
Managing Authentication 26
Adding Services 26
Plugging in Data Stores 28
Delegating Administrator Privileges 30
Configuring Policy 32
Defining Subjects 32
Creating Agent Profiles 33

3 Configuring Authentication 35
Understanding the Authentication Service 35
Authentication Service User Interface 36
Authentication Modules 36

Authentication Types 42
Authentication Post Processing 43
Configuring the Core Authentication Service 43
▼ To Modify Core Authentication Properties Globally 44
▼ To Modify Core Authentication Properties By Realm 45
Configuring the Authentication Process 55
Before You Begin 55
Configuring Authentication Modules 60
Creating Authentication Chains 62
Initiating the Authentication Type 64
Realm Authentication 64
Service Authentication 67
User Authentication 69
Authentication Level-based Authentication 72
Module Authentication 75
Role Authentication (Legacy Mode) 77
Accessing the Authentication Service User Interface with a Login URL 80
`goto` Parameter 81
`gotoOnFail` Parameter 81
`realm` Parameter 82
`user` Parameter 82
`locale` Parameter 83
`module` Parameter 83
`service` Parameter 84
`arg` Parameter 84
`authlevel` Parameter 84
`forceAuth` Parameter 85
`IDToken`*N* Parameters 85
`iPSPCookie` Parameter 86
`PersistAMCookie` Parameter 86
`role` Parameter (Legacy Mode) 86
`org` Parameter (Legacy Mode) 87
`domain` Parameter (Legacy Mode) 87
Customizing Authentication 87
Enabling Account Lockout 87
Authentication Service Failover 89

Mapping Fully Qualified Domain Names 89
Using Persistent Cookies 90
Upgrading Sessions 91
Sharing User Credentials Among Authentication Modules (Shared State) 92
Redirecting Users After Authentication 93
Configuring Multiple LDAP Authentication Modules (Legacy Mode) 93

4 Managing Policies 97
Understanding the Authorization Process 97
Defining a Policy and a Referral 99
Policy 99
Referral 106
Creating Policies and Referrals 107
▼ To Add Multiple Policies Using the `ssoadm` Command Line Utility 107
▼ To Create a Policy Using the OpenSSO Enterprise Console 111
▼ To Create a Referral Using the OpenSSO Enterprise Console 112
Modifying Policies and Referrals 114
Modifying Policies 114
Modifying Referrals 117
Using Wild Cards in Policies 119
Applying Policy Logic 119
Enabling Policy in a Service 120
▼ To Add a New Policy Enabled Service 121
Authenticating Based on Resource 123
▼ To Configure Resource Authentication 123

5 Creating Subjects 127
Storing Subjects 127
Creating Users 127
▼ To Create a User 128
▼ To Modify a User 128
Creating Groups 130
▼ To Create a Group 130
▼ To Add Users to a Group 130
Administrative Users and Default Subjects 131

`amadmin` ... 131

`amldapuser` ... 132

`UrlAccessAgent` ... 132

`Directory Manager` ... 133

`Administrator` ... 133

`demo` ... 133

`test` ... 133

`dsameuser` ... 133

`puser` ... 134

`anonymous` ... 134

6 Storing Policy Agent and Web Services Security Agent Profiles ... 135

Centralizing Agent Profiles ... 135

Web Policy Agent Profile ... 136

J2EE Policy Agent Profile ... 136

Web Service Provider Security Agent Profile ... 136

Web Service Client Security Agent Profile ... 136

STS Client Agent Profile ... 137

2.2 Agents ... 138

Agent Authenticator ... 138

Creating New Agent Profiles and Groups ... 138

▼ To Create a New Agent Profile ... 138

▼ To Create a New Group ... 140

▼ To Modify an Agent Profile to Inherit Properties From a Group ... 141

Part II Federation, Web Services, and SAML Administration ... 143

7 Configuring and Managing Federation ... 145

Configuring Federation ... 145

Managing Federation Using the Console ... 146

Creating an Entity ... 146

Circle of Trust ... 152

Managing Federation Using `ssoadm` ... 155

Managing Entity Metadata using `ssoadm` ... 155

Managing Circles of Trust Using `ssoadm` 160

8 **Federated Operations** 163
Finding an Identity Provider for Authentication 163
Configuring the SAMLv2 Identity Provider Discovery Service 164
Configuring the ID-FF Identity Provider Introduction Service 164
Configuring WS-Federation Home Realm Discovery Service 165
Customizing SAMLv2 the Identity Provider Discovery Service and the ID-FF Identity Provider Introduction Service 166
Bulk Federation 167
ID-FF Federation Operations 168
The Pre-Login URL 168
Configuring ID-FF Single Sign-on 171
ID-FF Auto-Federation 172
Enabling ID-FF XML Signing 174
Dynamic Identity Provider Proxying 175
SAMLv2 Operations 179
POST Binding with Single Sign-on and Single Logout 180
Creating Affiliations 181
Requesting Attribute Values Using a SAMLv2 Assertion 182
Requesting a SAMLv2 Assertion 185
Requesting a SAMLv2 Assertion for Authentication Context 188
Encoding Artifacts 189
Managing SAMLv2 Name Identifiers 189
Mapping SAMLv2 Name Identifiers 191
Enhanced Client and Proxy 192
Formatting Name Identifiers 194
Configuring SAMLv2 Single Sign-on without Service Provider User Accounts 195
Auto-creation of User Accounts 198
Using Non-Default Federation Attributes 199
Enabling XML Signing and Encryption 200
Securing SOAP Binding 200
Load Balancing 201
Access Control 203
Certificate Revocation List Checking 206

SAMLv2 IDP Discovery Service 211
Bootstrapping the Liberty ID-WSF with SAML v2 212
WS-Federation Operations 218
▼ To Configure OpenSSO Enterprise as a Service Provider 219
▼ To Configure OpenSSO Enterprise as an Identity Provider 222

9 Identity Web Services 227
Authentication Web Service 227
Authentication Web Service Attribute 228
Liberty Personal Profile Service 229
Liberty Personal Profile Service Attributes 229
Discovery Service 233
Discovery Service Attributes 234
Storing Resource Offerings 237
SOAP Binding Service 244
SOAP Binding Service Attributes 244
Liberty Interaction Service 247
Liberty ID-WSF Security Service 248

10 SAML 1.x Administration 249
SAML Attributes 249
Target Specifier 250
Site Identifiers 251
Trusted Partners 251
Target URLs 255
Default Protocol Version 256
Default Assertion Version 256
Remove Assertion 256
Assertion Timeout 256
Assertion Skew Factor for `notBefore` Time 256
Artifact Timeout 257
SAML Artifact Name 257
Sign SAML Assertion 257
Sign SAML Request 257
Sign SAML Response 257

Attribute Query 257
SAML Operations 258
Setting Up SAML Single Sign-on 258

Part III Directory Management and Default Services 263

11 Directory Management 265
Managing Directory Objects 265
Organizations 265
Containers 268
Group Containers 269
Groups 270
People Containers 273
Users 274
Roles 277

12 Current Sessions 285
The Current Sessions Interface 285
Session Management 285
Session Information 285
Terminating a Session 286

13 Password Reset Service 287
Registering the Password Reset Service 287
▼ To Register Password Reset for Users in a Different Realm 287
Configuring the Password Reset Service 288
▼ To Configure the Service 288
▼ To Localize the Secret Question 289
Password Reset Lockout 289
Password Policies 290
▼ Example: To Create a Password Policy in Directory Server for Force Password Change After Reset 290
Password Reset for End Users 291
Customizing Password Reset 291

Resetting Forgotten Passwords 292

14 Logging Service 295
Log Files 295
OpenSSO Enterprise Logs 296
Session Logs 296
Console Logs 296
Authentication Logs 297
Federation Logs 297
Policy Logs 297
Agent Logs 297
SAML Logs 297
`ssoadm` Logs 297
Logging Features 298
Secure Logging 298
Logging Level Attributes and Properties 299
Database Logging 300
Remote Logging 300
Debug Files 301
Debug Levels 301
Debug Output Files 301

15 Backing Up and Restoring Configuration Data 303
Understanding Backup and Restore 303
Backing Up the Configuration Datastore 304
▼ To Backup the Configuration Datastore 304
Restoring the Configuration Data Store 305
▼ To Restore the Embedded Configuration Datastore by Loading XML 305
▼ To Restore by Replication of the OpenSSO Configuration Data store 306
▼ To Restore the Directory Server Configuration Datastore by Loading XML 307
▼ To Restore by Replication of the Directory Server Configuration Datastore 308

A Changing the Host Name of an OpenSSO Instance 309

Index 313

Preface

The *Sun OpenSSO 8.0 Administration Guide* provides information on how to use the OpenSSO Enterprise Administration Console as well as how to manage user and service data using the command-line interface (CLI).

Contents

- "Who Should Use This Guide" on page 13
- "Before You Read This Guide" on page 13
- "Related Documentation" on page 14
- "Searching Sun Product Documentation" on page 16
- "Related Third-Party Web Site References" on page 16
- "Documentation, Support, and Training" on page 16
- "Default Paths and Directory Names" on page 17
- "Sun Welcomes Your Comments" on page 18

Who Should Use This Guide

This guide is intended for system administrators, system integrators, and others who are installing and configuring OpenSSO Enterprise.

Before You Read This Guide

Readers should be familiar with the following components and concepts:

- OpenSSO Enterprise technical concepts, as described in the *OpenSSO Enterprise 8.0 Technical Overview*
- Deployment platform: Solaris™, Linux, or Windows operating system
- Web container that will run OpenSSO Enterprise, such as 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™ technology, JavaServer Pages™ (JSP™) technology, HyperText Transfer Protocol (HTTP), HyperText Markup Language (HTML), and eXtensible Markup Language (XML)

Related Documentation

Related documentation is available as follows:

- "OpenSSO Enterprise Documentation Set" on page 14
- "Related Product Documentation" on page 15

OpenSSO Enterprise Documentation Set

The following table describes the OpenSSO Enterprise documentation set.

TABLE P–1 OpenSSO Enterprise Documentation Set

Title	Description
Sun OpenSSO Enterprise 8.0 Release Notes	Describes new features, installation notes, and known issues and limitations. The Release Notes are updated periodically after the initial release to describe any new features, patches, or problems.
Sun OpenSSO Enterprise 8.0 Installation and Configuration Guide	Provides information about installing and configuring OpenSSO Enterprise. about, including OpenSSO Enterprise server, Administration Console only, client SDK, scripts and utilities, Distributed Authentication UI server, and session failover.
Sun OpenSSO Enterprise 8.0 Technical Overview	Provides an overview of how components work together to consolidate access control functions, and to protect enterprise assets and web-based applications. It also explains basic concepts and terminology.
Sun OpenSSO Enterprise 8.0 Deployment Planning Guide	Provides planning and deployment solutions for OpenSSO Enterprise.
Sun OpenSSO Enterprise 8.0 Administration Guide	Describes how to use the OpenSSO Enterprise Administration Console as well as how to manage user and service data.
Sun OpenSSO Enterprise 8.0 Administration Reference	Provides reference information for the OpenSSO Enterprise command-line interface (CLI), configuration attributes, log files, and error codes.
Sun OpenSSO Enterprise 8.0 Developer's Guide	Provides information about customizing OpenSSO Enterprise and integrating its functionality into an organization's current technical infrastructure. It also provides details about the programmatic aspects of the product and its API.
Sun OpenSSO Enterprise 8.0 C API Reference for Application and Web Policy Agent Developers	Provides summaries of data types, structures, and functions that make up the public OpenSSO Enterprise C APIs.
Sun OpenSSO Enterprise 8.0 Java API Reference	Provides information about the implementation of Java packages in OpenSSO Enterprise.

TABLE P–1 OpenSSO Enterprise Documentation Set *(Continued)*

Title	Description
Sun OpenSSO Enterprise 8.0 Performance Tuning Guide	Provides information about how to tune OpenSSO Enterprise and its related components for optimal performance.
Sun OpenSSO Enterprise Policy Agent 3.0 User's Guide for J2EE Agents	Provide an overview of version 3.0 web and J2EE policy agents.
Sun OpenSSO Enterprise Policy Agent 3.0 User's Guide for Web Agents	

Related Product Documentation

The following table provides links to documentation collections for related products.

TABLE P–2 Related Product Documentation

Product	Link
Sun Java System Directory Server 6.3	http://docs.sun.com/coll/1224.4
Sun Java System Web Server 7.0 Update 3	http://docs.sun.com/coll/1653.3
Sun Java System Application Server 9.1	http://docs.sun.com/coll/1343.4
Sun Java System Message Queue 4.1	http://docs.sun.com/coll/1307.3
Sun Java System Web Proxy Server 4.0.6	http://docs.sun.com/coll/1311.6
Sun Java System Identity Manager 7.1	http://docs.sun.com/coll/1514.3

Searching Sun Product Documentation

Besides searching Sun product documentation from the docs.sun.com^{SM} web site, you can use a search engine by typing the following syntax in the search field:

```
search-term site:docs.sun.com
```

For example, to search for "broker," type the following:

```
broker site:docs.sun.com
```

To include other Sun web sites in your search (for example, java.sun.com, www.sun.com, and developers.sun.com), use `sun.com` in place of `docs.sun.com` in the search field.

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–3 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–4 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	`#`

Default Paths and Directory Names

The OpenSSO Enterprise documentation uses the following terms to represent default paths and directory names.

TABLE P–5 Default Paths and Directory Names

Term	Description
zip-root	Represents the directory where the `opensso.zip` file is unzipped.

OpenSSO-Deploy-base	Represents the deployment directory where the web container deploys the `opensso.war` file.
	
	This value varies depending on the web container. To determine the value of *OpenSSO-Deploy-base*, view the file name in the `.openssocfg` directory, which resides in the home directory of the user who deployed the `opensso.war` file. For example, consider this scenario with Application Server 9.1 as the web container:
	■ Application Server 9.1 is installed in the default directory: `/opt/SUNWappserver`.
	■ The `opensso.war` file is deployed by super user (`root`) on Application Server 9.1.
	
	The `.openssocfg` directory is in the `root` home directory (`/`), and the file name in `.openssocfg` is:
	
	`AMConfig_opt_SUNWappserver_domains_domain1_applications_j2ee-modules_opensso_`
	
	Then, the value for *OpenSSO-Deploy-base* is:
	
	`/opt/SUNWappserver/domains/domain1/applications/j2ee-modules/opensso`

ConfigurationDirectory	Represents the name of the configuration directory specified during the initial configuration of OpenSSO Enterprise server instance using the Configurator. The default is `opensso` in the home directory of the user running the Configurator. Thus, if the Configurator is run by `root`, *ConfigurationDirectory* is `/opensso`.

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 guide or at the top of the document.

For example, the title of this guide is the *Sun OpenSSO Enterprise 8.0 Administration Guide*, and the part number is 820–3885.

PART I

Access Control

This is part one of the Sun OpenSSO Enterprise 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, OpenSSO Enterprise 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, "Logging In To The Console"
- Chapter 2, "Organizing Data within Realms"
- Chapter 3, "Configuring Authentication"
- Chapter 4, "Managing Policies"
- Chapter 5, "Creating Subjects"
- Chapter 6, "Storing Policy Agent and Web Services Security Agent Profiles"

CHAPTER 1

Logging In To The Console

The OpenSSO Enterprise console is a web interface that allows both administrators and users without administrative privileges to log in and manage their respective entitlements. Depending on the level of access defined, an administrator can create realms, modify user membership in the realms, configure authentication chains, define data stores, and establish enforcement policies to protect and limit access to the realm's resources. In addition, administrators can view and terminate current user sessions and manage their federation configurations (create, delete and modify authentication domains and entity providers). Users without administrative privileges, on the other hand, can manage personal information (name, email address, password, telephone number, and so forth), subscribe and unsubscribe to groups, and view their roles. The following sections describe the different console views this functionality offers.

- "Administrator Interface" on page 21
- "User Interface" on page 22
- "Legacy Support" on page 22

Administrator Interface

When a user with an administrative role authenticates to OpenSSO Enterprise, the administrator view is displayed, by default. Using this interface, the administrator can accomplish most administrative tasks related to OpenSSO Enterprise. This includes realm-based access control, global service configuration, and web services and federation management. To access the Administrator Interface login screen, use *protocol*`://`*machine-name*`:`*port*`/`*uri*`/UI/Login` as the URL syntax where *protocol* is either `http` or `https` depending upon your deployment.

User Interface

When a user who has not been assigned an administrative role authenticates to OpenSSO Enterprise, the user's own profile is displayed. Using this interface, a user can modify the values of the personal profile attributes. This can include, but is not limited to, name, home address and password. To access the User Profile Interface login screen, use *protocol*`://`*machine-name*`:`*port*`/`*uri*`/`**`UI/Login`** as the URL syntax where *protocol* is either `http` or `https` depending upon your deployment. Although the URL for both the Administrator and Personal Profile interfaces is the same, entering the username and password of a user who has not been assigned an administrative role will direct the user to the User Profile interface.

Tip – The User Profile interface is based on information defined in the `amUser.xml` service file.

Legacy Support

Legacy support in OpenSSO Enterprise is based on the Sun Java System Access Manager 6.3 architecture. When OpenSSO Enterprise is deployed in legacy mode, the console is different from when it is installed in the default realm mode. In OpenSSO Enterprise 8.0 legacy mode is supported through upgrade only; if you have Access Manager 7.0 or 7.1 installed in legacy mode, you can upgrade to OpenSSO Enterprise 8.0 and keep legacy mode. For more information, see the *Sun OpenSSO Enterprise 8.0 Upgrade Guide*.

Caution – Legacy support is deprecated and will be removed in a future release. It is strongly recommended not to use this option.

CHAPTER 2

Organizing Data within Realms

A *realm* is the unit that OpenSSO Enterprise uses to manage resources. The following sections describe how to manage OpenSSO Enterprise using the realm mode.

After OpenSSO Enterprise is deployed and configured, the default `/ (Top Level Realm)` is created. The top-level realm is the root realm of the OpenSSO Enterprise instance; it cannot be changed after it is created. In general, you should use the top-level realm to configure identity data stores, and manage policies and authentication chains.

- "Understanding Realms" on page 23
- "Creating and Modifying Realms" on page 24
- "Managing Configuration Data Within Realms" on page 25

For more information on realms, see "Realms" in *Sun OpenSSO Enterprise 8.0 Technical Overview*.

Understanding Realms

A *realm* is the administrative unit for OpenSSO Enterprise. After OpenSSO Enterprise is deployed and configured, the default `/ (Top Level Realm)` is created. The top-level realm is the root realm that, with the exception of bootstrapping information configured during installation, contains the configuration data for the OpenSSO Enterprise instance. The top-level realm can contain sub realms. Sub realms under the top-level realm can also contain sub realms. Information that can be defined in the top-level or a sub realm includes:

- The location of one or more identity repositories containing users, groups, and roles to whom the remaining configuration information applies.

  Note – Roles are a feature of Sun Java System Directory Server but not Active Directory. Identity repositories defined in Active Directory cannot include roles.

- An authentication process that defines, among other information, the location of an authentication repository and the type of authentication required.
- Policies that are used to determine whether an authenticated user can access protected resources.
- Privileges to define administrative permissions within the realm.
- Configuration data for services that can be modified per realm. This includes the realm Administration Service, the Discovery Service, Globalization Settings, the Password Reset Service, the Session Service, and the User Profile.

The hierarchy of a top-level and sub realms can be used to identify users and groups with different authentication and authorization requirements. For example, users in the Human Resources department have access to more sensitive data than other users in an organization. By creating a sub realm for Human Resources personnel, you can enforce an authentication chain that might include entering an LDAP user identifier and password followed by a token generated using a SafeWord card. On the other hand, users not defined in this sub realm might need only enter an LDAP user identifier and password to access their own personal profile. The use of sub realms should be restricted to either of the following scenarios:

- Application-specific policy management in which the creation and delegation of policies for a particular application (for example, a vacation recording tool) is referred to a sub realm. In other words, policy administrators are created in the top-level realm and appointed a sub set of resources for which they can manage access. By default, the sub realm would have the same configuration data store and authentication chains as the top-level realm and all users login to the top-level realm to access the applications.
- Silo management where each sub realm has its own set of data stores (identities), authentication chains and policy management. A user, in this case, would authenticate to their respective sub realm.

Creating and Modifying Realms

This section describes how to create a realm and modify its General properties post creation.

- “To Create a New Realm” on page 24
- “To Modify a Realm's General Properties” on page 25

▼ To Create a New Realm

Before You Begin This procedure assumes you are logged into the OpenSSO console as the administrator, `amAdmin`.

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	Select the parent realm under which the new realm will exist.

3 **Define the following Realm attributes:**

Realm Status	Select 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. Selecting Inactive disables user access when logging in.
Realm/DNS Aliases	Allows you to configure a DNS-type alias for the realm. This attribute only accepts “real” domain aliases (as in `.sun.com`). Random strings are not allowed.

4 **Click OK to save the realm.**

▼ To Modify a Realm's General Properties

Before You Begin This procedure assumes you are logged into the OpenSSO console as the administrator, `amAdmin`.

1 **Click the Access Control tab.**

2 **Click the name of the realm that contains the properties to be modified.**

3 **Edit the appropriate values.**

4 **Click OK to save the realm.**

Managing Configuration Data Within Realms

To manage configuration data stored within a realm, click the Realm Name under the Access Control tab in the OpenSSO Enterprise console. The following sections contain more information (or links to more information) regarding the configuration data.

- “Managing Authentication” on page 26
- “Adding Services” on page 26
- “Plugging in Data Stores” on page 28
- “Delegating Administrator Privileges” on page 30
- “Configuring Policy” on page 32
- “Defining Subjects” on page 32
- “Creating Agent Profiles” on page 33

Managing Authentication

Authentication properties and processes may be customized by realm. Default values for authentication are defined in the Core authentication module under the Configuration tab in the OpenSSO Enterprise console. These values will be inherited when a new realm is created but, you can modify a particular value per realm. For more information on configuring for authentication and managing authentication processes, see Chapter 3, "Configuring Authentication."

Adding Services

A number of services can be added to a realm for more fine-grained configuration. These services may contain Global attributes (which are common to the OpenSSO Enterprise instance and inherited by all configured realms), Realm attributes (which can be customized per realm after the service has been added to it) and Dynamic attributes (which are inherited by users that belong to the realm in which the value is defined). Default values for all attributes in these services can be defined under the Configuration tab in the OpenSSO Enterprise console. The services that can be added to a realm include:

Administration	The Administration service is used to customize tasks performed by the OpenSSO Enterprise console. Default values for the Administration service are defined for the console under the Configuration tab in the OpenSSO Enterprise console. By adding the Administration service to a realm, the realm's administrator can customize these values per realm.
Discovery Service	An *identity service* is a web service that supports the query and modification of data regarding a principal. An identity service might host, for example, a principal's profile, such as name, address and phone number, or it might hold more sensitive information like a credit card number. The initial step in accessing the identity data a client is requesting is to determine in which identity service it is hosted. A *resource offering* defines the association between a piece of identity data and the identity service that provides access to it. A *discovery service* is a registry of resource offerings. By adding the Discovery Service to a realm, the realm's administrator can add resource offerings at the realm level as opposed to the user level. **Note –** This functionality is designed for business to business use cases.

Globalization Settings	The Globalization Settings service contains attributes to customize OpenSSO Enterprise for different locales and character sets. Default values for the Globalization Settings service are defined for the console under the Configuration tab in the OpenSSO Enterprise console. By adding the Globalization Settings service to a realm, the realm's administrator can customize these values per realm.
Password Reset	The Password Reset service allows users to reset their configured password or to receive an email message containing a new password. The Password Reset service does not need to be added to the realm in which a user resides to work. If the Password Reset service is not added to the realm in which the user resides, it will inherit the attribute values defined globally for the service under the Configuration tab in the OpenSSO Enterprise console. By adding the Password Reset service to a realm, the realm's administrator can customize these values per realm.
Policy Configuration	The Policy Configuration service is added to a realm, by default, when the realm is created. Default values for the Policy Configuration service are defined globally under the Configuration tab in the OpenSSO Enterprise console but the realm's administrator can customize them as needed. The service contains properties related to the Policy Service itself.
Session	The Session service defines values for properties that pertain to an authenticated user's session. This includes information such as maximum session time and maximum idle time. Default values for the Session service are defined globally under the Configuration tab in the OpenSSO Enterprise console. By adding the Session service to a realm, the realm's administrator can customize certain properties per realm.
User	Default user preferences for properties like time zone and locale are defined with the User service. Default values for the User service are defined globally under the Configuration tab in the OpenSSO Enterprise console. By adding the User service to a realm, the realm's administrator can customize certain properties per realm.

The following procedures pertain to adding and managing a realm's services.

- "To Add a Service to a Realm" on page 28
- "To Modify the Attributes of a Realm's Added Services" on page 28

▼ To Add a Service to a Realm

Before You Begin This procedure assumes you are logged into the OpenSSO console as the administrator, `amAdmin`.

1. **Click the Access Control tab.**
2. **Click the name of the realm to which the service will be added.**
3. **Click the Services tab.**
4. **Click Add in the Services list.**
5. **Select the service you want to add.**
6. **Click Next.**
7. **Configure the service by defining values for the appropriate attributes.**
8. **Click Finish.**

   The service will be listed under Services.

▼ To Modify the Attributes of a Realm's Added Services

Before You Begin This procedure assumes you are logged into the OpenSSO console as the administrator, `amAdmin`.

1. **Click the Access Control tab.**
2. **Click the name of the realm that contains the service to be modified.**
3. **Click the Services tab.**
4. **Click the name of the service you are modifying.**
5. **Edit the appropriate values.**
6. **Click Save to save the new values.**

Plugging in Data Stores

A data store is a database where you can store user attributes and user configuration data. OpenSSO Enterprise provides identity repository plug-ins that connect to an LDAPv3 identity repository framework. These plug-ins enable you to view and retrieve OpenSSO Enterprise user information without having to make changes in your existing user database. The OpenSSO Enterprise framework integrates data from the identity repository plug-in with data from other OpenSSO Enterprise plug-ins to form a virtual identity for each user. OpenSSO Enterprise 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.

An *identity repository* is a data store where information about users is stored. The data store might contain, for example, a user identifier and password, email address, application preferences and other forms of identity data. OpenSSO Enterprise provides an interface that enables a realm administrator to plug one or more identity data stores in to a realm. These plug-ins enable you to view and retrieve OpenSSO Enterprise user information without having to make changes in your existing user database. The OpenSSO Enterprise framework integrates data from the identity repository plug-in with data from other OpenSSO Enterprise plug-ins to form a virtual identity for each user. Because the plug-ins allow more than one identity data store to be configured per realm, OpenSSO Enterprise can access the many profiles of one identity across multiple identity repositories. This allows for the virtual identity for each user to be accessed for purposes of authentication and authorization. You can create a new data store instance using the following data store types:

Active Directory	This data store type uses the LDAP version 3 specification to write identity data to an instance of Microsoft Active Directory.
Generic LDAPv3	This data store type allows identity data to be written to any LDAPv3–compliant database. **Note –** If the LDAPv3 database you are using does not support Persistent Search, then you can not use the caching feature.
Sun Directory Server With OpenSSO Schema	This data store type resides in a Sun Directory Server instance and holds the OpenSSO Enterprise information tree. It differs from the OpenSSO Enterprise Repository Plug-in, in that more configuration attributes allow you to better customize the data store.

The following procedure documents how to configure a new data store.

▼ To Create a New Data Store

Before You Begin This procedure assumes you are logged into the OpenSSO console as the administrator, `amAdmin`.

1. **Click the Access Control tab.**

2. **Click the name of the realm in which you want to add a new data store.**

3. **Click the Data Stores tab.**

4. **Click New from the Data Stores list.**

5. **Enter a name for the data store.**

6. **Select the type of data store you wish to create.**

7. **Click Next.**

8. **Configure the data store by entering the appropriate attribute values.**

   See the *Sun OpenSSO Enterprise 8.0 Administration Reference* for attribute definitions.

9. **Click Finish.**

Delegating Administrator Privileges

OpenSSO Enterprise administrators are delegated responsibilities based on privileges assigned to groups. A *privilege* is an action that can be performed on a resource; for example, a READ operation on a log. Privileges can be dynamically assigned to users deemed administrators by creating a group, assigning to it the appropriate privilege, and adding the appropriate user as a member of the group.

Note – For more information on groups, see Chapter 5, "Creating Subjects."

Once a group is created, it appears under the realm's Privileges tab. To add privileges, click the group name and assign the appropriate operation. Members belonging to the group would then be able to perform the assigned operation(s). The following privileges can be delegated.

- *Read and write access to all configured agents* delegates read and write permissions for all configured agent profiles to an agent administrator.

- *Read and write access to all log files* delegates read and write permissions for all log records to a log administrator.

Caution – OpenSSO Enterprise logging interfaces are public so it is *possible* that any authenticated user can read and write OpenSSO Enterprise log records. The log administrator privileges prevent this abuse. Policy agents, the main users of the logging interfaces, only require permission to write log records, and should not be delegated the permission to read them. Similarly, administrators who read log records should not be delegated the permission to write to them.

- *Read access to all log files* delegates read permission for all log records to a log administrator.
- *Write access to all log files* delegates write permission for all log records to a log administrator.
- *Read and write access only for policy properties* delegates read and write permissions for all policies and policy configurations to a policy administrator. Policy administrators can create, modify and delete policies which consists of Rules, Subjects, Conditions and Response Attributes.

Caution – In order to manage the policies themselves (not policy configurations), a policy administrator needs permission to read the identity data store(s) and should be delegated the *Read and write access to all realm and policy properties* permission.

- *Read and write access to all realm and policy properties* delegates read and write permissions for all realm configurations data to a realm administrator. Realm administrators can create sub-realms, modify configurations for the realm's services and create, modify and delete Users, Groups, and Agents.

Note – If you have upgraded Access Manager from version 7.0 to OpenSSO Enterprise, the privilege configuration differs from that of a fresh installation. To assign or modify privileges, click the name of the role or group you wish to edit and select from the following:

- *Read only access to data stores* defines read access privileges to data stores. This privilege definition is for use only with *Read and write access only for policy properties* to control delegation for policy administrators.
- *Read and write access to all log files* defines both read and write permission to log records for log administrators.
- *Write access to all log files* defines write permission to log records for log administrators.
- *Read access to all log files* defines only read permission to log records for log administrators.
- *Read and write access only for policy properties* defines read and write permission regarding policies and policy configurations for policy administrators.
- *Read and write access to all realm and policy properties* defines read and write permissions to all realm configurations data for realm administrators.
- *Read only access to all properties and services* defines read permission to all properties and services. This privilege definition is for use only with *Read and write access only for policy properties* to control delegation for policy administrators.

Configuring Policy

A *policy* defines rules that specify who is authorized to access an organization's resources, including applications and services. Policies control this access by defining who can do what to which resource, and when and how it can be done. In OpenSSO Enterprise resources are defined and policies are created at the realm (or sub realm) level. The Policy Service enables the top-level administrator or policy administrator to create, delete, modify and view policies for a specific resource within the realm. For more information on configuring policies, see Chapter 4, "Managing Policies."

Defining Subjects

OpenSSO Enterprise offers a basic identity management functionality. Subjects can be defined within a realm: a *user* represents an individual identity and a *group* represents a collection of users with a common function, feature or interest. A subject can be used in the definition for a policy. For more information on defining subjects, see Chapter 5, "Creating Subjects."

Note – OpenSSO Enterprise is not a user provisioning product. This basic identity management functionality should only be used for demonstrations and proof of concepts. The user provisioning solution provided by Sun Identity Manager should be used in real deployments. For more information see Sun Identity Manager.

Creating Agent Profiles

Policy agents and web services security agents function based on properties. The configuration of these properties is centralized and managed using the OpenSSO Enterprise console or command line interface; the defined values are stored in the embedded configuration data store. Agent profiles are added to a realm and managed by the realm administrator. For more information on adding agent profiles, see Chapter 6, "Storing Policy Agent and Web Services Security Agent Profiles."

◆ ◆ ◆ CHAPTER 3

Configuring Authentication

The function of the OpenSSO Enterprise Authentication Service is to retrieve credentials from an authenticating party (a user, administrator, or client application), and validate them against a configured repository. Towards this end, the OpenSSO Enterprise console is used to specify the authentication process by instantiating an *authentication module*, and using the instantiated modules to establish an *authentication chain* (one or more authentication module instances configured so that a user must pass authentication credentials to all of them). The following sections describe how to configure and manage authentication for your deployment.

- "Understanding the Authentication Service" on page 35
- "Configuring the Core Authentication Service" on page 43
- "Configuring the Authentication Process" on page 55
- "Initiating the Authentication Type" on page 64
- "Accessing the Authentication Service User Interface with a Login URL" on page 80
- "Customizing Authentication" on page 87

Understanding the Authentication Service

The core function of the Authentication Service is to validate the required credentials of end users, administrators or applications requesting access to information or protected resources in the OpenSSO Enterprise single sign-on environment. When creating the end-to-end authentication process by which a user (for example) will be authenticated, there are things to consider such as where the authentication process will be initiated, which authentication module(s) will be used in the process, and what, if any, post authentication steps will be implemented. The authentication type (where the authentication process will be initiated) is specified by appending an appropriate parameter to the login URL, or through the authentication API. The authentication process is configured (simply) by instantiating the desired authentication module(s) or (with more complexity) by creating an *authentication chain*. Post authentication processing steps are developed programmatically through implementation of a provided interface. The following sections contain information on these considerations.

- "Authentication Service User Interface" on page 36
- "Authentication Modules" on page 36
- "Authentication Types" on page 42
- "Authentication Post Processing" on page 43

A detailed technical explanation of the authentication process through session validation and policy evaluation is in Chapter 6, "Models of the User Session and Single Sign-On Processes," in *Sun OpenSSO Enterprise 8.0 Technical Overview*.

Authentication Service User Interface

The Authentication Service user interface provides a means for gathering the credentials needed by the Authentication Service to validate a user. In most cases, the Authentication Service user interface is accessed by entering a login URL into the Location Bar of a web browser. A URL parameter may be appended to the end of the URL to define, for example, specific authentication types or redirection URLs. The format of this URL is:

```
http://OpenSSO-server..domain_name:port/service_deploy_uri/UI/Login?parameter1=value1&parameter2=value2&parameterN=valueN
```

Note – During installation, the *service_deploy_uri* is configured as `opensso`.

After entering this URL, login screens are dynamically displayed based on the invoked authentication module, and a user is able to communicate an identifier and password, for example, back to the Authentication Service to receive (or not receive) validation. For more information on accessing the Authentication Service, see "Accessing the Authentication Service User Interface with a Login URL" on page 80.

Authentication Modules

An *authentication module* is a plug-in that collects information from a principal requesting access to a protected resource and checks the information against entries in a data store. If the information provided meets the authentication criteria, the user is validated. If the information provided does not meet the authentication criteria, the user is denied validation. OpenSSO Enterprise provides a number of authentication modules. You instantiate an authentication module before accessing it for a simple authentication process (see "Authentication Types" on page 42), or adding it to an authentication chain. (In an authentication chain, you can configure, for example, two instances of the LDAP authentication module — each pointing to a different LDAP data store configured within the same realm; or you can configure different authentication modules so a user must pass authentication credentials to all of them for validation.)

The following sections contain information about the individual authentication modules provided by OpenSSO Enterprise. Some of them require configuration before they can be instantiated. Where applicable, the configuration steps are listed in the module type descriptions.

- "Active Directory" on page 37
- "Anonymous" on page 37
- "Certificate" on page 38
- "Data Store" on page 39
- "Federation" on page 39
- "HTTP Basic" on page 39
- "JDBC" on page 39
- "LDAP" on page 40
- "Membership" on page 40
- "MSISDN" on page 40
- "RADIUS" on page 40
- "SAE" on page 40
- "SafeWord" on page 41
- "SecurID" on page 41
- "Unix" on page 41
- "Windows Desktop SSO" on page 41
- "Windows NT" on page 42
- "WSSAuth" on page 42

Active Directory

The Active Directory authentication module performs authentication in a manner similar to that of the LDAP module, but uses Microsoft's Active Directory^{TM} server. Although the LDAP authentication module can be configured for Active Directory, using this module allows you to have both LDAP and Active Directory authentication exist under the same realm. For information on the Active Directory authentication module attributes, see "Active Directory" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

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

Anonymous

The Anonymous authentication module allows a user to log in to OpenSSO Enterprise without specifying validating credentials. Anonymous connections are usually customized by the OpenSSO Enterprise administrator so that Anonymous users have limited access to the server (for example, access for read or access for search), or to specific trees or entries within the identity data store. A list of anonymous users can be defined by adding the applicable user identifier(s) to the module's Valid Anonymous Users attribute. Granting anonymous access means that it can be accessed without providing a password. A default anonymous user is created during OpenSSO Enterprise installation. For information on the Anonymous authentication module attributes, see "Anonymous" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

Certificate

The Certificate authentication module requires a personal digital certificate (PDC) to identify and authenticate a user. The module can be configured to require a match between the user's PDC and a PDC stored in the configuration data store, as well as verification against a Certificate Revocation List. It can also require the use of the Online Certificate Status Protocol (OCSP) to determine the state of a certificate. Successful or failed authentication is dependent on whether or not the certificate is valid. The following information should be considered before using the Certificate authentication module.

- The web container that is installed with the OpenSSO Enterprise needs to be secured and configured for Certificate-based Authentication.
- To add this module, log in to OpenSSO Enterprise as the realm Administrator and have OpenSSO Enterprise and the web container configured for SSL and with client authentication enabled.
- To check the certificates presented by a client against those found in a directory server, import the root CA certificate for the client certificate into the trust store of the JVM on the OpenSSO Enterprise host machine (by default *JDK_HOME*`/jre/lib/security/cacerts`). You can use the `keytool` command line interface.
- If you are configuring OpenSSO Enterprise Certificate authentication with an SSL-enabled Sun Java System Web Server 6.1 instance, and wish to have the Web Server defined to accept both certificate based and non certificate based authentication requests, you must set the following value in the Web Server's `obj.conf` file:

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

  This is due to a limitation in the Web Server console when setting the optional attribute for this behavior. Is this the web container on which OSSO is deployed?
- Before enabling the Certificate-based module, see Using Certificates and Keys in the *Sun ONE Web Server 6.1 Administration Guide* at **`http://docs.sun.com/db/prod/s1websrv#hic`** or the *Sun ONE Application Server Administrator's Guide to Security* at **`http://docs.sun.com/db/prod/s1appsrv#hic`** for initial configuration steps.
- Each user that will authenticate using the Certificate authentication module must request a PDC for the browser. Instructions depend upon the browser being used. See the specific documentation for more information.

For information on the Certificate authentication module attributes, see "Certificate" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

Data Store

The Data Store authentication module allows a user to authenticate against one or more of a realm's identity data stores (depending on the authentication process configuration). This authentication module is enabled after installation to authenticate to the identity data store configured for the top level realm during installation. Depending on the deployment, this might be the *embedded* configuration data store (for user storage in sample deployments) or a defined external data store (for user storage in real world deployments). If you change the identity data store definition for the top level realm, the Data Store authentication module adapts to the modified data store definition. After installation, additional identity data stores can be added to the top level or sub realms and configured for Data Store authentication. The Data Store authentication module extends the `com.sun.identity.idm.IdRepo` interface. You can include or remove a configured data store from identity authentication by unchecking User in the Identity Types That Can Be Authenticated attribute in the Data Store profile. For more information on the Data Store authentication module attributes, see "Data Store" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

Federation

The Federation authentication module is used by all federation protocols implemented by OpenSSO Enterprise (including SAML v1.x and SAML v2, Liberty ID-FF, and WS-Federation). This module can not be invoked as other authentication modules. (If you do this, you have to provide the necessary authentication callbacks as it will not initiate single sign-on and provide them itself.) The Federation authentication module requires a federation setup and will be invoked internally on the service provider side to create an `SSOToken` after validating the federation protocol messages. For information on the Federation authentication module attributes, see "Federation" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

HTTP Basic

The HTTP Basic authentication module uses basic authentication in the context of HTTP communication. A web browser issues a request for a username and password, and sends the credentials to the web server as part of the authentication request. OpenSSO Enterprise retrieves the username and password and authenticates the user using the LDAP authentication module. In order for HTTP Basic to function correctly, both the LDAP authentication module and the HTTP Basic authentication module must be added to the appropriate realm. Once the user successfully authenticates, the user will be able to authenticate again without being prompted for username and password. For information on the HTTP Basic authentication module attributes, see "HTTP Basic" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

JDBC

The Java Database Connectivity (JDBC) authentication module provides a mechanism to allow OpenSSO Enterprise to authenticate users through any SQL database that provides JDBC technology-enabled drivers. The connection to the SQL database can be either directly through a JDBC driver, or through a JNDI connection pool. For information on the HTTP Basic authentication module attributes, see "JDBC" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

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

LDAP

The LDAP authentication module uses a Distinguished Name (DN) and password to authenticate to an LDAP data store. If the submitted credentials are found in the directory, the user is authenticated and an `SSOToken` created. This module is enabled during OpenSSO Enterprise installation for the top level realm. For information on the LDAP authentication module attributes, see "LDAP" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

Membership

The Membership authentication module is implemented for personalized sites where a user is able to create an account and define preferences without the aid of an administrator. Once created, the user can access the resource as an added user. After the account is created, the user can authenticate to the appropriate OpenSSO Enterprise realm as configured. For information on the Membership authentication module attributes, see "Membership" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

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 compares it against the data store to find a user that matches the number. If found, the user is validated. For information on the MSISDN authentication module attributes, see "MSISDN" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

RADIUS

The RADIUS authentication module enables authentication to an installed and configured RADIUS server currently being used for authentication. For information on the RADIUS authentication module attributes, see "RADIUS" in *Sun OpenSSO Enterprise 8.0 Administration Reference*. See "Before You Begin" on page 55 for special pre-configuration instructions when using the RADIUS authentication module.

SAE

The Secure Attribute Exchange (also known as Virtual Federation) authentication module is used when an external entity (such as an existing application) has already authenticated the user and wishes to securely inform a local instance of OpenSSO Enterprise to trigger the creation of an `SSOToken` for the user. This module is also used when the existing entity instructs the local instance of OpenSSO Enterprise to use federation protocols to transfer authentication and attribute information to a partner application. This module can not be invoked as other authentication modules. It requires setting up parties for Secure Attribute Exchange and will be invoked internally. For information on the Secure Attribute Exchange authentication module attributes, see "SAE" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

SafeWord

The SafeWord authentication module handles authentication requests to Secure Computing's SafeWord™ or SafeWord PremierAccess™ authentication servers. The SafeWord server may exist on the system on which OpenSSO Enterprise is installed, or on a separate system. For information on the SafeWord authentication module attributes, see "SafeWord" in *Sun OpenSSO Enterprise 8.0 Administration Reference*. See "Before You Begin" on page 55 for special pre-configuration instructions when using the SafeWord authentication module.

SecurID

The SecurID authentication module handles authentication requests to RSA's ACE/Server authentication servers. OpenSSO Enterprise provides the client portion of SecurID authentication. The ACE/Server may exist on the system on which OpenSSO Enterprise is installed, or on a separate system. For information on the SecurID authentication module attributes, see "SecurID" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

Note – The SecurID Authentication module is available for the Solaris/SPARC, Solaris/x86, Linux, and Windows platforms.

Unix

The Unix authentication module is a Pluggable Authentication Module (PAM) that authenticates user identifiers known to the Solaris or Linux system (local or NIS) on which OpenSSO Enterprise is installed. This module makes use of an authentication helper daemon called `amunixd` which opens a socket on a specified port in order to listen for Unix authentication requests. This daemon process is separate from the authentication process. The PAM Service Name attribute defaults to `other` for Solaris, and `password` for Linux. For information on the Unix authentication module attributes, see "Unix" in *Sun OpenSSO Enterprise 8.0 Administration Reference*. For instructions on setting up and running `amunixd`, see "Running the Unix Authentication Helper (amunixd Daemon)" in *Sun OpenSSO Enterprise 8.0 Installation and Configuration Guide*.

Windows Desktop SSO

The Windows Desktop SSO authentication module is a Kerberos-based authentication plug-in module targeted for Windows™ desktop users. It allows a user who has already authenticated to a Kerberos Distribution Center (KDC) to authenticate to OpenSSO Enterprise without re-submitting login credentials (in effect, single sign-on). In order to perform Kerberos-based single sign-on to OpenSSO Enterprise, the user on the client side must support the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) protocol. (In general, any user that supports this protocol should be able to use this module.) The user presents the Kerberos token to OpenSSO Enterprise using SPNEGO and the client sends back a SPNEGO token embedded with a Kerberos token. The module retrieves the Kerberos token, authenticates the user using the Java GSS API and, if successful, returns an `SSOToken` to the client.

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.

For information on the Windows Desktop SSO authentication module attributes, see "Windows Desktop SSO" in *Sun OpenSSO Enterprise 8.0 Administration Reference*. See "Before You Begin" on page 55 for special pre-configuration instructions when using the Windows Desktop SSO authentication module.

Windows NT

The Windows NT authentication module handles authentication against an installed Windows NT or Windows 2000 server. For information on the Windows NT authentication module attributes, see "Windows NT" in *Sun OpenSSO Enterprise 8.0 Administration Reference*. See "Before You Begin" on page 55 for special pre-configuration instructions when using the Windows NT authentication module.

WSSAuth

The WSSAuth authentication module is used for authenticating the WS-Security Username Token profile when using the digest option. When configured as part of the authentication chain in the web services provider (wsp) agent profile, the user is authenticated using the user name and password (or password equivalent) presented via the Username Token profile. The communicating WSC and WSP must agree upon a common password policy in their back end.

Authentication Types

The *authentication type* specifies from where the authentication process will be initiated. The authentication type is specified by appending the appropriate parameter to a login URL, or through the authentication API. OpenSSO Enterprise supports the following authentication types:

- *Authentication level* allows authentication based on a security level defined for each authentication module by an administrator.
- *Module* allows authentication by specifying an authentication module instance.
- *Realm* allows authentication using the authentication process defined for a specific realm.

- *Service* allows authentication using the authentication process defined for a specific service or application.
- *User* allows a user to authenticate using the authentication process defined in the user's profile.

For each of these authentication types, the user can either pass or fail the defined authentication process (either a module or a chain). For more information, see "Initiating the Authentication Type" on page 64. For more information on specifying an authentication type, see "Accessing the Authentication Service User Interface with a Login URL" on page 80.

Authentication Post Processing

Post processing functionality can be added to the authentication process. This would define actions taken after a successful or failed authentication. For more information see "Adding Authentication Post Processing Features" in *Sun OpenSSO Enterprise 8.0 Developer's Guide*.

Configuring the Core Authentication Service

Core Authentication contains general Authentication Service properties that can be defined globally for the OpenSSO Enterprise deployment (under the Configuration tab) or specifically for a configured realm (under the Access Control tab). The Core authentication module is added and enabled for the top level realm during installation. As new realms are created under the top level realm, these properties (and the values defined globally for them) are dynamically added to each new realm. Once added, new values can be defined and configured 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. The Authentication Service finds the general operating data it needs in the following order of precedence.

1. Authentication process property values (specified authentication module instance, authentication chain or both) in the configured realm.
2. Realm property values defined for the realm's users, roles, services, and so forth.
3. Global property values if no overriding value is defined in the configured realm or authentication process.

The following procedures illustrate the levels at which the Core Authentication module can be modified after installation.

- "To Modify Core Authentication Properties Globally" on page 44
- "To Modify Core Authentication Properties By Realm" on page 45

▼ To Modify Core Authentication Properties Globally

Before You Begin This procedure assumes you are logged into the OpenSSO Enterprise console as the administrator.

1. **Click the Configuration tab.**

2. **Click Core under the Authentication tab.**

3. **Modify the Global attributes by adding or changing the values.**

   These properties contain operating values that are applied to the Authentication Service throughout the OpenSSO Enterprise deployment.

   Pluggable Authentication Module Classes
   : Specifies the Java classes of the available authentication modules. Takes a text string specifying the full class name (including package) of each authentication module. After writing a custom authentication module (by implementing the OpenSSO Enterprise `AMLoginModule` or the Java Authentication and Authorization Service [JAAS] `LoginModule` service provider interfaces), the new class value must be added to this property.

   Supported Authentication Modules for Clients
   : Specifies a list of authentication modules supported for a specific client. Formatted as:

     clientType `|` *module1*`,`*module2*`,`*module3*

     This attribute is read by the Client Detection Service when it is enabled.

   LDAP Connection Pool Size
   : Specifies the minimum and maximum connection pool to be used on a specific LDAP server and port. Formatted as:

     host`:`*port*`:`*min*`:`*max*

     This attribute is for LDAP and Membership authentication services only.

   Default LDAP Connection Pool Size
   : Sets the default minimum and maximum connection pool to be used with all LDAP authentication module configurations. Formatted as:

     min`:`*max*

     This value is superseded by a value defined for a specific host and port in the LDAP Connection Pool Size property.

   Remote Auth Security
   : Requires that OpenSSO Enterprise validate the identity of the calling application; thus all remote authentication requests require the calling application's `SSOToken`. This allows the Authentication Service to obtain the username and password associated with the application.

Keep Post Process Objects for Logout Processing
Requires that the user session hold the instances of any post processing authentication classes used during the log in process after authentication is complete. When user log out is later invoked, the `onLogout()` method of these instances is called. If this attribute is not enabled, the post processing instances are not preserved and new instances are created when logout is invoked.

Keep Authentication Module Objects for Logout Processing
Requires that the user session hold the instances of authentication modules used during the log in process after authentication is complete. When user log out is later invoked, the `destroyModuleState()` method of these instances is called. If this attribute is not enabled, the authentication module instances are not preserved and no method on the authentication modules is called upon log out.

4 **Modify the top level Realm attributes by adding or changing the values.**

These realm properties (as defined globally under the Configuration tab) are specific to the top level realm. Top level realm properties can also be modified by navigating to the top level realm itself. See "To Modify Core Authentication Properties By Realm" on page 45 for instructions and definitions of the attributes.

5 **Click Save.**

6 **Click Back to Service Configuration.**

7 **Logout of the OpenSSO Enterprise console.**

▼ To Modify Core Authentication Properties By Realm

Realm attributes are applied to the realm under which they are configured.

Before You Begin This procedure assumes you are logged into the OpenSSO Enterprise console as the administrator.

1 **Click the Access Control tab.**

2 **Click the name of the realm that contains the properties to be modified.**

3 **Click the Authentication tab.**

The Core authentication module's General properties for the realm are displayed.

4 **Modify the values of the realm's General properties.**

The General properties are defined in the Core authentication module and are configurable by realm. Those documented in this step are the ones most likely to be modified.

Default Authentication Chain	Defines the default authentication chain used by the realm's users. The authentication chain must first be created before it is displayed as an option in the drop down list. For more information see "To Create an Authentication Chain" on page 62.
Administrator Authentication Chain	Defines the authentication chain used by administrators when the process needs to be different from the authentication chain defined for end users. The authentication chain must first be created before it is displayed as an option in the drop down list. For more information see "To Create an Authentication Chain" on page 62.
Default Success Login URL	Specifies a URL that the user will be redirected to upon successful authentication to the realm.

5 **Click Advanced Properties.**

The Core authentication module's General Advanced Properties for the realm are displayed.

6 **Modify the attributes.**

The General Advanced Properties are defined in the Core authentication module and are configurable by realm. Those documented in this step are the ones less likely to be modified.

User Profile	This option determines the profile status of a successfully authenticated user.	
	Dynamic	Specifies that on successful authentication the Authentication Service will create a user profile if one does not already exist. The `SSOToken` will then be issued. The user profile is created in the realm's configured user data store.

	Dynamic With User Alias	Specifies that on successful authentication the Authentication Service will create a user profile that contains the User Alias List attribute which defines one or more aliases that for mapping a user's multiple profiles.
	Ignore	Specifies that a user profile is not required for the Authentication Service to issue an `SSOToken` after a successful authentication.
	Required	Specifies that on successful authentication the user must have a user profile in the realm's configured user data store in order for the Authentication Service to issue an `SSOToken`.
Administrator Authentication Configuration	Defines the authentication chain used by administrators when the process needs to be	

	different from the authentication chain defined for end users. The authentication chain must first be created before it is displayed as an option in the drop down list. For more information see "To Create an Authentication Chain" on page 62.
User Profile Dynamic Creation Default Roles	Specifies the DN of a role to be assigned to a new user whose profile is created when either of the Dynamic options is selected under the User Profile attribute. There are no default values. The role specified must be within the realm for which the authentication process is configured. **Tip –** This role can be either an OpenSSO Enterprise or LDAP role, but it cannot be a filtered role.
Persistent Cookie Mode	Determines whether users can return to their authenticated session after restarting the browser. When enabled, a user session will not expire until its *persistent* cookie expires (as specified by the value of the Persistent Cookie Maximum Time attribute), or the user explicitly logs out. By default, the Authentication Service uses only *memory* cookies (expires when the browser is closed). **Tip –** A persistent cookie must be explicitly requested by the client by appending the `iPSPCookie=yes` parameter to the login URL. For more information see "`iPSPCookie` Parameter" on page 86.
Persistent Cookie Maximum Time	Specifies the interval after which a persistent cookie expires. The interval begins when the user's session is successfully authenticated. The maximum value is 2147483647 (time in seconds). The field will accept any integer value less than the maximum.

Alias Search Attribute Name	After a user is successfully authenticated, the user's profile is retrieved. This field specifies a second LDAP attribute to use in a search for the profile if a search using the first LDAP attribute fails to locate a matching user profile. Primarily, this attribute will be used when the user identification returned from an authentication module is not the same as that specified in User Naming Attribute. For example, a RADIUS server might return `abc1234` but the user name is `abc`. There is no default value for this attribute. The field takes any valid LDAP attribute.
Default Authentication Locale	Specifies the default language subtype to be used by the Authentication Service. The default value is `en_US`. See "Supported Language Locales" in *Sun OpenSSO Enterprise 8.0 Administration Reference* for a list of supported language subtypes. To use a different locale, authentication templates for that locale must first be created. A new directory must then be created for these templates. For more information see "`locale` Parameter" on page 83.
Organization Authentication Configuration	Defines the default authentication chain used by the realm's users. The authentication chain must first be created before it is displayed as an option in this attribute's drop down list. For more information see "To Create an Authentication Chain" on page 62.
Account Lockout Attributes	These attributes are relevant to account lockout in which a user will be *locked out* from authenticating after a defined number of log in attempts has failed. For more information on the account lockout options, see "Enabling Account Lockout" on page 87.
	Login Failure Lockout Mode Selecting this attribute enables a *physical lockout*. Physical lockout will inactivate an LDAP attribute (defined in the Lockout Attribute Name property) in the user's profile.

	This attribute works in conjunction with several other lockout and notification attributes.
Login Failure Lockout Count	Defines the number of attempts that a user has to authenticate, within the time interval defined in Login Failure Lockout Interval, before being locked out.
Login Failure Lockout Interval	Defines (in minutes) the time in which failed login attempts are counted. If one failed login attempt is followed by a second failed attempt, within this defined lockout interval time, the lockout count is begun and the user will be locked out if the number of attempts reaches the number defined in Login Failure Lockout Count. If an attempt within the defined lockout interval time proves successful before the number of attempts reaches the number defined in Login Failure Lockout Count, the lockout count is reset.
Email Address to Send Lockout Notification	Specify one (or more) email address(es) to which notification will be sent if a user lockout occurs. If sending: ■ To multiple addresses, separate each address with a space. ■ To non-English locales, format the address as *email_address*|*locale*|*charset* where *locale* is the language locale and *charset* is the character set.
Warn User After N Failures	Specifies the number of authentication failures that can occur before OpenSSO Enterprise displays a warning message that the user will be locked out.
Login Failure Lockout Duration	Defines (in minutes) how long a user must wait after a lockout before attempting to authenticate again. Entering a value greater than 0, enables *memory lockout* and disables physical lockout. Memory lockout is when the user's account is locked in memory for the number of minutes specified. The account is unlocked after the time period has passed.

Lockout Duration Multiplier	Defines a value with which to multiply the value of the Login Failure Lockout Duration for each successive lockout. For example, if Login Failure Lockout Duration is set to 3 minutes, and the Lockout Duration Multiplier is set to 2, the user will be locked out of the account for 6 minutes. Once the 6 minutes has elapsed, if the user again provides the wrong credentials, the lockout duration would then be 12 minutes. With the Lockout Duration Multiplier, the lockout duration is incrementally increased based on the number of times the user has been locked out.
Lockout Attribute Name	Defines the LDAP attribute to be marked as `inactive` for physical lockout. The default value is `inetuserstatus` (although the field in the OpenSSO Enterprise console is empty). The Lockout Attribute Value field must also contain an appropriate value.
Lockout Attribute Value	Specifies the action to take on the attribute defined in Lockout Attribute Name. The default value is `inactive` (although the field in the OpenSSO Enterprise console is empty). The Lockout Attribute Name field must also contain an appropriate value.
Invalid Attempts Data Attribute Name	Defines the attribute to which information regarding failed authentication attempts will be stored when the Store Invalid Attempts in Data Store attribute is enabled. The value of this attribute is used if the OpenSSO Enterprise schema is not loaded. **Tip** – The specified attribute needs to be defined in the LDAP User Attributes property of the data store configuration if the data store type is either Active Directory, Generic LDAPv3 or Sun DS with OpenSSO schema.
Default Success Login URL	Accepts a list of values that specifies where users are directed after successful

	authentication. The format of this attribute is *client-type*|*URL* although the only value you can specify at this time is a URL which assumes the type HTML. The default value is `/opensso/console`. Values that don't specify HTTP or HTTP(s) will be appended to the deployment URI.
Default Failure Login URL	Accepts a list of values that specifies where users are directed after an attempted authentication has failed. The format of this attribute is *client-type*|*URL* although the only value you can specify at this time is a URL which assumes the type HTML. Values that don't specify HTTP or HTTP(s) will be appended to the deployment URI.
Authentication Post Processing Classes	Specifies one or more Java classes used to customize post authentication processes for successful or unsuccessful logins. The Java class must implement the `com.sun.identity.authentication.spi.AMPostAuthProc` OpenSSO Enterprise interface. Additionally, add a JAR containing the post processing class to the classpath of the web container instance on which OpenSSO Enterprise is configured. If the web container on which OpenSSO Enterprise is configured explodes the WAR follow this procedure. a. Stop the web container instance. b. Change to the `WEB-INF/lib` directory in the exploded OpenSSO Enterprise WAR directory. For example, if using Sun Application Server, *AS=Deploy=BaseAS=Domain-Dir*/*AS-Domain***/applica** c. Copy the JAR that contains the post processing class to the `lib` directory. d. Restart the web container instance.
Generate UserID Mode	When enabled, the Membership module will generate a list of alternate user identifiers if the one entered by a user during the

	self-registration process is not valid or already exists. The user identifiers are generated by the class specified in the Pluggable User Name Generator Class property.
Pluggable User Name Generator Class	Specifies the name of the class used to generate alternate user identifiers when Generate UserID Mode is enabled. The default value is `com.sun.identity.authentication.spi.DefaultUs`
Identity Types	Lists the type or types of identities for which OpenSSO Enterprise will search. Options include: ■ Agent ■ agentgroup ■ agentonly ■ Group ■ User
Pluggable User Status Event Classes	Specifies one or more Java classes used to provide a callback mechanism for user status changes during the authentication process. The Java class must implement the `com.sun.identity.authentication.spi.AMAuthCal` OpenSSO Enterprise interface. Account lockout and password changes are supported — the latter through the LDAP authentication module as the feature is only available for the module.
Store Invalid Attempts in Data Store	Enables the storage of information regarding failed authentication attempts to a user data store, allowing the information to be shared among multiple instances of OpenSSO Enterprise. (If this attribute is not enabled, the information would be local to the instance where the lockout occurred.) This function requires the use of a data store enabled with the OpenSSO Enterprise schema and its `sunAMAuthInvalidAttemptsData` attribute. You can remove the dependency on the OpenSSO Enterprise schema by defining a disparate attribute in which to store the

	information. To store data in an attribute not defined by the OpenSSO Enterprise schema, define a value for the Invalid Attempts Data Attribute Name attribute. This is enabled by default.
Module-based Authentication	Enables users to authenticate using module-based authentication. Otherwise, all attempts with the **`module=`***module_instance_name* login parameter will result in failure. See "Module Authentication" on page 75 for more information.
User Attribute Mapping to Session Attribute	Enables the authenticating user's identity attributes (stored in the identity repository) to be set as session properties in the user's `SSOToken`. The value takes the format *User-Profile-Attribute*`|`*Session-Attribute-Name*. If *Session-Attribute-Name* is not specified, the value of *User-Profile-Attribute* is used. All session attributes contain the `am.protected` prefix to ensure that they cannot be edited by the Client SDK.
Default Authentication Level	The authentication level value indicates how much to trust authentications. Once a user has authenticated, this value is stored in the user's `SSOToken`. When the `SSOToken` is presented to an application, the application can use the stored value to determine whether the level is sufficient to grant the user access. If the authentication level does not meet the minimum value required by the application, it can prompt the user to authenticate again in order to attain a higher authentication level. The authentication level should be set within a realm's specific authentication template. The Default Authentication Level value described here will apply only when no authentication level has been specified in the Authentication Level field for a specific realm's authentication template. The Default Authentication Level default value is 0. The value of this attribute is not used by OpenSSO

Enterprise but by any external application that may chose to use it. See "Authentication Level-based Authentication" on page 72 for more information.

7 **Click Save.**

8 **Click Back to Service Configuration.**

9 **Logout of the OpenSSO Enterprise console.**

Configuring the Authentication Process

As discussed in "Understanding the Authentication Service" on page 35 the authentication process itself is configured (simply) by instantiating the desired authentication module or (with more complexity) by creating an *authentication chain* of modules. The following sections contain the procedures for configuring an authentication process. (If you are configuring a process that includes the RADIUS, SafeWord, or Windows-based authentication modules, see "Before You Begin" on page 55.)

- "Before You Begin" on page 55
- "Configuring Authentication Modules" on page 60
- "Creating Authentication Chains" on page 62

Before You Begin

The procedures in this section are relevant when using the RADIUS, SafeWord, or the Windows-based authentication modules.

- "Setting Up for RADIUS and SafeWord Authentication" on page 55
- "Setting Up Windows Desktop SSO Authentication" on page 56
- "Installing a Samba Client for Windows NT Authentication" on page 60

Setting Up for RADIUS and SafeWord Authentication

"To Set Up RADIUS or SafeWord with Sun Java System Application Server" on page 56 should be performed before configuring an authentication process that uses the RADIUS or SafeWord authentication modules.

▼ To Set Up RADIUS or SafeWord with Sun Java System Application Server

A Java Platform, Enterprise Edition `SocketPermission` class represents access to a network using sockets; it consists of a host location and a set of actions specifying ways to connect to that host. When the SafeWord client forms a socket connection to its server, only the `connect` action of the `SocketPermission` object is allowed in the Application Server's `server.policy` file. In order for the SafeWord authentication module to work properly, permission needs to be granted to the `accept`, `listen`, and `resolve` actions manually.

1 **Open the `server.policy` file in a text editor.**

2 **Add an entry for the appropriate actions into the Application Server `server.policy` file.**

For example, **`permission java.net.SocketPermission "localhost:1024-", "accept,connect,listen";`**

If this 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. 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. This second example (**`permission java.net.SocketPermission "machine1.example.com:1645", "connect,accept";`**) allows the code to connect to, and accept connections on, port `1645` on `machine1.example.com`.

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

---

3 **Save the `server.policy` file.**

4 **Restart Application Server.**

See Also For more information on `SocketPermission`, see the Java Platform, Enterprise Edition API Specification

Setting Up Windows Desktop SSO Authentication

In addition to configuring the Windows Desktop SSO authentication module using the OpenSSO Enterprise console, a user must be created in the Windows 2000 Domain Controller and Internet Explorer must be configured.

- "To Create a User in the Windows Domain Controller" on page 57
- "To Reduce the Size of a Kerberos Ticket" on page 58
- "To Set Up Internet Explorer" on page 59

▼ To Create a User in the Windows Domain Controller

1 In the domain controller, create a user account for the OpenSSO Enterprise 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 OpenSSO Enterprise host name as the User ID (login name).

The OpenSSO Enterprise host name should not include the domain name.

2 Associate the user account with a service provider name.

3 Install the `ktpass` utilities to the `c:\program files\support` tools directory.

The `ktpass` utilities are not installed as part of the Windows 2000 server. You must install it from the installation CD.

4 Export the keytab files to the system in which OpenSSO Enterprise is installed by running 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
```

The `ktpass` command accepts the following parameters:

hostname. The host name (without the domain name) on which OpenSSO Enterprise runs.

domainname . The OpenSSO Enterprise domain name.

DCDOMAIN. The domain name of the domain controller. This may be different from the OpenSSO Enterprise 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 KTPASS Syntax.

5 **Restart the server.**

To Reduce the Size of a Kerberos Ticket

All Active Directory groups to which a user belongs are encoded within an issued Kerberos ticket, increasing the size of the HTTP header. Choose one of the following options to reduce the ticket's size.

1 **Increase the default maximum header size of the web container being used.**

For example, when using Glassfish replace:

```
<connection-pool max-pending-count="4096"
queue-size-in-bytes="4096" receive-buffer-size-in-bytes="4096"
send-buffer-size-in-bytes="8192"/>
```

with

```
<connection-pool max-pending-count="4096"
queue-size-in-bytes="65536" receive-buffer-size-in-bytes="65536"
send-buffer-size-in-bytes="65536"/>
```

2 **Disable the PAC for the OpenSSO service account**

This is the Microsoft extension to Kerberos that contains the Active Directory groups. See `http://support.microsoft.com/kb/832572`.

▼ 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 OpenSSO Enterprise is in the browser's internet zone and enable Native Windows Authentication.
- Internet Explorer (5.01 or later) running on Windows 2000 (or later) currently supports SPNEGO. 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.

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 OpenSSO Enterprise to the local zone (if it is not added already).**

Troubleshooting If you are using Microsoft Internet Explorer 6.x for Windows Desktop SSO authentication and the browser does not have access to a Kerberos/SPNEGO token for the user that matches the (KDC) realm defined in the Windows Desktop SSO module, the browser will behave incorrectly for other modules after Windows Desktop SSO authentication fails. The direct cause of the problem is that after Windows Desktop SSO fails, the browser becomes incapable of passing all authentication module callbacks to OpenSSO Enterprise until the browser is restarted. Therefore all the modules coming after Windows Desktop SSO will fail due to null user credentials. See `http://support.microsoft.com/default.aspx?scid=kb;en-us;308074` for related information.

Note – As of this release, this restriction has been fixed by Microsoft. For more information, see http://www.microsoft.com/technet/security/bulletin/ms06-042.mspx.

Installing a Samba Client for Windows NT Authentication

To activate the Windows NT authentication module, Samba Client 3.x must be downloaded and installed. Samba is a file and print server for blending Windows and UNIX machines without requiring a separate Windows NT/2003 Server. See "To Install the Samba Client" on page 60 for more information.

Tip – Red Hat Linux ships with a Samba client located in the `/usr/bin` directory.

▼ To Install the Samba Client

1 **Download the Samba software from** http://www.samba.org**.**

2 **Install the software to the** *OpenSSO-Deploy-base*/*OpenSSO-deploy-uri*/`bin` **directory.**

3 **(Optional) When running OpenSSO Enterprise on AIX, follow this sub procedure.**

See the Samba `README` for the reasons this startup script might be necessary for your deployment.

a. **Create a** `smbclient` **wrapper script.**

For example:

```
#!/bin/sh

unset LIBPATH
/usr/bin/smbclient $@
```

b. **Copy the wrapper script to the** *OpenSSO-Deploy-base*/*OpenSSO-deploy-uri*//`bin` **directory.**

4 **(Optional) Set multiple interfaces by configuration of the** `smb.conf` **file which is passed to the** `smbclient`**.**

Configuring Authentication Modules

Before defining an authentication process, instances of the authentication modules to be used in the process must be added in the appropriate realm. Before adding an authentication module instance to a realm, default global values can be defined which will then be carried over to any instance created from the module. Once the module instance is added to the realm, you can modify the values specifically for that realm. See "Authentication Modules" on page 36 for information about each authentication module.

- "To Define Global Values for an Authentication Module" on page 61
- "To Add an Authentication Module Instance to a Realm or Sub Realm" on page 61

▼ To Define Global Values for an Authentication Module

1 **Log in to the OpenSSO Enterprise console as the administrator.**

By default, `amadmin`.

2 **Click the Configuration tab.**

3 **Under Authentication, select the module you want to modify.**

4 **Modify the values for the appropriate properties and click Save.**

The default values defined globally will be assigned per realm or sub realm when the authentication module is added. See "To Add an Authentication Module Instance to a Realm or Sub Realm" on page 61.

▼ To Add an Authentication Module Instance to a Realm or Sub Realm

Authentication module instances can be expressly accessed for authentication (outside of an authentication chain) by using the module authentication type. See "Module Authentication" on page 75 for more information.

1 **Log in to the OpenSSO Enterprise console as the administrator.**

By default, `amadmin`.

2 **Click the Access Control tab.**

3 **Click the name of the realm under which you are adding an authentication module instance.**

4 **Select the Authentication tab.**

5 **Click New under Module Instances.**

6 **Enter a Name for the module instance.**

The name must be unique.

7 **Select the Type of authentication module.**

8 **Click OK.**

9 **Click the name of the new instance to edit the properties for this realm.**

See "Authentication" in *Sun OpenSSO Enterprise 8.0 Administration Reference*, or click Help for definitions of the properties.

10 **Click Save.**

Repeat these steps to add multiple authentication module instances.

Creating Authentication Chains

An *authentication chain* is a configured authentication process in which a user must pass credentials to all module instances defined in it. Authentication chains are used to define the authentication process for all authentication types except Module. See "Initiating the Authentication Type" on page 64 for more information.

Note – Authentication chaining is achieved using the Java Authentication and Authorization Service (JAAS) framework integrated with the Authentication Service.

▼ To Create an Authentication Chain

Before You Begin Instances of all authentication modules to be used in the authentication chain must first be added to the realm under which you are creating the chain. See "Configuring Authentication Modules" on page 60 for instructions.

1 **Log in to the OpenSSO Enterprise console as the administrator.**

By default, `amadmin`.

2 **Click the Access Control tab.**

3 **Click the name of the realm under which you are creating a new authentication chain.**

4 **Select the Authentication tab.**

5 **Click New under Authentication Chaining.**

6 **Enter a name for the authentication chain and click OK.**

The chain's properties page is displayed.

7 **Click Add to include the desired authentication module instance(s).**

A drop down list of authentication modules instantiated in the realm is displayed.

8 **Select the desired authentication module instance from the Instance list.**

9 **Select the appropriate criteria for the module instance from the Criteria list.**

These flags establish the enforcement criteria for the module instance within a chain. There is a hierarchy for enforcement: REQUIRED is the highest and OPTIONAL is the lowest. More information can be found in the javax.security.auth.login.Configuration class document.

REQUIRED	Successful authentication to this module instance is required for the authentication process to succeed. If authentication to any REQUIRED module instances defined in a chain fails, authentication will fail. The authentication process will continue through the authentication chain whether authentication to the REQUIRED module instance succeeds or fails.
REQUISITE	Successful authentication to this module instance is required to proceed through the authentication chain. If authentication is successful, the authentication process moves to the next module instance in the authentication chain. If authentication fails, the chain is broken, control returns to the Authentication Service, and the user is not authenticated.
SUFFICIENT	Successful authentication to this module is not required but, if authentication does succeed, the user is authenticated and the authentication process will not continue through the authentication chain. If authentication to a SUFFICIENT module instance fails, the authentication process continues through the module instances in the authentication chain.
OPTIONAL	Successful authentication to this module instance is not required but, whether it succeeds or fails, the authentication process continues through the module instances in the authentication chain.

10 **Enter options for the chain.**

This enables additional options to be defined for the module as a *key*=*value* pair. For example, if the authentication module supports debugging, enter `debug=true`. Multiple options are separated by a space. More information can be found in the javax.security.auth.login.Configuration class document.

11 **(Optional) Add values for the following attributes.**

Successful Login URL	Specify a URL that the user will be redirected to upon successful authentication.
Failed Login URL	Specify a URL that the user will be redirected to upon failed authentication.

Post Authentication Processing Class	Specify the name of a Java class used to customize any post authentication processes (successful or failed).

12 Click Save.

Initiating the Authentication Type

"Authentication Service User Interface" on page 36 and "Authentication Types" on page 42 give high level views of the types of authentication available with OpenSSO Enterprise and how the authentication type is initiated: by appending an appropriate parameter to the login URL (or programmatically using the authentication API). The base of the login URL is:

`http://`*OpenSSO-machine-name*`.`*domain*`:`*port*`/`*service_deploy_uri*`/UI/Login`

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

The following sections contain more information about the specific authentication types.

- "Realm Authentication" on page 64
- "Service Authentication" on page 67
- "User Authentication" on page 69
- "Authentication Level-based Authentication" on page 72
- "Module Authentication" on page 75
- "Role Authentication (Legacy Mode)" on page 77

More information on these authentication types and the URL parameters with which they work can be found in "Accessing the Authentication Service User Interface with a Login URL" on page 80. Information on initiating the authentication type using the programmatic interfaces is in the *Sun OpenSSO Enterprise 8.0 Developer's Guide*.

Realm Authentication

Realm authentication is the default authentication type for OpenSSO Enterprise. It allows a member of a realm to authenticate using the authentication process configured for that particular realm (or sub realm). The following sections contain more information.

- "Configuring Realm Authentication" on page 65
- "Initiating Realm Authentication with the Login URL" on page 65
- "Redirecting Users After Realm Authentication" on page 66

Configuring Realm Authentication

The authentication process for a realm is defined by selecting the appropriate authentication chain in the realm or sub realm's configuration.

▼ To Configure A Realms's Authentication Process

1 **Log in to the OpenSSO Enterprise console as the administrator.**

By default, `amadmin`.

2 **Click the Access Control tab.**

3 **Click the name of the realm under which you configuring an authentication process.**

4 **Click the Authentication tab.**

5 **Select the appropriate authentication chain as a value for the Default Authentication Chain attribute.**

See "Creating Authentication Chains" on page 62 for information.

6 **(Optional) Select the appropriate authentication chain as a value for the Administrator Authentication Chain attribute.**

This authentication chain is used if the authentication process for administrators needs to be different from the process for end users.

7 **Click Save.**

Initiating Realm Authentication with the Login URL

To initiate authentication for a member of a particular realm, append the **`domain=`***realm-name* parameter or the **`realm=`***realm-name* parameter to the base login URL as in:

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?realm=sun
```

Note – If there is no defined parameter, the realm will be determined from the server host and domain specified in the login URL. The base login URL will initiate authentication for the top level realm without the `realm` 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 Realm/DNS Alias Names attribute.

   After calling the correct realm, the authentication module(s) to which the user will authenticate are retrieved from the Default Authentication Chain attribute or the Administrator Authentication Chain attribute.

Caution – If `User1` is authenticated to `realmA` and then tries to access `realmB`, a warning page is displayed that asks the user to authenticate to `realmB` with the authentication process specified for `realmB`, or return to the existing authenticated session with `realmA`. If the user chooses to authenticate to `realmB`, only the values of the `realm` and `module` (if specified) parameters are passed and honored for determining the new authentication process.

Redirecting Users After Realm Authentication

Upon a successful or failed realm authentication, OpenSSO Enterprise 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 Authentication Redirection URL Precedence" on page 66
- "Failed Realm Authentication Redirection URL Precedence" on page 67

Successful Realm Authentication Redirection URL Precedence

The redirection URL for successful realm 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. The value of the Success URL attribute in the user's profile specific to the client type from which the request was received.
4. The value of the Success URL attribute in the role entry of the user's profile specific to the client type from which the request was received.
5. The value of the Default Success Login URL attribute in the realm to which the user is a member specific to the client type from which the request was received.
6. The value of the Default Success Login URL attribute in the top level realm specific to the client type from which the request was received.
7. The value of the Success URL attribute in the user's profile.
8. The value of the Success URL attribute in the role entry of the user's profile.
9. The value of the Success URL attribute in the realm to which the user is a member.
10. The value of the Default Success Login URL attribute in the top level realm.

Failed Realm Authentication Redirection URL Precedence

The redirection URL for failed realm 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. The value of the Failure URL attribute in the user's profile specific to the client type from which the request was received.
4. The value of the Failure URL attribute in the role entry of the user's profile specific to the client type from which the request was received.
5. The value of the Default Failure Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.
6. The value of the Default Failure Login URL attribute in the top level realm specific to the client type from which the request was received.
7. The value of the Failure URL attribute in the user's profile.
8. The value of the Failure URL attribute in the role entry of the user's profile.
9. The value of the Default Failure Login URL attribute in the realm entry of the user's profile.
10. The value of the Default Failure Login URL attribute in the top level realm.

Service Authentication

Service authentication allows a user to authenticate to a specified authentication chain configured in a realm or sub realm. For authentication to be successful, the user must authenticate to each module defined in the chain. The following sections contain more information.

- "Configuring Service Authentication" on page 67
- "Initiating Service Authentication with the Login URL" on page 68
- "Redirecting Users After Service Authentication" on page 68

Configuring Service Authentication

To authenticate using service authentication, simply create an authentication chain in the appropriate realm.

1. Add authentication module instances to the realm. (See "To Add an Authentication Module Instance to a Realm or Sub Realm" on page 61.)
2. Create an authentication chain in the realm. (See "Creating Authentication Chains" on page 62.)

3. Create a login URL. (See "Initiating Service Authentication with the Login URL" on page 68.)

Initiating Service Authentication with the Login URL

To initiate the authentication process defined for a particular service, append the **service=***auth-chain-name* parameter to the base login URL as in:

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?service=bankauth
```

Additionally, you can append the **realm=***realm-name* parameter to the base login URL as in:

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login
?realm=bankrealm?service=bankauth
```

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

Redirecting Users After Service Authentication

Upon a successful or failed service authentication, OpenSSO Enterprise 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 Authentication Redirection URL Precedence" on page 68
- "Failed Service Authentication Redirection URL Precedence" on page 69

Successful Service Authentication Redirection URL Precedence

The redirection URL for successful service 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. The value of the Success URL attribute in the user's profile specific to the client type from which the request was received.
4. The value of the Success URL attribute in the service to which the user is authenticated specific to the client type from which the request was received.
5. The value of the Success URL attribute in the role entry of the user's profile specific to the client type from which the request was received.
6. The value of the Default Success Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.

7. The value of the Default Success Login URL attribute of the top level realm specific to the client type from which the request was received.
8. The value of the Success URL attribute in the user's profile.
9. The value of the Success URL attribute in the service to which the user is authenticated.
10. The value of the Success URL attribute in the role entry of the user's profile.
11. The value of the Default Success Login URL attribute in the realm entry of the user's profile.
12. The value of the Default Success Login URL attribute of the top level realm.

Failed Service Authentication Redirection URL Precedence

The redirection URL for failed service 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. The value of the Failure URL attribute in the user's profile specific to the client type from which the request was received.
4. The value of the Failure URL attribute of the service to which the user has authenticated specific to the client type from which the request was received.
5. The value of the Failure URL attribute in the role entry of the user's profile specific to the client type from which the request was received.
6. The value of the Default Failure Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.
7. The value of the Default Failure Login URL attribute in the top level realm specific to the client type from which the request was received.
8. The value of the Failure URL attribute in the user's profile.
9. The value of the Failure URL attribute of the service to which the user has authenticated.
10. The value of the Failure URL attribute in the role entry of the user's profile.
11. The value of the Default Failure Login URL attribute in the realm entry of the user's profile
12. The value of the Default Failure Login URL attribute in the top level realm.

User Authentication

User authentication allows a user to authenticate using an authentication chain specifically defined 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 in the chain. The following sections contain more information.

- "Configuring User Authentication" on page 70
- "Initiating User Authentication with the Login URL" on page 70
- "Redirecting Users After User Authentication" on page 71

Configuring User Authentication

To authenticate using user authentication, simply create an authentication chain in the appropriate realm and select it in the user's profile.

1. Add authentication module instances to the realm. (See "To Add an Authentication Module Instance to a Realm or Sub Realm" on page 61.)
2. Create an authentication chain in the realm. (See "Creating Authentication Chains" on page 62.)
3. Select the authentication chain as the value for the User Authentication Configuration attribute in the user's profile. (See "To Configure A User Authentication Process" on page 70.)
4. Create a login URL. (See "Initiating Service Authentication with the Login URL" on page 68.)

▼ To Configure A User Authentication Process

1 **Log in to the OpenSSO Enterprise console as the administrator.**

By default, `amadmin`.

2 **Click the Access Control tab.**

3 **Click the name of the realm that contains the user for whom you are configuring an authentication process.**

4 **Click the Subjects tab.**

5 **Under the User tab, click the user's Name.**

6 **Select the appropriate authentication chain as a value for the User Authentication Configuration attribute.**

7 **Click Save.**

Initiating User Authentication with the Login URL

To initiate the authentication process defined for a particular user, append the **`user=`***Universal-ID* parameter to the base login URL as in:

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?user=awhite
```

Additionally, you can append the **`realm=`***realm-name* parameter to the base login URL as in:

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login
?realm=bankrealm?user=awhite
```

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

Tip – The User Alias List attribute in the User profile is where the disparate Universal IDs defined for one user are mapped. On receiving a request for user authentication, the Authentication Service first verifies that the Universal ID passed with the login URL maps to a valid user. It then retrieves the specified Authentication Configuration data from the user's profile. In the case, for example, where there is more than one module in the authentication chain and a different Universal ID is defined for the user, all user profiles must map to the Universal ID specified in the URL or the user will be denied a validated `SSOToken`. An exception would be if one of the Universal IDs belongs to a top level administrator whereby the user mapping validation is not done and the user is given top level administrator rights.

Redirecting Users After User Authentication

Upon a successful or failed user authentication, OpenSSO Enterprise 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 Authentication Redirection URL Precedence" on page 71
- "Failed User Authentication Redirection URL Precedence" on page 72

Successful User Authentication Redirection URL Precedence

The redirection URL for successful user 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. The value of the Success URL attribute in the user's profile specific to the client type from which the request was received.
4. The value of the Success URL attribute in the role entry of the user's profile specific to the client type from which the request was received.
5. The value of the Default Success Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.

6. The value of the Default Success Login URL attribute in the top level realm specific to the client type from which the request was received.
7. The value of the Success URL attribute in the user's profile.
8. The value of the Success URL attribute in the role entry of the user's profile.
9. The value of the Default Success Login URL attribute in the realm entry of the user's profile.
10. The value of the Default Success Login URL attribute in the top-level realm.

Failed User Authentication Redirection URL Precedence

The redirection URL for failed user 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. The value of the Failure URL attribute in the user's profile specific to the client type from which the request was received.
4. The value of the Failure URL attribute in the role entry of the user's profile specific to the client type from which the request was received.
5. The value of the Default Failure Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.
6. The value of the Default Failure Login URL attribute in the top level realm specific to the client type from which the request was received.
7. The value of the Failure URL attribute in the user's profile.
8. The value of the Failure URL attribute in the role entry of the user's profile.
9. The value of the Failure URL attribute in the realm entry of the user's profile.
10. The value of the Default Failure Login URL attribute in the top-level realm.

Authentication Level-based Authentication

Authentication Level—based authentication allows an administrator to specify the security level of the authentication modules used in a particular authentication process. Each authentication module can be assigned an *authentication level* — an integer defined as the value of the module's Authentication Level attribute. A user that has successfully authenticated to an authentication module with a higher authentication level is deemed to have a higher level of trust. If successfully authenticate, the authentication level of the module will be set in the user's `SSOToken`. (If the user has successfully authenticated to multiple authentication modules, the highest authentication level will be set in the user's `SSOToken`.) Now when the user attempts to access a service which demands authentication trust at a particular level, the service can use the authentication level to determine if the user is meets the criteria. If not, the user is redirected to authenticate to an authentication module with the appropriate authentication level. The following sections contain more information.

- "Configuring Authentication Levels" on page 73
- "Initiating Authentication Level-based Authentication with the Login URL" on page 73
- "Redirecting Users After Authentication Level-based Authentication" on page 74

Configuring Authentication Levels

To set an authentication level for an authentication module, simply define an integer in the Authentication Level attribute of the desired authentication module.

- To set a global authentication level for a module, see "To Define Global Values for an Authentication Module" on page 61.
- To set the authentication level for an authentication module instance within a realm, see "To Add an Authentication Module Instance to a Realm or Sub Realm" on page 61.

Initiating Authentication Level-based Authentication with the Login URL

When Authentication Level-based authentication is initiated, the Authentication Service displays a 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's parameter. Users can select a module from the presented list. Once the user selects a module, the remaining process is based on Module Authentication. (See "Module Authentication" on page 75.)

To initiate Authentication Level-based authentication, append the **authlevel=***auth-level-value* parameter to the base login URL as in:

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?authlevel=8
```

Additionally, you can append the **realm=***realm-name* parameter to the base login URL as in:

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login
?realm=bankrealm?authlevel=8
```

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

All modules whose authentication level is larger or equal to *auth-level-value* will be displayed in an authentication menu. After the authentication menu with the relevant list of modules is displayed, the user must choose one with which to authenticate. If only one matching module is found, then the login page for that authentication module will be directly displayed.

Redirecting Users After Authentication Level-based Authentication

Upon a successful or failed authentication level-based authentication, OpenSSO Enterprise 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 URL Precedence" on page 74
- "Failed Authentication Level-based Authentication Redirection URL Precedence" on page 74

Successful Authentication Level-based Authentication Redirection URL Precedence

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. The value of the Success URL attribute in the user's profile specific to the client type from which the request was received.
4. The value of the Success URL attribute in the role entry of the user's profile specific to the client type from which the request was received.
5. The value of the Default Success Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.
6. The value of the Default Success Login URL attribute in the top level realm specific to the client type from which the request was received.
7. The value of the Success URL attribute in the user's profile.
8. The value of the Success URL attribute in the role entry of the user's profile.
9. The value of the Default Success Login URL attribute in the realm entry of the user's profile.
10. The value of the Default Success Login URL attribute in the top level realm.

Failed Authentication Level-based Authentication Redirection URL Precedence

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. The value of the Failure URL attribute in the user's profile specific to the client type from which the request was received.

4. The value of the Failure URL attribute in the role entry of the user's profile specific to the client type from which the request was received.
5. The value of the Default Failure Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.
6. The value of the Default Failure Login URL attribute in the top level realm specific to the client type from which the request was received.
7. The value of the Failure URL attribute in the user's profile.
8. The value of the Failure URL attribute in the role entry of the user's profile.
9. The value of the Default Failure Login URL attribute in the realm entry of the user's profile.
10. The value of the Default Failure Login URL attribute in the top level realm.

Module Authentication

Module authentication allows a user to specify the authentication module with which they will authenticate. The specified module must be added as a module instance in the realm or sub realm that the user is accessing. On receiving a request for module authentication, the Authentication Service verifies that the module is correctly configured as noted; if the module is not defined, the user is denied access. The following sections contain more information.

- "Configuring Module Authentication" on page 75
- "Initiating Module Authentication with the Login URL" on page 75
- "Redirecting Users After Module Authentication" on page 76

Configuring Module Authentication

To use module authentication, simply create an instance of the authentication module in the appropriate realm. See "To Add an Authentication Module Instance to a Realm or Sub Realm" on page 61.

Initiating Module Authentication with the Login URL

To initiate the authentication using a particular authentication module, append the **module=***auth-module-name* parameter to the base login URL as in:

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?module=DataStore
```

Additionally, you can append the **realm=***realm-name* parameter to the base login URL as in:

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login
?realm=bankrealm?module=LDAP
```

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

Redirecting Users After Module Authentication

Upon a successful or failed module authentication, OpenSSO Enterprise 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 Authentication Redirection URL Precedence" on page 76
- "Failed Module Authentication Redirection URL Precedence" on page 76

Successful Module Authentication Redirection URL Precedence

The redirection URL for successful module 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. The value of the Success URL attribute in the user's profile specific to the client type from which the request was received.
4. The value of the Success URL attribute in the role entry of the user's profile specific to the client type from which the request was received.
5. The value of the Default Success Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.
6. The value of the Default Success Login URL attribute in the top level realm specific to the client type from which the request was received.
7. The value of the Success URL attribute in the user's profile.
8. The value of the Success URL attribute in the role entry of the user's profile.
9. The value of the Default Success Login URL attribute in the realm entry of the user's profile.
10. The value of the Default Success Login URL attribute in the top level realm.

Failed Module Authentication Redirection URL Precedence

The redirection URL for failed module 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. The value of the Failure URL attribute in the user's profile specific to the client type from which the request was received.

4. The value of the Failure URL attribute in the role entry of the user's profile specific to the client type from which the request was received.
5. The value of the Default Failure Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.
6. The value of the Default Failure Login URL attribute in the top level realm specific to the client type from which the request was received.
7. The value of the Failure URL attribute in the user's profile.
8. The value of the Failure URL attribute in the role entry of the user's profile.
9. The value of the Default Failure Login URL attribute in the realm entry of the user's profile.
10. The value of the Default Failure Login URL attribute in the top level realm.

Role Authentication (Legacy Mode)

Role authentication allows a user to authenticate as a member of a specified role (either static or filtered) configured within a realm or sub realm. Role authentication is only available when the Access Manager SDK (AMSDK) Identity Repository Plug-in is enabled. See Chapter 15, "Enabling the Access Manager SDK (AMSDK) Identity Repository Plug-in," in *Sun OpenSSO Enterprise 8.0 Installation and Configuration Guide* for requirements and procedures to enable this legacy feature.

For role authentication to be initiated, the user must belong to the role and authenticate to each module defined in the authentication chain specified for that role. The following sections contain more information.

- "Configuring Role Authentication" on page 77
- "Initiating Role Authentication with the Login URL" on page 78
- "Redirecting Users After Role Authentication" on page 78

Configuring Role Authentication

The authentication method for a role is set by adding the legacy Authentication Configuration Service to the role and choosing the appropriate authentication chain from the displayed choices.

▼ To Configure An Authentication Process for a Role

1 Log in to the OpenSSO Enterprise console as the administrator.

By default, `amadmin`.

2 Click the Access Control tab.

3 **Click the name of the realm that contains the role for which you are configuring an authentication process.**

4 **Click the Subjects tab.**

5 **Click the Roles tab.**

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

7 **Click the Services tab.**

8 **Click Add.**

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

10 **Select the appropriate authentication chain from those displayed.**

See "Creating Authentication Chains" on page 62.

11 **Click Finish.**

Initiating Role Authentication with the Login URL

To initiate the authentication process defined for a particular role, append the **`role=`***role-name* parameter to the base login URL as in:

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?role=manager
```

A user who is not a member of the specified role will receive an error message when they attempt to authenticate using this parameter.

Redirecting Users After Role Authentication

Upon a successful or failed role authentication, OpenSSO Enterprise 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 Authentication Redirection URL Precedence" on page 76
- "Failed Module Authentication Redirection URL Precedence" on page 76

Successful Role Authentication Redirection URL Precedence

The redirection URL for successful role 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. The value of the Success URL attribute in the user's profile specific to the client type from which the request was received.
4. The value of the Success URL attribute in the role entry of the user's profile specific to the client type from which the request was received.
5. The value of the Success URL attribute in another role entry of the user's profile specific to the client type from which the request was received. (This option is a fallback if the previous redirection URL fails.)
6. The value of the Default Success Login URL attribute in the realm to which the user is a member specific to the client type from which the request was received.
7. The value of the Default Success Login URL attribute in the top level realm specific to the client type from which the request was received.
8. The value of the Success URL attribute in the user's profile.
9. The value of the Success URL attribute in the role entry of the user's profile.
10. The value of the Success URL attribute in another role entry of the user's profile. (This option is a fallback if the previous redirection URL fails.)
11. The value of the Default Success Login URL attribute in the realm to which the user is a member.
12. The value of the Default Success Login URL attribute in the top level realm.

Failed Role Authentication Redirection URL Precedence

The redirection URL for failed role 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. The value of the Failure URL attribute in the user's profile specific to the client type from which the request was received.
4. The value of the Failure URL attribute in the role entry of the user's profile specific to the client type from which the request was received.
5. The value of the Default Failure Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.
6. The value of the Default Failure Login URL attribute in the top level realm specific to the client type from which the request was received.
7. The value of the Failure URL attribute in the user's profile.
8. The value of the Failure URL attribute in the role entry of the user's profile.
9. The value of the Default Failure Login URL attribute in the realm entry of the user's profile.

10. The value of the Default Failure Login URL attribute in the top level realm.
11. The value of the Success URL attribute in the role entry of the user's profile.
12. The value of the Success URL attribute in another role entry of the user's profile. (This option is a fallback if the previous redirection URL fails.)

Accessing the Authentication Service User Interface with a Login URL

The Authentication Service user interface is accessed by entering a login URL into the Location Bar of a web browser. "Authentication Service User Interface" on page 36 and "Authentication Types" on page 42 give views of this login URL and how the authentication type is initiated by appending the appropriate parameter to the login URL. A parameter is a name/value pair appended to the end of a URL. The parameter starts with a question mark (`?`) which is followed by the form *name*`=`*value* (and an ampersand for multiple parameters). The format of the login URL with parameter(s) is:

```
http://OpenSSO-machine-name.domain:port/service_deploy_uri/UI/Login?parameter1=value1&parameter2=value2&parameterN=valueN
```

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

In addition to the parameters documented in "Authentication Types" on page 42, there are others that can be appended to the login URL. If more than one parameter exists, they 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 login URL, achieve various authentication functionality.

- "`goto` Parameter" on page 81
- "`gotoOnFail` Parameter" on page 81

- "`realm` Parameter" on page 82
- "`user` Parameter" on page 82
- "`locale` Parameter" on page 83
- "`module` Parameter" on page 83
- "`service` Parameter" on page 84
- "`arg` Parameter" on page 84
- "`authlevel` Parameter" on page 84
- "`forceAuth` Parameter" on page 85
- "`IDToken`*N* Parameters" on page 85
- "`iPSPCookie` Parameter" on page 86
- "`PersistAMCookie` Parameter" on page 86
- "`role` Parameter (Legacy Mode)" on page 86
- "`org` Parameter (Legacy Mode)" on page 87
- "`domain` Parameter (Legacy Mode)" on page 87

Tip – 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 defines a URL to which the user will be redirected after successfully authenticating.

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?
goto=A http://www.sun.com/homepage.html
```

A **`goto=`***logout-URL* parameter can also be set to link to a specified URL when the user logs out.

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?
goto=A http://www.sun.com/logout.html
```

There is an order of precedence in which OpenSSO Enterprise looks for redirection URLs. The order of preference is based on the type of authentication initiated. See "Initiating the Authentication Type" on page 64 for the order specific to each authentication type.

`gotoOnFail` Parameter

A **`gotoOnFail=`***failed-authentication-URL* parameter defines a URL to which the user will be redirected after failing the defined authentication process.

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?
goto=A http://www.sun.com/auth_fail.html
```

There is an order of precedence in which OpenSSO Enterprise looks for redirection URLs. The order of preference is based on the type of authentication initiated. See "Initiating the Authentication Type" on page 64 for the order specific to each authentication type.

`realm` Parameter

The **`realm=`***realm-name* parameter allows a member of a realm to authenticate using the authentication process configured for that particular realm (or sub realm). A user who is not already a member of the realm will receive an error message when they attempt to authenticate using the `realm` parameter. Realm authentication is the default authentication type for OpenSSO Enterprise.

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?realm=sun
```

A user profile can be dynamically created in the realm's configured user data store if all of the following are TRUE:

- The User Profile attribute in the Core Authentication module must be set to `Dynamic` or `Dynamic with User Alias`. See "To Modify Core Authentication Properties By Realm" on page 45.
- The user must successfully authenticate to the required module or authentication chain.
- The user does not already have a profile in the user data store.

If there is a value for this parameter, the correct login page (based on the realm name and locale setting) will be displayed. If this parameter is not set, the login page for the default top level realm is displayed. For more information, see "Realm Authentication" on page 64.

`user` Parameter

The **`user=`***Universal-ID* parameter forces authentication based on the authentication chain configured as the value of the User Authentication Configuration attribute in the user's profile. Using this parameter sends the user to a specific authentication process rather than the process configured for the user's organization.

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?user=jsmith
```

For more information, see "User Authentication" on page 69.

`locale` Parameter

OpenSSO Enterprise has the capability to display screens that are translated into languages other than English. These *localized* screens can be configured for the authentication process as well as for the console itself. The **`locale=`***language-locale* parameter allows the specified locale to take precedence over any other defined locales for the authentication process.

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?locale=ja
```

The login locale is displayed by the client after searching for the configured locale in the following places, order-specific:

1. Value of the `locale` parameter in login URL

   The value of the **`locale=`***language-locale* parameter takes precedence over all other defined locales. See "Supported Language Locales" in *Sun OpenSSO Enterprise 8.0 Administration Reference* for a list of supported language subtypes.
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's profile.
3. Locale defined in the HTTP header

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

   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.
6. Operating system locale

The locale derived from this pecking order is stored in the user's `SSOToken` and OpenSSO Enterprise 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 more information, see Localizing the Sun OpenSSO Enterprise 8.0 Login Page.

`module` Parameter

The **`module=`***module-name* parameter allows authentication using the specified authentication module. Any authentication module can be specified although it must first be registered and configured under the realm to which the user belongs.

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?module=Unix
```

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

For more information, see "Module Authentication" on page 75.

`service` Parameter

The **`service=`***authentication-chain-name* parameter allows a user to authenticate using a specific authentication chain. For authentication to be successful, the user must authenticate to each authentication module defined in the chain.

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?service=chain1
```

For more information, see "Service Authentication" on page 67.

`arg` Parameter

The **`arg=newsession`** parameter is used to end a user's current session and begin a new one. (The parameter is appended as is; there is no variable.) The Authentication Service will destroy a user's existing session token and perform a new login in one request. This option is typically used by the Anonymous authentication module. The user first authenticates with an anonymous session, and then clicks a register or login link.

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?arg=newsession
```

`authlevel` Parameter

An **`authlevel=`***integer* parameter tells the Authentication Service to call a module with an authentication level equal to or greater than the specified authentication level integer. The Authentication Level value is set in each authentication module's profile whether defined globally or per realm.

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?authlevel=3
```

When Authentication Level-based authentication is initiated, the Authentication Service displays a login page with a menu containing the authentication modules that have authentication levels equal to or greater then the value specified in the `authlevel` parameter. Users can select a module from the presented list. For more information, see "Authentication Level-based Authentication" on page 72.

`forceAuth` Parameter

The **`forceAuth=true`** query parameter forces the user to authenticate - even if the user currently has a valid session. (**`forceAuth=false`** is the default but is not explicitly appended to the URL.) **`forceAuth`** is useful in the following cases:

- If a user is authenticated with **`http://`***OpenSSO-machine-name*`.`*domain*`:`*port***`/opensso/UI/Login?module=LDAP`** and accesses the URL again, there would be no prompt for authentication. However, if the user is authenticated with **`http://`***OpenSSO-machine-name*`.`*domain*`:`*port***`/opensso/UI/Login?module=LDAP`** and accesses **`http://`***OpenSSO-machine-name*`.`*domain*`:`*port***`/opensso/UI/Login?module=LDAP&forceAuth=tru`** the user is prompted to authenticate again. If authentication is successful, the existing session token is updated accordingly. If authentication fails, the existing session token is still valid but it is not updated.
- If a user is authenticated with **`http://`***OpenSSO-machine-name*`.`*domain*`:`*port***`/opensso/UI/Login?module=LDAP`** and accesses **`http://`***OpenSSO-machine-name*`.`*domain*`:`*port***`/opensso/UI/Login?module=DataStore`** (**`forceAuth=true`** is not appended to the URL), the user is prompted to authenticate again using the Data Store authentication module. After successfully authenticating to the second module, OpenSSO Enterprise creates a new session token, and copies the properties from the old session before destroying it. However, if the user is authenticated with **`http://`***OpenSSO-machine-name*`.`*domain*`:`*port***`/opensso/UI/Login?module=LDAP`** and accesses **`http://`***OpenSSO-machine-name*`.`*domain*`:`*port***`/opensso/UI/Login?module=DataStore&forceAut`** (**`forceAuth=true`** is appended to the URL), the user is prompted to authenticate again but, after successfully authenticating using Data Store, the existing session token is updated.

See "Upgrading Sessions" on page 91 for more information.

`IDToken`*N* Parameters

The **`IDToken`***N***`=`***credential* parameter enables a user to pass authentication credentials using the login URL, allowing authentication without accessing the Authentication Service User Interface. This *zero page login* process works only for authentication modules with one login page. The values of **`IDToken1=`***credential***`&IDToken2=`***credential***`&IDToken`***N***`=`***credential* map to the fields on the authentication module's login page. For example, the LDAP authentication module might use `IDToken1` for the user identifier and `IDToken2` for the password. In this example, the URL would be **`http://`***OpenSSO-machine-name*`.`*domain*`:`*port***`/opensso/UI/Login?module=LDAP&IDToken1=awhite&`** (**`module=LDAP`** may be omitted if LDAP is the default authentication module.) The Anonymous

authentication module the URL would be
http://*OpenSSO-machine-name***.***domain***:***port***/opensso/UI/Login?module=Anonymous&IDToken1=anonymo**
as *anonymous* is a default OpenSSO Enterprise anonymous user.

`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. If the user is successfully authenticated and the browser is closed, the user can login with a new browser session and will be directed to the console without having to authenticate again. For example:

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?realm=hr&iPSPCookie=yes
```

To use this parameter, the Persistent Cookie Mode attribute must be enabled in the realm to which the user is logging in. The process will work until the value of the Persistent Cookie Maximum Time attribute elapses. For more information on these attributes, see "Configuring the Core Authentication Service" on page 43.

`PersistAMCookie` Parameter

The `PersistAMCookie` parameter will save the OpenSSO Enterprise cookie to memory, allowing an application (other than the browser) on the same machine to read it and create an `SSOToken`.

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?realm=people&iPersistAMCookiee=yes
```

`role` Parameter (Legacy Mode)

A **`role=`***role-name* 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.

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?role=manager
```

For more information, see "Role Authentication (Legacy Mode)" on page 77.

`org` Parameter (Legacy Mode)

The **`org=`***organization-name* parameter allows a member of the specified organization to authenticate using the authentication process configured for that particular organization. This is a legacy parameter for use with legacy directory information trees (DITs).

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?org=sun
```

The parameter would work much the same as the realm parameter. See "`realm` Parameter" on page 82 for more information.

`domain` Parameter (Legacy Mode)

The **`domain=`***domain* parameter allows a user to login to a realm identified as the specified domain. The specified domain must be set as a value in the Realm/DNS Aliases attribute of the realm's General profile.

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?domain=sun.com
```

The parameter would work much the same as the realm parameter. See "`realm` Parameter" on page 82 for more information.

Customizing Authentication

The following sections contain information on several features of the Authentication Service and the procedures used to enable them.

- "Enabling Account Lockout" on page 87
- "Authentication Service Failover" on page 89
- "Mapping Fully Qualified Domain Names" on page 89
- "Using Persistent Cookies" on page 90
- "Upgrading Sessions" on page 91
- "Sharing User Credentials Among Authentication Modules (Shared State)" on page 92
- "Redirecting Users After Authentication" on page 93
- "Configuring Multiple LDAP Authentication Modules (Legacy Mode)" on page 93

Enabling Account Lockout

The Authentication Service provides a feature where a user will be *locked out* from authenticating after a defined number of log in attempts has failed. By default, account lockout is disabled. When enabled, email notifications are sent to administrators regarding any account lockouts as well as recorded by the Logging Service. Only authentication modules that throw an Invalid Password Exception can leverage the Account Locking feature. These include Active Directory, Data Store, HTTP Basic, JDBC, LDAP, RADIUS, SafeWord, SecurID, and Unix. OpenSSO Enterprise supports two types of account lockout.

Physical Lockout This is the default lockout behavior, initiated by changing the status of a specified LDAP attribute in the user's profile to `inactive`. (The specified LDAP attribute is defined as the value of the Lockout Attribute Name attribute in the Core authentication module.) When physical account lockout is enabled an attribute in the user data store is used to hold information regarding the authentication attempts. This information includes:

- Invalid attempts count
- Last failed time
- Lockout time
- Lockout duration

Note – An *aliased* user is one that is mapped to an existing LDAP user profile through configuration of the User Alias List Attribute in the user profile. 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 Lockout Memory lockout is enabled by changing the value of the Login Failure Lockout Duration attribute (which defines how long a user must wait after a lockout before attempting to authenticate again) to a value greater then 0. The user's account is then locked in memory for the number of minutes specified. The account is unlocked after the time period has passed. There are special considerations when using the memory locking feature.

- If OpenSSO Enterprise is restarted, all accounts locked in memory are unlocked.
- If a user's account is locked in memory and the administrator resets the account lockout mechanism to physical lockout (by changing the value of the Login Failure Lockout Duration to 0), the user's account will be unlocked in memory and the lock count reset.
- During a memory lockout, if the user attempts to authenticate with the correct password (using authentication modules other than LDAP and Membership), a *User does not have profile in this realm error.* message is returned rather than a *User is not active.* error.

- If the Failure URL attribute in the user's profile is defined, neither the lockout warning message nor the message indicating that the account has been locked will be displayed; the user will be redirected to the defined URL.

For information on the account lockout attributes, see "Configuring the Core Authentication Service" on page 43.

Authentication Service Failover

The Authentication Service will automatically redirect an authentication request to a second server if the primary server becomes unavailable because of a hardware or software problem, or if the server is temporarily shut down. The Authentication Service URL is first passed to an instance of the `com.sun.identity.authentication.AuthContext` class. If this URL is unavailable, locations defined in the Servers attribute of the primary OpenSSO Enterprise instance's configuration will be checked and, if one is available, an authentication context will be created for this second instance of OpenSSO Enterprise through the authentication failover mechanism. (For more information, see "Servers and Sites" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.)

Failing the Servers and Sites check, 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 OpenSSO Enterprise are installed (generally, for failover purposes) sharing a one instance of the configuration data store.

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 "Naming Service". 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.

Mapping Fully Qualified Domain Names

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 in the Advance

Properties for Servers and Sites. For more information, see "Advanced" in *Sun OpenSSO Enterprise 8.0 Administration Reference*. 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 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 *server_name* property.

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 OpenSSO Enterprise 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.

Using Persistent Cookies

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 cookie value is a 3DES-encrypted string containing the userDN, realm name, authentication module name, maximum session time, idle time, and cache time. "To Use Persistent Cookies" on page 90 contains the procedure to enable this function.

Note – Persistent cookies do not work with the Distributed Authentication User Interface or when using Cross Domain Single Sign-on. If enabled and using either of these options, a regular cookie will be created.

▼ To Use Persistent Cookies

Programmatically, Persistent Cookie mode can be enabled by using the `setPersistentCookieOn()` method of the `com.sun.identity.authentication.spi.AMLoginModule` class. See Chapter 1, "Using the Authentication Service API and SPI," in *Sun OpenSSO Enterprise 8.0 Developer's Guide* for more information.

1 **Configure the Authentication Service to use persistent cookies in the Core authentication module.**

Enable the Persistent Cookie Mode attribute and modify, if necessary, the value of the Persistent Cookie Maximum Time attribute. See "To Modify Core Authentication Properties By Realm" on page 45 for the procedure.

2 **(Optional) Modify the persistent cookie name using this sub procedure.**

a. **Click Servers and Sites under the Configuration tab.**

b. **Click Default Server Settings.**

c. **Click Advanced.**

d. **Change the value of the `com.iplanet.am.pcookie.name` property.**

The default value is `DProPCookie`.

e. **Click Save.**

3 **Append the `iPSPCookie=yes` parameter to the User Interface Login URL.**

```
http://OpenSSO-machine-name.domain:port/opensso/UI/Login?realm=hr&iPSPCookie=yes
```

Once the user authenticates using this URL, if the browser is closed, the user can open a new browser window and will be redirected to the console without re-authenticating. This will work until the time defined as the value of Persistent Cookie Maximum Time elapses.

Upgrading Sessions

The Authentication Service enables you to upgrade a valid session token based on a second, successful authentication performed by the same user to the same 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 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 to the new realm 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 value of the `timeout` attribute of the `Callback` property of the page in each authentication module's specific XML file. The default value is 1 minute. This value can not be set using the OpenSSO Enterprise console.
- The value of the `Invalidate Session Max Time` property is the duration (in minutes) after which an invalid session will be removed from the session table after it is created but before the user logs in.
  1. Click Servers and Sites under the Configuration tab.
  2. Click Default Server Settings.
  3. Click the Session tab to modify `Invalidate Session Max Time`.
- The value of the `Maximum Session Time` attribute; by default, 120 minutes.
  1. Click Global under the Configuration tab.
  2. Click Session in the Service Name list.
  3. Modify `Maximum Session Time`.

Caution – The values of `Invalidate Session Max Time` and `Maximum Session Time` should be greater than the value of the `timeout` attribute; otherwise, the valid session information during session upgrade will be lost and URL redirection to the previous successful URL will fail.

Sharing User Credentials Among Authentication Modules (Shared State)

The Java Authentication and Authorization Service (on which the Authentication Service is built) has an option to enable *shared state* which allows for sharing of both the user ID and password between authentication modules. For example, assume an authentication chain is configured as follows:

- LDAP authentication is REQUIRED and shared state is enabled.
- Data Store authentication is REQUIRED and shared state is enabled. Additionally, the shared state behavior is configured to use the *first pass*.

For this authentication process, the user would be presented with an LDAP login page to enter a user ID and password. Assuming successful LDAP authentication, these credentials would then be passed to the Data Store module on the backend; the user would not see a Data Store login page. If Data Store authentication is successful, the user would be redirected to the appropriate page.

The shared state is enabled by entering the appropriate options to the authentication module as you configure an authentication chain. The options are:

`iplanet-am-auth-shared-state-enabled`
This option enables the use of a shared state map.

`iplanet-am-auth-store-shared-state-enabled`
This option enables the storage of credentials to a shared state map.

`iplanet-am-auth-shared-state-behavior-pattern`
To prevent a user from having to enter the user ID and password twice for authentication, set this option to `useFirstPass` for all modules in the chain (except the first). `tryFirstPass` (the default value) would prompt for new credentials if the shared state credentials failed

After a commit, an abort or a logout, the shared state will be cleared. To add shared state options to an authentication module in an authentication chain, see "Creating Authentication Chains" on page 62.

Redirecting Users After Authentication

Redirecting users to a particular URL upon successful or failed authentication can be configured in a number of ways.

- To set the redirection URL globally or per realm, see "Configuring the Core Authentication Service" on page 43.
- To set the redirection URL as a login URL parameter, see "Accessing the Authentication Service User Interface with a Login URL" on page 80.
- To set the redirection URL for a specific user, see Chapter 5, "Creating Subjects."

Information on how the precedence of redirection URLs is determined is in "Initiating the Authentication Type" on page 64.

Configuring Multiple LDAP Authentication Modules (Legacy Mode)

As a form of failover or to configure multiple values for an attribute when the OpenSSO Enterprise 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 Configure Multiple LDAP Authentication Modules

Any additional LDAP authentication module configurations created using this procedure can not be seen or modified using the OpenSSO Enterprise console.

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

The available LDAP authentication module attributes are in the `amAuthLDAP.xml` file. Any or all of these attributes can be used for the additional LDAP configurations defined in this step. Following is an example of a sub-configuration file that includes values for all attributes available to the LDAP authentication configuration.

```
>
<!--
  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 Load the XML file using the `ssoadm` command line tool.

CHAPTER 4

Managing Policies

The OpenSSO Enterprise Policy Service enables the top level administrator or top level policy administrator to view, create, delete, modify, and delegate *policies* to define access to an organization's protected resources. This chapter describes how to configure and manage policies using the OpenSSO Enterprise console. It contains the following sections.

- "Understanding the Authorization Process" on page 97
- "Defining a Policy and a Referral" on page 99
- "Creating Policies and Referrals" on page 107
- "Modifying Policies and Referrals" on page 114
- "Using Wild Cards in Policies" on page 119
- "Applying Policy Logic" on page 119
- "Enabling Policy in a Service" on page 120
- "Authenticating Based on Resource" on page 123

Understanding the Authorization Process

Businesses have applications and services that they need to protect. A *policy* defines rules that specify who or what can access these protected resources. The rules are, in effect, permissions describing when and how a principal can perform an action on a given protected resource. (A *principal* can be an individual, a corporation, a role, or a group.) In general, the permissions define what a principal can do to which resource and under what conditions. Towards this end, OpenSSO Enterprise provides the *Policy Configuration Service* to define global and realm preferences that effect the *Policy Service* which is used to create and manage the policies. (For more on the Policy Configuration Service, see "Policy Configuration" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.)

The Policy Service allows administrators to define, modify, grant, revoke and delete the policies that protect resources within the OpenSSO Enterprise deployment. Typically, a policy service includes a repository to store policies, interfaces that allow for the creation, administration and evaluation of policies, and a *policy agent* to enforce the policies. By default, OpenSSO Enterprise uses the configuration data store for policy storage, and the OpenSSO Enterprise console for policy management. Policy agents specifically developed for OpenSSO Enterprise act as the policy enforcement point (PEP). (OpenSSO Enterprise also provides Java and C interfaces for policy creation, policy evaluation, policy agent development, and Policy Service customization. See the *Sun OpenSSO Enterprise 8.0 Developer's Guide* and the *Sun OpenSSO Enterprise 8.0 C API Reference for Application and Web Policy Agent Developers* for more information.)

The policy agent is installed remotely from OpenSSO Enterprise and serves as an authorization step (following user authentication) when a user requests access to a protected resource. For example, a Human Resources server that is protected by a remote instance of OpenSSO Enterprise has an agent installed on it to act as a PEP. This agent would prevent personnel without the proper permission from viewing confidential salary information or other sensitive data. (More information on installing and administrating the policy agents can be found in the *Sun OpenSSO Enterprise Policy Agent 3.0 User's Guide for J2EE Agents* and the *Sun OpenSSO Enterprise Policy Agent 3.0 User's Guide for Web Agents*. Most policy agents can be downloaded from the Sun Microsystems Download Center.)

The authorization process begins when a web browser requests a URL that resides on a server protected by the policy agent. The policy agent installed on the server intercepts the request and checks for an existing OpenSSO Enterprise session token (`SSOToken`) for authentication credentials. If the session token is valid, the user is allowed or denied access based on the applicable configured policy. If the token is invalid or there is no existing session token, the user is redirected to the Authentication Service and the following occurs.

1. The policy agent redirects the user to the Authentication Service login page.
2. The user submits credentials for authentication.
3. Assuming successful authentication, the agent issues a request to the OpenSSO Enterprise Naming Service.
4. The Naming Service returns locators for the Policy Service, the Session Service, and the Logging Service.
5. The policy agent sends a request to the Policy Service for the policy applicable to the requesting user and resource.
6. Based on the returned policy, a decision is reached and the user is allowed or denied access.

   The policy agent may redirect the request back to the Authentication Service if the policy decision indicates a different authentication level or authentication mechanism is needed.

Note – 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.

Defining a Policy and a Referral

There are two configurations that can be created using the OpenSSO Enterprise Policy Service: a *policy* controls access to a protected resource and consists of *rules*, *subjects*, *conditions*, and *response providers* values, and a *referral* that delegates policy creation and evaluation to a particular entity and consists of one or more *rules* and one or more *referrals*. The following sections have more information.

- "Policy" on page 99
- "Referral" on page 106

Policy

In OpenSSO Enterprise, a *policy* (referred to as a *normal* policy in previous releases) consists of *rules*, *subjects*, *conditions*, and *response providers*. A policy can be defined with 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. The following sections have more information on the components of a policy.

- "Rules" on page 99
- "Subjects" on page 101
- "Conditions" on page 103
- "Response Providers" on page 105

Rules

A *rule* contains a service type and one or more actions with an appropriate value that, in effect, defines the intent of the policy. The *service type* defines the policy-enabled resource that is being protected. An *action* is the operation that can be performed on the resource; examples of web server actions are POST or GET. A *value* defines the permission for the action, for example, Allow or Deny. An example of an allowable action for a human resources service might be to change a home telephone number.

Note – For some services, it is acceptable to define an action without resources.

By default, there are three OpenSSO Enterprise services enabled for policy protection. The following sections explain these policy-enabled service types and the actions that can be applied to each.

- "Discovery Service (with resource name)" on page 100
- "Liberty Personal Profile Service (with resource name)" on page 101
- "URL Policy Agent (with resource name)" on page 101

Tip – When assessing policy, deny rules always take precedence over allow rules. For example, if you have two policies for a given resource, one with a rule denying access and the other with a rule allowing access the result, provided that the conditions for both policies are met, is deny. Thus, it is recommended that deny policies be used with extreme caution as they may potentially lead to conflicts. For example, if explicit deny rules are used, policies that are assigned to a given user through different subject types (such as a role or group membership) may result in denied access to a resource. This is illustrated if there is a deny policy for a resource applicable to an Employee role and an allow policy for the same resource applicable to Manager role; policy decisions for users assigned both Employee and Manager roles would be denied. Typically, the policy should use allow unless there are no policies to accomplish the deny case.

One way to resolve this issue is to design policies using Condition plug-ins. In the example, 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 to differentiate subjects. Another option is to use the Authentication Level condition stipulating that subjects with the Manager role authenticate at a higher authentication level.

Discovery Service (with resource name)

Discovery Service (with resource name) allows administrators to create and manage policies corresponding to the LOOKUP and UPDATE actions that can be performed on the Discovery Service.

- LOOKUP
  - Allow: Enables access to the resource defined in the Rule.
  - Deny: Denies access to the resource defined in the Rule.
- UPDATE
  - Allow: Enables access to the resource defined in the Rule.

  - Deny: Denies access to the resource defined in the Rule.

Liberty Personal Profile Service (with resource name)

Liberty Personal Profile Service (with resource name) allows administrators to create and manage policies corresponding to the MODIFY and QUERY actions that can be performed on the Liberty Personal Profile Service.

- MODIFY
  - Interact for Value: Invokes the Liberty Alliance Project Interaction protocol for a value on a resource.
  - Interact for Consent: Invokes the Liberty Alliance Project Interaction protocol for consent on a resource.
  - Allow: Enables access to the resource defined in the Rule.
  - Deny: Denies access to the resource defined in the Rule.
- QUERY
  - Interact for Value: Invokes the Liberty Alliance Project Interaction protocol for a value on a resource.
  - Interact for Consent: Invokes the Liberty Alliance Project Interaction protocol for consent on a resource.
  - Allow: Enables access to the resource defined in the Rule.
  - Deny: Denies access to the resource defined in the Rule.

URL Policy Agent (with resource name)

URL Policy Agent (with resource name) allows administrators to create and manage policies for policy agents that enforce decisions on `http/http(s)` URLs.

- GET
  - Allow: Enables access to the resource defined in the Rule.
  - Deny: Denies access to the resource defined in the Rule.
- POST
  - Allow: Enables access to the resource defined in the Rule.
  - Deny: Denies access to the resource defined in the Rule.

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:

Authenticated Users	This subject type implies that any user with a valid `SSOToken` is a member. Thus, all authenticated users are member of this Subject even if they have authenticated to a realm that is different from the one in which the policy is defined. This is useful if the resource owner would like to offer access to resources managed for users from other organizations. To restrict access to protected resources belonging to a specific organization, use the Organization subject.
OpenSSO Identity Subject	This subject type implies that the identities defined under the Subjects tab of a particular realm can be added as a member.
Web Services Clients	This subject type implies that a web services client (WSC) identified by a valid `SSOToken` is a member IF the Distinguished Name (DN) of any principal contained in the `SSOToken` matches any value of this subject. Valid values are the DNs of trusted certificates in a local Java keystore that correspond to the certificates of trusted WSCs. This subject type has dependency on the Liberty Alliance Project Web Services Framework and should be used to authorize WSCs only by service providers that implement it. Also, be sure to create the keystore before you add this Subject to a policy.

The following additional subjects are available by selecting them in the Policy Configuration Service of the realm. If you enable any of them, you should also change the values of the LDAP Bind DN and LDAP Bind Password attributes in the Policy Configuration Service of the realm to reflect valid credentials for the LDAP directory. Please note that `cn=amldapuser,ou=DSAME Users` and the top level suffix is not created in the default configuration directory.

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.

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

LDAP Users	This subject type implies that any LDAP user is a member of this subject.
OpenSSO Roles (Legacy Mode)	This subject type implies that any member of an OpenSSO Enterprise role is a member of this subject. An OpenSSO Enterprise role is created using OpenSSO Enterprise running in legacy mode with the AMSDK datastore and has object classes mandated by OpenSSO Enterprise. OpenSSO Roles can only be accessed through the hosting OpenSSO Enterprise Policy Service.
Organization	This subject type implies that any member of a realm is a member of this subject

---

Note – All OpenSSO Roles can be used as Directory Server roles. However, all LDAP roles are not necessarily OpenSSO Enterprise roles. LDAP roles can be leveraged from an existing directory by configuring the Policy Configuration Service. OpenSSO Enterprise roles can only be accessed through the hosting OpenSSO Enterprise Policy Service. The LDAP Role Search filter can be modified in the Policy Configuration Service to narrow the scope and improve performance.

Conditions

A condition allows you to define constraints on the policy. For example, if you want to limit access to a paycheck application, you can define a condition specifying the hours; or, you may wish to define a condition that only grants access if the request originates from a given set of IP addresses or from a company intranet. For example, to configure `http://org.example.com/hr/*.jsp` so that it can only be accessed by users from `org.example.com` between 9 a.m. to 5 p.m., use an IP Condition along with a Time Condition. By 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. The default conditions are:

Active Session Time

Sets the condition based on user session data.

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 by Module Chain
: The policy applies if the user has successfully authenticated to the authentication chain in the specified realm. If no realm is specified, authentication to any realm at the authentication chain will satisfy the condition.

Authentication by Module Instance
: The policy applies if the user has successfully authenticated to the instantiated authentication module in the specified realm. If no realm is specified, authentication to any realm at the authentication module 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.

Current Session Properties
: Decides whether a policy is applicable to the request based on values set in the user's OpenSSO Enterprise 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 in the condition with multiple values, it is sufficient if the token has at least one value for the property.

Note – Session properties set by OpenSSO are prefixed with `am.protected` to ensure that they cannot be edited by the Client SDK.

Identity Membership
: Checks if the invocator `uuid` specified in the environment is a member of at least one `AMIdentity` object specified in the Condition. The `uuid` invocator is specified as the key value of Condition. `INVOCATOR_PRINCIPAL_UUID` in the `environment` parameter of the evaluation request. This is primarily used to apply authorization rules for WSC (Web Service Client). The identity of the WSC is passed as the value of `uuid` invocator.

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 that the policy is defined.

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 OpenSSO Enterprise JVM.

If a request for access is negated as determined by the condition, an *advice message* can be produced to indicate why. Advice messages are propagated in the policy decision and sent to the policy agent which retrieves the advice and takes appropriate action — for example, redirecting the user to the Authentication Service to authenticate to a higher level. If, in this example, the user successfully authenticates to a higher level the policy might then become applicable. See `com.sun.identity.policy` in the *Sun OpenSSO Enterprise 8.0 Java API Reference* for more information.

Tip – Custom conditions can also produce advices. However, the policy agents respond only for Auth Level Advice and Auth Scheme Advice. Custom agents can be written to respond to more advices and existing OpenSSO Enterprise policy agents can be extended to do the same. See the *Sun OpenSSO Enterprise Policy Agent 3.0 User's Guide for J2EE Agents* for more information.

Response Providers

Response providers are plug-ins that provide values, in the final policy decision, for particular attributes in the authorized user's profile. The attributes are sent with the policy decision to the policy agent which, in turn, passes them in headers to an application. The application typically uses these attributes for customizing pages such as a portal page. OpenSSO Enterprise includes one implementation of the `com.sun.identity.policy.interfaces.ResponseProvider` interface, the `IDResponseProvider`. Custom response providers can be implemented.

Referral

A *referral* (referred to as a *referral* policy in previous releases) controls the delegation of both policy creation and evaluation. It consists of one or more *rules* and one or more *referrals*. Using a referral policy allows an administrator to delegate the administration of a realm's policy definitions and decisions to a sub or peer realm. Alternatively, policy decisions for a resource can be delegated to other policy products.

Note – The Policy Configuration service contains a global attribute called Realm 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.

The following sections have more information on the components of a referral.

- "Rules" on page 106
- "Referrals" on page 106

Rules

A referral *rule* defines the resource whose policy definition and evaluation is being referred. By default, there are three OpenSSO Enterprise services enabled for policy referral.

- Discovery Service (with resource name) allows administrators to create and manage policies for the OpenSSO Enterprise implementation of the Liberty Alliance Project
- Liberty Personal Profile Service (with resource name)
- URL Policy Agent (with resource name) allows administrators to create and manage policies for a policy agent. Policy agents enforce decisions on `http/http(s)` URLs.

Referrals

The *referral* defines the realm to which policy definition and evaluation is being referred. The referral can delegate to a *peer realm* (on the same level) or a *sub realm* (on a subordinate level). The realm to which policy definition or evaluation is referred can define and evaluate access only for those protected resources (or sub-resources) that have been referred to it. (This restriction does not apply to the top-level realm.) For more information, see "To Create a Referral Using the OpenSSO Enterprise Console" on page 112.

Creating Policies and Referrals

Policies are generally configured by creating an XML file and importing the data to OpenSSO Enterprise using the `ssoadm` command line utility but, they can also be created using the OpenSSO Enterprise console. (You can also create, modify and delete policies using the Policy API. See the *Sun OpenSSO Enterprise 8.0 Developer's Guide* for more information.) The following sections contain procedures for creating policies or referrals using the `ssoadm` command line utility and the OpenSSO Enterprise console. In general, policy is created at the realm (or sub realm) level for use throughout the particular realm's tree.

- "To Add Multiple Policies Using the `ssoadm` Command Line Utility" on page 107
- "To Create a Policy Using the OpenSSO Enterprise Console" on page 111
- "To Create a Referral Using the OpenSSO Enterprise Console" on page 112

Tip – Wildcards are supported in policy definitions. For information see *Sun OpenSSO Enterprise Policy Agent 3.0 User's Guide for J2EE Agents*.

▼ To Add Multiple Policies Using the `ssoadm` Command Line Utility

Before You Begin Before adding policies to OpenSSO Enterprise using `ssoadm` ensure that:

- The relevant realms exist (for example, if creating sub realm or peer realm referrals).
- Any authentication modules defined in the policies are registered with the appropriate realm.
- The corresponding LDAP objects (groups, roles and users) exist.
- OpenSSO Enterprise roles exist when creating `IdentityServerRoles` subjects.

1 Create an XML file with policy definitions.

To add multiple policies simultaneously, place all policy definitions in one XML file (as opposed to having one policy per XML file). This will help to avoid issues with the policy index. Following is an example of an XML file with policy definitions.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Policies
PUBLIC "-//Sun Java System Access Manager 7.1 2006Q3 Admin CLI DTD//EN"
"jar://com/sun/identity/policy/policyAdmin.dtd">

<Policies>
<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  OpenSSO Enterprise, O=Sun, C=US</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>
</Policies>
```

Note – The Value element of the following subject attributes takes the full DN:

- `SubrealmReferral`
- `PeerRealmReferral`
- `Realm`
- `IdentityServerRoles`
- `LDAPGroups`
- `LDAPRoles`
- `LDAPUsers`

---

2 **Add the defined policies to OpenSSO Enterprise using `ssoadm` with the XML file as input.**

`ssoadm create-policies --realm` *realm-name* `--xmlfile`
policy-xml-filename `--adminid` *administrator-id*
`--password-file` *password-filename*

For more information, see Chapter 1, "ssoadm Command Line Interface Reference," in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

▼ To Create a Policy Using the OpenSSO Enterprise Console

Before You Begin This procedure assumes you are logged into the OpenSSO Enterprise console as the administrator.

1 **Under the Access Control tab, click the name of the realm for which you are creating policy.**

2 **Click the Policies tab.**

3 **Click New Policy.**

4 **Enter a name for the policy.**

5 **(Optional) Enter a description of the policy.**

6 **(Optional) Select Yes to activate the policy.**

It is not necessary to define all of the policy's fields at this time. You may choose to add Rules, Subjects, Conditions, and Response Providers later. See "Modifying Policies and Referrals" on page 114 for information on these component's attributes.

7 **Click OK.**

▼ To Create a Referral Using the OpenSSO Enterprise Console

In order to create policies for peer realms or sub realms, you must first create a referral in the parent (or peer) realm pointing to the appropriate peer or sub realm. The Rule definition in the referral must contain the location of the resource(s) that will be managed. Once the referral is created, policies can be created for the appropriate peer or sub realm.

Before You Begin This procedure assumes you are logged into the OpenSSO Enterprise console as the administrator.

1 **Under the Access Control tab, click the name of the realm in which you are creating the referral.**

This might be the `/ Top Level Realm` or a sub realm.

2 **Click the Policies tab.**

3 **Click New Referral.**

4 **Enter a name for the referral.**

5 **(Optional) Enter a description of the referral.**

6 **(Optional) Select Yes to activate the referral.**

7 **Click New under Rules.**

8 **Select the appropriate Service Type and click Next.**

This value can not be changed once the Rule has been created. The options are:

- Discovery Service (with resource name) defines the authorization actions for Discovery Service query and modify protocol invocations by web services clients.
- Liberty Personal Profile Service (with resource name) defines the authorization actions for Liberty Personal Profile Service query and modify protocol invocations by web services clients.
- URL Policy Agent (with resource name) 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.

You may see a larger list if more services are enabled for the policy. (See "Enabling Policy in a Service" on page 120.) For more information, see "Rules" on page 99.

9 **Add a Name for the Rule.**

10 **Add a URL as the value for Resource Name and click Finish.**

In this procedure, `o=example.com` is the sub realm that manages access to `http://www.example.com` and its sub-resources.

11 **Click New under Referral.**

12 **(Optional) Select the Referral Type and click Next.**

The choices are Peer Realm or Sub Realm. This page is displayed only when the realm in which you are creating the referral has both peer and sub realms. It will not be displayed, for example, when creating a referral in the `/ Top Level Realm` because all realms are sub to the `/ Top Level Realm`.

13 **Enter a name for the referral.**

14 **Select the realm to which you are referring policy management from the drop down list and click Finish.**

15 **Click Save to update.**

16 **Navigate to the sub realm to create policy.**

Now that policy management for the resource is referred to the peer or sub realm, policies can be created to control access for `http://www.example.com` or any resource starting with `http://www.example.com`. See "To Add Multiple Policies Using the `ssoadm` Command Line Utility" on page 107 or "To Create a Policy Using the OpenSSO Enterprise Console" on page 111.

Modifying Policies and Referrals

Once a policy or referral is created, you can modify its components using the OpenSSO Enterprise console. Policies cannot be modified directly with `ssoadm`; you must first delete the policy before adding a modified version of it back. The following sections contain procedures on how to modify policies or referrals.

- "Modifying Policies" on page 114
- "Modifying Referrals" on page 117

Modifying Policies

You can modify a policy that has already been created. "To Modify a Policy" on page 114 describes the procedure to change or delete a policy.

▼ To Modify a Policy

Before You Begin This procedure assumes:

- You are logged into the OpenSSO Enterprise console as the administrator.
- You have already created the policy. See "Creating Policies and Referrals" on page 107.
- You did not previously define the policy's rules, subjects, conditions, and response providers. See "Creating Policies and Referrals" on page 107.

1 **Under the Access Control tab, click the name of the realm in which the policy you are modifying was created.**

2 **Click the Policies tab.**

3 **Click the name of the policy you are modifying.**

The policy's component page is displayed.

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

You can click the name of a Rule that has already been defined. The Rules attributes are the same whether you are defining them now or modifying definitions made in "Creating Policies and Referrals" on page 107. You can also select a defined Rule and delete it.

5 Select a Service Type for the rule.

This value can not be changed once the Rule has been created. The options are:

- Discovery Service (with resource name) defines the authorization actions for Discovery Service query and modify protocol invocations by web services clients.
- Liberty Personal Profile Service (with resource name) defines the authorization actions for Liberty Personal Profile Service query and modify protocol invocations by web services clients.
- URL Policy Agent (with resource name) 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 OpenSSO Enterprise policies.

You may see a larger list if more services are enabled for the policy. (See "Enabling Policy in a Service" on page 120.) For more information, see "Rules" on page 99.

6 Click Next to display the New Rule page and modify the following components.

a. Enter a Name for the Rule.

b. Enter a Resource Name for the rule.

Currently, policy agents only support `http://` and `https://` resources thus the value should be a URL. IP addresses are not supported. Wildcards are supported for protocol, host, port and resource name. For example:

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

For the URL Policy Agent service type, the default port number is 80 for `http://` and 443 for `https://` if no port number is defined.

c. Select the appropriate value for each Action.

Actions displayed are dependent on the chosen Service Type. See "Rules" on page 99 for an explanation of each Action.

d. Click Finish to return to the policy's components page.

7 Under the Subjects menu, click New.

You can also click the name of a Subject that has already been defined. The Subject attributes are the same whether you are defining them now or modifying definitions made in "Creating Policies and Referrals" on page 107. You can also select a defined Subject and delete it.

8 Select a Subject Type and click Next.

This value can not be changed once Subjects has been created. The options are:

- Authenticated Users implies that any user with a valid `SSOToken` is a member.

- OpenSSO Identity Subject implies that the identities defined under the Subjects tab of a particular realm can be added as a member.
- Web Services Clients implies that a WSC identified by a valid `SSOToken` is a member IF the Distinguished Name (DN) of any principal contained in the `SSOToken` matches any value of this subject.

For more information, see "Subjects" on page 101.

9 Enter a Name for the Subject.

10 Select whether the Subject is Exclusive.

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 identity.

11 If applicable to the selected Subject Type, choose entries to add for the subject.

a. Perform a search to display qualified entries.

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

b. Select an individual entry and click Add to move it to the Selected list.

Alternately, click Add All to add all of the entries at once.

12 Click Finish to return to the policy's components page.

13 Under the Conditions menu, click New.

You can also click the name of a Condition that has already been defined. The Conditions attributes are the same whether you are defining them now or modifying definitions made in "Creating Policies and Referrals" on page 107. You can also select a defined Condition and delete it.

14 Select a Condition Type and click Next.

15 Enter values for the Condition Type's listed attributes.

For more information, see "Conditions" on page 103.

16 Click Finish to return to the policy's components page.

17 Under the Response Provider menu, click New.

You can also click the name of a Response Provider that has already been defined. The Response Provider attributes are the same whether you are defining them now or modifying definitions made in "Creating Policies and Referrals" on page 107. You can also select a defined Response Provider and delete it.

18 Enter a Name for the Response Provider.

19 Define the following values:

Static Attribute	These are static attributes defined in an instance of `IDResponseProvider` stored in the policy. The value takes the format `attribute=value`.
Dynamic Attribute	The response attributes chosen here need to first be defined in the Selected Dynamic Response Attributes field of the Policy Configuration Service for the corresponding realm. The attribute names defined should be a subset of those existing in the identity repository. To select specific or multiple attributes, hold the Control key and click the left mouse button. For details, see "Policy Configuration" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

20 Click Finish.

21 Click Save to update the policy.

Modifying Referrals

You can modify the components of a referral that has already been created. "To Modify a Referral" on page 117 describes the procedure to change or delete a referral.

▼ To Modify a Referral

Before You Begin This procedure assumes:

- You are logged into the OpenSSO Enterprise console as the administrator.
- You have already created the referral. See "Creating Policies and Referrals" on page 107.
- You did not previously define the referral's components. See "Creating Policies and Referrals" on page 107.

1 Under the Access Control tab, click the name of the realm in which the policy you are modifying was created.

2 Click the Policies tab.

3 **Click the name of the referral you are modifying.**

The referral's component page is displayed.

4 **Under the Rules menu, click New to display the New Rule page and modify as follows.**

You can click the name of a Rule that has already been defined. The Rules attributes are the same whether you are defining them now or modifying definitions made in "Creating Policies and Referrals" on page 107. You can also select a defined Rule and delete it.

a. **Select the appropriate Service Type and click Next.**

This value can not be changed once the Rule has been created. The options are:

- Discovery Service (with resource name) defines the authorization actions for Discovery Service query and modify protocol invocations by web services clients.
- Liberty Personal Profile Service (with resource name) defines the authorization actions for Liberty Personal Profile Service query and modify protocol invocations by web services clients.
- URL Policy Agent (with resource name) 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.

You may see a larger list if more services are enabled for policy. (See "Enabling Policy in a Service" on page 120.) For more information, see "Rules" on page 99.

b. **Add a Name for the Rule.**

c. **Add a URL as the value for Resource Name and click Finish to return to the referral's components page.**

Currently, policy agents only support `http://` and `https://` resources thus the value should be a URL. IP addresses are not supported. Wildcards are supported for protocol, host, port and resource name. For example:

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

For the URL Policy Agent service type, the default port number is 80 for `http://` and 443 for `https://` if no port number is defined. In this example, `o=example.com` is the sub realm that manages access to `http://www.example.com` and its sub-resources.

5 **Under the Referrals menu, click New.**

You can click the name of a Referral that has already been defined. The Referrals attributes are the same whether you are defining them now or modifying definitions made in "Creating Policies and Referrals" on page 107. You can also select a defined Referral and delete it.

6 **Enter a Name for the Referral.**

7 **Specify a filter and click Search.**

This action defines the realm names that will be displayed in the Value field. By default, it will display all realm names.

8 **Select the realm to which you are referring policy administration from the drop down list.**

9 **Click Finish to return to the referral's components page.**

10 **Click Save to update the referral.**

Using Wild Cards in Policies

The Policy Service supports policy definitions using an asterisk (*) as the wild card. Only * is supported as a wild card and it can not be escaped as in `\*`. A * must:

- Match zero or more occurrences of any character.
- Span across multiple levels in a URL.

The following wild card matching rules assume the wildcard character is * and the delimiter character is `/`.

- * matches zero or more characters, including `/`, in the resource name.
- * matches one or more characters, including `/`, if the * appears at the end of the resource name and it is immediately preceded by a `/`. For example, `abc/*`doesn't match `abc`.
- Multiple consecutive `/` characters don't match with a single `/`. For example, `abc/*/xyz` doesn't match `abc/xyz`.
- For purposes of comparison, trailing `/` characters will not be considered as part of the resource name. For example, `abc/` or `abc//` will be treated the same as `abc`.

Applying Policy Logic

All of the following should be satisfied for a policy to be applicable to a request.

- The Resource Name defined in a policy's Rules should match that of the protected, requested resource. The match can be an exact literal match or one due to the presence of wild cards. Currently, policy agents only support `http://` and `https://` URLs as a Resource Name; they do not support IP addresses in place of the host name. Wild cards are supported as a substitution for a protocol, a host name, a port number, or a resource - as in `http*://*:*/*.html`

- The requesting user should satisfy at least one of the Subject(s) defined by the policy. For example, if the Subject type is defined as Access Manager Identity Subject, the requesting user should be a member of the role selected in the policy.
- At least one Condition in EACH selected Condition Type defined in a policy should be satisfied by the requesting user, resource and/or environment parameters. For example, if the policy is defined with two Time conditions and two IP Address/DNS Name conditions, the request should satisfy at least one Time condition and at least one IP Address/DNS Name condition.

Sometimes policies collide. When this happens, the following rules take effect.

- When multiple policies are applicable to a particular resource, the order in which the policies are evaluated is not deterministic.
- If a policy decision for a requested action is boolean, a value of false overrides one of true. For example, when deciding authorization for a web URL, deny overrides allow.
- If a policy decision for a requested action is boolean and the request is determined to be false based on policies evaluated thus far, no further policies will be evaluated for the requested action. This behavior can be changed by toggling the Continue Evaluation On Deny Decision attribute in the Policy Configuration Service.

Enabling Policy in a Service

You can protect services using the OpenSSO Enterprise Policy Service only if the service schema configures to the `sms.dtd` and contains values for the `<Policy>` schema and sub elements. If you want to create a custom policy agent or require more fine-grained enforcement than the OpenSSO Enterprise Policy Service offers, you can add the `<Policy>` schema to an OpenSSO Enterprise formatted service file, enabling it for interaction with your policy agent and the Policy Service. (For more information, see *Sun OpenSSO Enterprise 8.0 C API Reference for Application and Web Policy Agent Developers*.) Once enabled, the service will be displayed as a Service Type when creating policies and referrals. (See "Rules" on page 99.)

Tip – By default, OpenSSO Enterprise policy agents protect only URL resources in tandem with the URL Policy Agent Service (for which a policy evaluator is created and used to get policy decisions). The most common scenario is to use the policy agents developed specifically for OpenSSO Enterprise and the URL Policy Agent Service. For more information, see *Sun OpenSSO Enterprise Policy Agent 3.0 User's Guide for Web Agents* or *Sun OpenSSO Enterprise Policy Agent 3.0 User's Guide for J2EE Agents*.

"To Add a New Policy Enabled Service" on page 121 contains more information.

▼ To Add a New Policy Enabled Service

1 **Develop the new service in an XML file based on the `sms.dtd`.**

`amWebAgent.xml` is the XML service file for the URL Policy Agent service and can be used as a template to create a policy-enabled service file. It is located in the `templates` directory of the exploded `opensso.zip`. Here is another sample template.

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

<!--
Copyright (c) 2005 Sun Microsystems, Inc. All rights reserved
Use is subject to license terms.
-->





<!DOCTYPE ServicesConfiguration
  PUBLIC "=//iPlanet//Service Management Services (SMS) 1.0 DTD//EN"
  "jar://com/sun/identity/sm/sms.dtd">

<ServicesConfiguration>
  <Service name="SampleWebService" version="1.0">
     <Schema
        i18nFileName="SampleWebService"
        i18nKey="SampleWebService">

  <Policy>
    <AttributeSchema name="GET"
          type="single"
          syntax="boolean"
          uitype="radio"
          i18nKey="get">
            <IsResourceNameAllowed/>
               <BooleanValues>
                 <BooleanTrueValue i18nKey="allow">allow</BooleanTrueValue>
                 <BooleanFalseValue i18nKey="deny">deny</BooleanFalseValue>
               </BooleanValues>
    </AttributeSchema>

    <AttributeSchema name="POST"
          type="single"
        syntax="boolean"
        uitype="radio"
            i18nKey="post">
              <IsResourceNameAllowed/>
             <BooleanValues>
```

```
                <BooleanTrueValue i18nKey="allow">allow</BooleanTrueValue>
                <BooleanFalseValue i18nKey="deny">deny</BooleanFalseValue>
             </BooleanValues>
    </AttributeSchema>

    <AttributeSchema name="PUT"
            type="single"
        syntax="boolean"
        uitype="radio"
            i18nKey="put">
              <IsResourceNameAllowed/>
             <BooleanValues>
                <BooleanTrueValue i18nKey="allow">allow</BooleanTrueValue>
                <BooleanFalseValue i18nKey="deny">deny</BooleanFalseValue>
             </BooleanValues>
    </AttributeSchema>

    <AttributeSchema name="DELETE"
            type="single"
        syntax="boolean"
        uitype="radio"
            i18nKey="delete">
              <IsResourceNameAllowed/>
             <BooleanValues>
                <BooleanTrueValue i18nKey="allow">allow</BooleanTrueValue>
                <BooleanFalseValue i18nKey="deny">deny</BooleanFalseValue>
             </BooleanValues>
    </AttributeSchema>

    </Policy>
    </Schema>
  </Service>
</ServicesConfiguration>
```

2 **Save the XML file to the** `/config/xml/` **directory of the exploded** `opensso.zip`**.**

For example, `/config/xml/`*newServiceWithPolicy*`.xml`

3 **Load** `/config/xml/`*newServiceWithPolicy*`.xml` **using the** `ssoadm` **command line utility.**

See Chapter 1, "ssoadm Command Line Interface Reference," in *Sun OpenSSO Enterprise 8.0 Administration Reference* for more information.

4 **Define policy to protect the resource as documented in "Creating Policies and Referrals" on page 107.**

Authenticating Based on Resource

In a typical authentication scenario, if a user attempts access to a web resource without authentication credentials, the policy agent redirects the user to the default OpenSSO Enterprise authentication module login page even if the resource is protected by a different authentication module. Thus, the user must authenticate to both modules. The Gateway servlet provides *resource authentication* which allows the user to bypass the default authentication module and authenticate against the module protecting the resource only.

The Gateway servlet has the following limitations:

- The web resource being protected can not be defined in more than one policy. For example, if `abc.html` is defined in one policy as requiring LDAP authentication, it can not be defined in a second policy as requiring Certificate authentication.
- Level and scheme are the only conditions that can be defined in the policy protecting the resource.
- It does not work across different DNS domains.

To use resource authentication, you must make ensure certain configurations on the web container installed on the resource server machine as well as make configurations to OpenSSO Enterprise and the policy agent.

▼ To Configure Resource Authentication

Once both OpenSSO Enterprise and a policy agent have been installed and profile has been created for the policy agent, resource-based authentication can be configured. To do this, it is necessary to point OpenSSO Enterprise to the Gateway servlet.

Before You Begin Ensure the following configurations on the web container installed on the resource's server machine. Check your container's documentation for more information.

- Check that a certificate for the server (Server-Cert) and a certificate for the trusted Certificate Authority are installed.
- Add a listen socket for simple Secure Sockets Layer (SSL) and one for SSL client authentication.
- Ensure that the listener port configuration requires SSL for client authentication.

1 **Log in to the OpenSSO Enterprise console as administrator; by default,** `amadmin`**.**

2 **Under the Configuration tab, click Authentication.**

3 **Click the Certificate Service Name.**

4 **Enable Match Certificate in LDAP by checking the box.**

5 **Select Subject UID as the value for Certificate Field Used to Access User Profile.**

6 **Enter 54430 as a value for SSL Port Number.**

This port number must match the port number used for the web container's SSL client authentication listener port.

7 **Type 2 as the value for the Authentication Level attribute.**

The value used must be greater that the level defined for LDAP authentication.

8 **Click Save.**

9 **Click Back to Service Configuration.**

10 **Under the Access Control tab, click the name of the realm to which the policy agent belongs.**

11 **Click the Policies tab and add policies as follows.**

- Policy 1 has a condition of LDAP authentication only for `http://`*agent-machine*`.`*domain*`/banner.html`.
- Policy 2 has a condition of Certificate authentication only for `http://`*agent-machine*`.`*domain*`/banner2.html`
- Policy 3 has a condition of LDAP authentication and a level of Certificate authentication for `http://`*agent-machine*`.`*domain*`/banner3.html`.

12 **Click the Agents tab.**

13 **Click on the Web or J2EE tab depending on the agent being used.**

14 **Click on the Agent Profile name of the policy agent.**

15 **Under OpenSSO Services, enter the following URL as the value of the OpenSSO Login URL attribute:**

`http://`*OpenSSO Enterprise_host*`.`*domain*`:`*port*`/opensso/gateway`

16 **Add the following line to the file:**

`com.sun.am.policy.am.loginURL = http://`*OpenSSO Enterprise_host*`.`*domain*`:`*port***`/opensso/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 *Sun OpenSSO Enterprise 8.0 Developer's Guide*.

◆ ◆ ◆ CHAPTER 5

Creating Subjects

The Subjects interface enables basic identity management within a realm. Any identity created using this OpenSSO Enterprise console interface is created in the identity repository that was defined during the installation process. (If no external user data store is defined during installation, the user data store is the OpenSSO Enterprise embedded configuration data store; this configuration should be used for testing and demonstration purposes only.) The following sections contain more information on the entities you can create and modify:

- "Storing Subjects" on page 127
- "Creating Users" on page 127
- "Creating Groups" on page 130
- "Administrative Users and Default Subjects" on page 131

Storing Subjects

Any subject created using the OpenSSO Enterprise console is stored in the identity repository that was defined during the installation process. If no external user data store is defined, the user data store is, in effect, the OpenSSO Enterprise embedded configuration data store, and Subjects will be stored in it. This deployment should be used for testing and demonstration purposes only. Defining an external data store during OpenSSO Enterprise configuration, or pointing a realm to an instance of an identity repository after configuration, allows you to store subjects for real deployments.

Creating Users

A *user* represents an individual's identity. Users can be created and deleted, and can be added or removed from roles and/or groups. You can also assign services to the user. The following procedures contain more information.

- "To Create a User" on page 128

- "To Modify a User" on page 128

▼ To Create a User

Before You Begin This procedure assumes you are logged into the OpenSSO Enterprise console as the administrator; by default, `amadmin`.

1. **Under the Access Control tab, click the name of the realm in which you are creating the user.**
2. **Click the Subjects tab.**
3. **Click New.**
4. **Enter data for the following fields:**

   ID This field takes the identifier of the user purposes of logging into the OpenSSO Enterprise console. This property does not have to be a DN.

   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 user.

   Password (Confirm) Confirm the password.

   User Status This option indicates whether the user is allowed to authenticate through OpenSSO Enterprise.
5. **Click OK.**

   You can now modify the user profile by clicking the name of the user. For information on the user attributes, see the User attributes. Other modifications you can perform:

▼ To Modify a User

To add a user to a group or role, assign a service to a user profile or add values to the additional user profile attributes, modify the user profile.

Before You Begin This procedure assumes you are logged into the OpenSSO Enterprise console as the administrator; by default, `amadmin`.

1. **Under the Access Control tab, click the name of the realm in which you are creating the user.**
2. **Click the Subjects tab.**

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

The Edit User page is displayed under the General tab.

4 **(Optional) Add values to the following user profile attributes.**

- Password can be used to change the user's defined password.

Note – The top level administrator's username and password is created when you configure OpenSSO Enterprise. This password can be changed at any time through the console, or with the `ampassword` command line utility. This attribute is used to change the top level administrator password through the console. For more information on `ampassword`, see Chapter 3, "The ampassword Command Line Tool," in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

- Email Address
- Employee Number
- Telephone Number
- Home Address
- Account Expiration Date
- User Authentication Configuration defines the process to which the user must successfully authenticate.
- User Alias List 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.
- Success URL specifies the URL that the user will be redirected to upon successful authentication.
- Failure URL specifies the URL that the user will be redirected to upon failed authentication.
- Password Reset Options forces the user to change a defined password at the next login.
- MSISDN Number defines the user's Mobile Station International Subscriber Directory Number if using MSISDN authentication.

5 **Click Save to save the values.**

6 **Click the Services tab.**

7 **Click Add.**

8 **Select from the displayed services and click Next.**

9 **Modify the service's attributes and click Finish.**

10 **Click Finish.**

11 **Click the Groups tab to add the user to a specific group.**

12 **Add a group displayed in the Available list to the Selected list and click Save.**

13 **Click Back to Subjects.**

Creating Groups

A *group* represents a collection of users with a common function, feature or interest. Typically, a group has no privileges associated with it. Groups can exist at two levels; within a realm and within other managed groups. The following procedures have more information.

- "To Create a Group" on page 130
- "To Add Users to a Group" on page 130

▼ To Create a Group

Before You Begin This procedure assumes you are logged into the OpenSSO Enterprise console as the administrator; by default, `amadmin`.

1 **Under the Access Control tab, click the name of the realm in which you are creating the user.**

2 **Click the Subjects tab.**

3 **Click the Group tab.**

4 **Click New under the Group list.**

5 **Enter a name for the group in the ID field.**

6 **Click OK.**

Once you have created the group, you can add users to it. See "To Add Users to a Group" on page 130.

▼ To Add Users to a Group

Before You Begin This procedure assumes you are logged into the OpenSSO Enterprise console as the administrator; by default, `amadmin`.

1. **Under the Access Control tab, click the name of the realm in which you are creating the user.**
2. **Click the Subjects tab.**
3. **Click the Group tab.**
4. **Click the name of the group in the Group list to which you want to add users.**
5. **Click the User tab.**
6. **Add any Available users to the Selected list.**
7. **Click Save.**

   Users can also be added to Groups by modifying the User profile. See "To Modify a User" on page 128.

Administrative Users and Default Subjects

A number of administrative (and other) users are created as subjects during installation of OpenSSO Enterprise. The following sections contain information about each.

- "amadmin" on page 131
- "amldapuser" on page 132
- "UrlAccessAgent" on page 132
- "Directory Manager" on page 133
- "Administrator" on page 133
- "demo" on page 133
- "test" on page 133
- "dsameuser" on page 133
- "puser" on page 134
- "anonymous" on page 134

amadmin

The OpenSSO administrative user is amadmin (uid=amAdmin,ou=People,dc=opensso,dc=java,dc=net in the embedded configuration data store). This top-level administrator has unlimited access to all entries managed by OpenSSO. During installation, you must provide a password for amadmin. The amadmin profile is a Subject under the top-level realm. You cannot change the default amadmin identifier.

▼ To Change the `amadmin` Password

1. **Under the Access Control tab, click `/ (Top Level Realm)`.**
2. **Click the Subjects tab.**
3. **Click `amadmin` in the Users table.**
4. **Under the General tab, click the Password attribute's Edit link.**
5. **Type the old and new passwords as directed and click OK.**
6. **Click Save on the Edit User page.**

`amldapuser`

`amldapuser` (`cn=amldapuser,ou=DSAME Users,dc=opensso,dc=java,dc=net` in the embedded configuration data store) has read and search access to all embedded data store entries; it is used when the OpenSSO schema extends the embedded data store schema. `amldapuser` binds to the directory to retrieve data for the LDAP and Membership authentication modules and the Policy Configuration Service. The default password for `amldapuser` is *changeit*. You can change the password by modifying the value of the AMLDAPUSERPASSWD property in the *OpenSSO-Deploy-base*`/opensso/WEB-INF/classes/serviceDefaultValues.properties` file BEFORE running the OpenSSO configurator. Changing the `amldapuser` password after configuration is not supported.

`UrlAccessAgent`

`UrlAccessAgent` is the user that a web agent uses to login to OpenSSO. The password for `UrlAccessAgent` is defined during OpenSSO configuration.

Note – `amService-UrlAccessAgent` (`cn=amService-UrlAccessAgent,ou=DSAME Users,dc=opensso,dc=java,dc=net` in the embedded configuration data store) is the same user as `UrlAccessAgent`. When entered as `UrlAccessAgent` on the server side, the Authentication Service prepends to it the string `amService-`. The Authentication Service then authenticates it is a special user with an entry in the data store.

Directory Manager

CN=Directory Manager,CN=Users,dc=opensso,dc=java,dc=net is the default top level administrator for the embedded configuration data store (OpenDS). Directory Manager has read and write access to all entries in the embedded configuration data store and would be used to bind to it if the OpenSSO schema is not installed.

Administrator

CN=Administrator,CN=Users,dc=opensso,dc=java,dc=net is the default top level administrator for Microsoft Active Directory. This is similar to Directory Manager.

demo

demo is the user used to demonstrate the federation-related features of OpenSSO. By default, its password is changeit. This user is displayed as a subject of the top-level realm in the OpenSSO console and its default password can be changed.

test

test user is used to execute some OpenSSO samples. These samples would create the test user and test will be displayed as a subject of the top-level realm in the OpenSSO console after executing them. The default password is test.

dsameuser

dsameuser (cn=dsameuser,ou=DSAME Users,dc=opensso,dc-java,dc=net) binds to the embedded configuration data store when the OpenSSO SDK performs operations on it that are not linked to a particular user (for example, retrieving service configuration information).

After installation, it is recommended that you change the password for dsameuser. Do not use the same password that was set for amadmin or amldapuser. To change the password, use the ampassword utility with the --admin (or -a) option. (This option does not change the amadmin password.) If OpenSSO is deployed on multiple host servers, change the password in the embedded configuration data store and the local serverconfig.xml file on the first server as documented using ampassword. For each additional server, encrypt the new password using ampassword with the --encrypt (or -e) option and swap the new encrypted password with the old in the serverconfig.xml file manually. Restart each OpenSSO web container after the modification.

`puser`

Proxy user (`cn=puser,ou=DSAME Users,dc=opensso,dc=java,dc=net`) is a proxy user that works behind the scenes for the legacy AMSDK. This user is created during installation and cannot be modified or found in the OpenSSO console.

`anonymous`

`anonymous` is the default anonymous user. If the Anonymous authentication module is enabled, an anonymous user can log into OpenSSO without providing a password. You can define a list of anonymous users by adding user identifiers to the anonymous profile using the OpenSSO console.

CHAPTER 6

Storing Policy Agent and Web Services Security Agent Profiles

OpenSSO Enterprise offers a centralized configuration interface for remote policy agent profiles and web services security related information. The profiles are stored in the embedded configuration data store and managed by an administrator using the OpenSSO Enterprise console. This chapter contains the following sections:

- "Centralizing Agent Profiles" on page 135
- "Creating New Agent Profiles and Groups" on page 138

Centralizing Agent Profiles

OpenSSO Enterprise leverages its embedded configuration data store for centralizing the storage of remote policy agent profiles and web services security related information. By moving this configuration data to OpenSSO Enterprise, an administrator can use the console or the command line interface tools to manage the properties and values. Any configuration changes to the *hot-swappable* properties are conveyed immediately. The following sections have more infomration on the agent profiles that can be configured.

- "Web Policy Agent Profile" on page 136
- "J2EE Policy Agent Profile" on page 136
- "Web Service Provider Security Agent Profile" on page 136
- "Web Service Client Security Agent Profile" on page 136
- "STS Client Agent Profile" on page 137
- "2.2 Agents" on page 138
- "Agent Authenticator" on page 138

Attribute descriptions for the agent profiles are in Chapter 5, "Centralized Agent Configuration Attributes," in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

Web Policy Agent Profile

Values for the configuration properties of a web policy agent can be defined using OpenSSO Enterprise if, during the web policy agent profile creation, centralized configuration was chosen. If local configuration was selected, the properties related to this policy agent profile must be modified directly in the `OpenSSOAgentConfiguration.properties` file in the agent installation directory on the agent's host machine. For detailed information on web policy agents, see the *Sun OpenSSO Enterprise Policy Agent 3.0 User's Guide for Web Agents*

J2EE Policy Agent Profile

Values for the configuration properties of a J2EE policy agent can be defined using OpenSSO Enterprise if, during the J2EE policy agent profile creation, centralized configuration was chosen. If local configuration was selected, the properties related to this agent must be modified directly in the `OpenSSOAgentConfiguration.properites` file in the agent installation directory on the agent's host machine. For detailed information on J2EE policy agents, see the *Sun OpenSSO Enterprise Policy Agent 3.0 User's Guide for J2EE Agents*.

Web Service Provider Security Agent Profile

The Web Service Provider (WSP) security agent profile stores the configuration data related to validating a request from a web service client and securing the response returned by the WSP. The data includes the WSP's supported security mechanisms, keystore locations, SAML configurations and endpoints. The WSP agent profile also has a mechanism to authenticate against OpenSSO Enterprise to generate a session for the WSP. For more information, see Part IV, "The Web Services Stack, Identity Services, and Web Services Security," in *Sun OpenSSO Enterprise 8.0 Technical Overview*.

Out of the box, `wsp` is the default WSP security agent profile. Additional profiles can be defined with the profile name dependant on the endpoint of the service defined in the web service provider's WSDL file. (The security agent searches based on the endpoint.) This allows multiple web service providers to use the same configuration data store. The name of the web service provider must be unique across all agents.

Caution – The Group functionality is not supported with the Web Service Provider Security Agent Profile.

Web Service Client Security Agent Profile

The Web Service Client (WSC) security agent profile stores the configuration data related to securing a request from a WSC and validating the request when received by the WSP. The data includes the WSP's supported security mechanisms, keystore locations, SAML configurations, signing and encryption instructions, and endpoints. It also defines whether an end user token should be generated. For more information, see Part IV, "The Web Services Stack, Identity Services, and Web Services Security," in *Sun OpenSSO Enterprise 8.0 Technical Overview*.

Out of the box, `wsc` is the default WSC security agent profile. Additional profiles can be defined with the profile name dependant on the service name defined in the web service client's WSDL file. (The security agent searches based on the service name.) This allows multiple web service clients to use the same configuration data store. The name of the web service client must be unique across all agents.

Caution – The Group functionality is not supported with the Web Service Client Security Agent Profile.

STS Client Agent Profile

The Security Token Service (STS) Client agent profile stores the configuration data related to securing an outbound request to the OpenSSO Enterprise Security Token Service or Discovery Service to obtain a security token. The data includes the supported security mechanisms, keystore locations, signing and encryption instructions, and endpoints.

- The *Discovery Agent* allows you to store data used to communicate with the Discovery Service to obtain a security token based on the Liberty Alliance Project specifications. The token secures communications between the client and the Discovery Service end point. This option is defined as the value of the Discovery Configuration attribute in the WSC security agent profile.
- The *STS Agent* allows you to store data used to communicate with the Security Token Service to obtain a security token based on the WS-Trust specifications. The token secures communications between the client and the Security Token Service end point. This option is defined as the value of the STS Configuration attribute in the WSC security agent profile. Out of the box, `SecurityTokenService` is the default token agent profile for the Security Token Service. Additional profiles can be defined with the profile name dependant on the service name defined in the security token service's WSDL file. (The security agent searches based on the service name.) This allows multiple security token services to use the same configuration data store. The name of the security token service must be unique across all agents.

For more information, see Part IV, "The Web Services Stack, Identity Services, and Web Services Security," in *Sun OpenSSO Enterprise 8.0 Technical Overview*.

Caution – The Group functionality is not supported with the STS Client Agent Profile.

2.2 Agents

OpenSSO Enterprise is backward compatible with OpenSSO Enterprise web and J2EE Policy Agents 2.2. Policy Agents 2.2 must be configured local to the deployment container on which it is installed thus, from the OpenSSO Enterprise console, there are a limited number of options that can be configured. For more information, see *Sun Java System Access Manager Policy Agent 2.2 User's Guide*.

Agent Authenticator

An agent authenticator is a type of agent that, once it is authenticated, can obtain the read-only data of agent profiles of any type (policy, security or token) for purposes of authenticating the agent. The agent profiles must be defined in the Agent Authenticator profile and exist in the same realm. Users that have the Agent Authenticator's username and password can read agent profile data, but do not have the create, update, or delete permissions of the Agent Administrator.

Creating New Agent Profiles and Groups

This section contains the following procedures.

- "To Create a New Agent Profile" on page 138
- "To Create a New Group" on page 140
- "To Modify an Agent Profile to Inherit Properties From a Group" on page 141

▼ To Create a New Agent Profile

You can create a new agent profile using the OpenSSO Enterprise console. Some of the individual steps documented do not apply to all agent profile types.

Before You Begin This procedure assumes you are logged into the OpenSSO Enterprise console as the administrator; by default, `amadmin`.

1. **Under the Access Control tab, click the name of the realm in which you are creating the agent profile.**

2. **Click the Agents tab.**

3. **Select the tab for the appropriate agent type.**

   - **Web Agents**

- **J2EE Agents**
- **Web Service Provider Agents**
- **Web Service Client Agents**
- **STS Client Agent**
- **2.2 Agents**
- **Agent Authenticator**

4 In the Agent section, click New.

The STS Client agent profile displays a pop-up from which you choose the token agent type: Discovery or STS. For more information, see “STS Client” in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

5 In the Name field, enter the name for the new agent profile.

6 Enter and confirm the Password.

Caution – For web policy agents only, this password must be the same password that you enter in the agent profile password file that you specify when you run the `agentadmin` program to install the agent.

Steps 7–9 Apply to Web and J2EE policy agents only.

7 For Web and J2EE policy agents only, configure using the following sub procedure.

For other agent profile types, configure the attributes as documented in Chapter 5, “Centralized Agent Configuration Attributes,” in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

a. Select Local or Centralized configuration.

When local configuration is selected, the properties related to this agent cannot be edited using the console. In such a scenario, the agent retrieves configuration data from the local (to the installed agent) `OpenSSOAgentBootstrap.properties` and `OpenSSOAgentConfiguration.properties` files in the agent installation directory. Property values for the locally configured agents are modified directly in the `OpenSSOAgentConfiguration.properties` file.

b. In the Server URL field, enter the OpenSSO Enterprise server URL.

For example:

`http://`*OpenSSO-Host*`.example.com:8080/OpenSSO`

c. **In the Agent URL field, enter the URL for the agent application,** `agentapp`**.**

- For a web policy agent: **`http://`***Agent-Host***`.example.com:8090`**
- For a J2EE policy agent: **`http://`***Agent-Host***`.example.com:8090/agentapp`**

8 **Click Create.**

The agent profile is created. To do additional configurations for the agent profile, click the profile name to display the Edit agent page. Agent attribute descriptions are listed and defined in Chapter 5, "Centralized Agent Configuration Attributes," in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

▼ To Create a New Group

Agents can inherit properties from their group. For example, web policy agents can inherit properties from a web policy agent group.

Caution – The Group functionality is not supported with the web services and STS Client Agent Profiles.

Before You Begin This procedure assumes you are logged into the OpenSSO Enterprise console as the administrator; by default, `amadmin`.

1 **Under the Access Control tab, click the name of the realm to which the group will belong.**

2 **Click the Agents tab.**

3 **Select the tab for the appropriate agent type.**

4 **In the Group section, click New.**

5 **Enter a name for the new group.**

6 **Enter the OpenSSO Enterprise Server URL (for Web and J2EE policy agents only).**

For example, **`http://`***OpenSSO-Host***`.example.com:8080/OpenSSO Enterprise`**. For other agent profile types, configure the attributes as documented in Chapter 5, "Centralized Agent Configuration Attributes," in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

7 **Click Create.**

The agent group is created. To do additional configurations for the agent group, click the group name to display the Edit Group page. Attribute descriptions are listed and defined in Chapter 5, "Centralized Agent Configuration Attributes," in *Sun OpenSSO Enterprise 8.0 Administration*

Reference. (The properties you can set for a group are almost the same as those for an individual agent; the Group, Password, and Password Confirm properties are not available at the group level.)

Caution – Some group properties have variable values assigned that, in most cases, should not be changed. **`@AGENT_PROTO@://@AGENT_HOST@:@AGENT_PORT@/amagent`** is an example of such a value.

▼ To Modify an Agent Profile to Inherit Properties From a Group

The Group functionality is not supported with the web services and STS Client Agent Profiles.

Before You Begin This procedure assumes you are logged into the OpenSSO Enterprise console as the administrator (by default, `amadmin`) and the group has been created. See "To Create a New Group" on page 140.

1. **Under the Access Control tab, click the name of the realm to which the agent belongs.**
2. **Click the Agents tab.**
3. **Select the tab for the appropriate agent type.**
4. **Click the name of the agent profile you want to modify.**
5. **elect the name of the group from which you want the agent to inherit properties as a value for the Group attribute under the Global tab.**
6. **Click Save.**

   At the top of the page, the Inheritance Settings button becomes active.
7. **Click Inheritance Settings.**

   A list of inheritance settings for the Global tab appears in alphabetical order.
8. **Select the properties that you want the agent to inherit from the group.**

   At the top of the page, the Inheritance Settings button becomes active.
9. **Click Save.**

Next Steps This task just describes how to change the inheritance settings for properties listed in the Global tab. For the inheritance settings of properties listed in the other tabs (such as Application and SSO), click the desired tab and edit the inheritance settings in the same manner described in the preceding steps.

PART II

Federation, Web Services, and SAML Administration

This is part two of the *Sun OpenSSO Enterprise 8.0 Administration Guide* and describes how to implement, configure and manage the OpenSSO Enterprise features based on the Liberty Alliance Project Identity Federation Framework, the Liberty Alliance Project Web Services Framework, the WS-Federation specifications, and the Security Assertion Markup Language (SAML) versions 1.x and 2. The part contains the following chapters.

- Chapter 7, “Configuring and Managing Federation”
- Chapter 8, “Federated Operations”
- Chapter 9, “Identity Web Services”
- Chapter 10, “SAML 1.x Administration”

CHAPTER 7

Configuring and Managing Federation

OpenSSO Enterprise has a federation framework that can be used to configure and manage federation configurations based on the Liberty Alliance Project Identity Federation Framework (Liberty ID-FF), the Liberty Alliance Project Web Services Framework (Liberty ID-WSF), the WS-Federation specifications and the Security Assertion Markup Language (SAML) versions 1.x and 2. Federation configurations can be implemented using the OpenSSO Enterprise console or the `ssoadm` command line utility. For a detailed overview of the Federation framework architecture, see Chapter 10, "Federation Management with OpenSSO Enterprise," in *Sun OpenSSO Enterprise 8.0 Technical Overview*. This chapter contains:

- "Configuring Federation" on page 145
- "Managing Federation Using the Console" on page 146
- "Managing Federation Using `ssoadm`" on page 155

Configuring Federation

To configure for federation, create a circle of trust and populate it with entity types using the following high-level procedure.

1. Decide whether the instance of OpenSSO Enterprise you are configuring will act as an identity provider, a service provider, or both, and create standard and extended *metadata* XML files containing the specific protocols, profiles, endpoints, and security mechanisms being used by the instance.
   - **Standard metadata** properties are defined in the Liberty ID-FF and SAMLv2 specification.
   - **Extended metadata** properties are proprietary and used by features specific to OpenSSO Enterprise.
2. Create an *entity* to hold the metadata for every identity and service provider that will become a member of the circle of trust (including the instance of OpenSSO Enterprise for which you previously created metadata).

The metadata for other entities may come from the providers themselves. See "Creating an Entity" on page 146.

3. Configure a circle of trust to denote the group of entities that have joined together to exchange authentication information for purposes of federation.

   See "Circle of Trust" on page 152.

4. Add the appropriate entities to the circle of trust by configuring both the entity's metadata (to add the authentication domain of the circle of trust) and the circle of trust's properties (to add the entity).

Information on an entity provider's properties are located in Chapter 6, "Federation Attributes for Entity Providers," in *Sun OpenSSO Enterprise 8.0 Administration Reference*. Information on a circle of trust's properties can be found in "To Modify a Circle of Trust Profile" on page 153.

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.

Managing Federation Using the Console

The Federation component of the OpenSSO Enterprise console provides an interface for creating, modifying, and deleting circles of trust, and the corresponding member entity providers (both identity and service).

- "Creating an Entity" on page 146
- "Circle of Trust" on page 152

Creating an Entity

An *entity* holds the metadata for individual identity and service providers. (Metadata contains the specific protocols, profiles, endpoints, and security mechanisms being used by the entity.) OpenSSO Enterprise allows you create an entity for communication using either the SAML v2, the Liberty ID-FF, and the WS-Federation specifications. Within each entity type, you can assign *roles* by configuring the attributes to perform the specific function. The following sections describe the entity types and the roles you can assign.

- "SAMLv2 Entity" on page 147
- "SAMLv2 Hosted Affiliation Customization" on page 148
- "ID-FF Entity Provider" on page 149
- "WS-Federation Entity Provider" on page 151

SAMLv2 Entity

The SAMLv2 entity type is based on the SAML v2 specification. This entity supports various profiles (including single sign-on and single logout) and allows you to assign and configure the following roles:

- Identity Provider
- Service Provider
- XACML PEP
- XACML PDP
- Attribute Authority
- Attribute Query
- Authentication Authority
- Hosted Affiliation

▼ To Create a SAMLv2 Entity Provider

Use these steps to create a hosted entity provider based on the SAMLv2 protocol. You can assign one, more than one, or all of the provider roles to the entity, but all of the roles that you define will belong to the same entity provider.

1 **Log in as an administrator.**

2 **Go to the Federation tab in the console and click New in the Entity Provider table.**

3 **When prompted, select SAMLv2 as the entity provider.**

4 **Select the Realm to which the entity provider will belong.**

5 **Type a name in the Entity Identifier field.**

6 **Enter values for the following attributes under the role category to which the entity provider will be assigned.**

Entering data in the Meta Alias field will automatically create and assign the entity provider role to the entity provider upon completion.

Meta Alias	Specifies a `metaAlias` for the provider role 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 coupled with a forward slash and the provider name. For example, `/suncorp/travelprovider`.

Caution – The names used in the `metaAlias` must not contain a `/`.

Signing Certificate Alias	Specifies the provider certificate alias used to find the correct signing certificate in the keystore.
Encryption Certificate alias	Specifies the provider certificate alias used to find the correct encryption certificate in the keystore.
Owner ID (Hosted Affiliation only)	An identifier for the owner of the affiliation.
Affiliation Members (Hosted Affiliation only)	A provider must be a member of a circle of trust, or it cannot participate in SAMLv2-based communications. The provider can belong to one or more affiliations. The selected provider must have the Affiliation Federation attribute enabled. Enter the meta alias of the provider in the New Value field and click Add.

7 **Click Create.**

The entity provider, its assigned provider roles, and location will be displayed in the Entity Providers table. To customize the entity providers' roles behavior, click the name of the entity provider from the list and choose the tab that corresponds to the role you wish to customize. See Chapter 6, "Federation Attributes for Entity Providers," in *Sun OpenSSO Enterprise 8.0 Administration Reference* for definitions attributes for provider customization.

SAMLv2 Hosted Affiliation Customization

A Hosted Affiliation contains a grouping of service providers. The affiliation is formed and maintained by an affiliation owner who chooses the member providers from already configured provider entities. 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.

A hosted affiliation provider 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. 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. Hosted Affiliation contains the following attributes for customization:

- "Meta Alias" on page 149
- "Members" on page 149
- "Cert Alias" on page 149

Meta Alias

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 OpenSSO Enterprise) coupled with a forward slash and the provider name. For example, `/suncorp/travelprovider`.

Caution – The names used in the `metaAlias` must not contain a `/`.

Members

A provider must be a member of a circle of trust, or it cannot participate in Liberty-based communications. The provider can belong to one or more affiliations. Enter the entity ID of the provider in the New Value field and click Add.

Cert Alias

This attribute defines the certificate alias elements for the provider. `Signing` specifies the provider certificate alias used to find the correct signing certificate in the keystore. `Encryption` specifies the provider certificate alias used to find the correct encryption certificate in the keystore.

ID-FF Entity Provider

The ID-FF provider entity is based on the Liberty Alliance Project Identity Federation Framework for implementing single sign-on with federated identities. The ID-FF provider entity allows you to assign and configure the following roles:

- Identity Provider
- Service Provider

▼ To Create an ID-FF Entity Provider

Use these steps to create an entity provider based on the ID-FF protocol for Federation Services. You can assign the identity provider or service provider (or both) role to the entity, but multiple roles will belong to the same entity provider.

1. **Log in as an administrator.**
2. **Go to the Federation tab in the console and click New in the Entity Provider table.**
3. **When prompted, select ID-FF as the entity provider.**
4. **Select the Realm to which the entity provider will belong.**
5. **Type a name in the Entity Identifier field.**
6. **Choose the entity provider role you wish to assign to the entity provider.**

   Entering data in the Meta Alias field will automatically create and assign the entity provider role to the entity provider upon completion.
7. **Enter values for the following attributes for one or more roles:**

Meta Alias	Specifies a `metaAlias` for the provider role 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 coupled with a forward slash and the provider name. For example, `/suncorp/travelprovider`. **Caution –** The names used in the `metaAlias` must not contain a `/`.
Signing Certificate Alias	Specifies the provider certificate alias used to find the correct signing certificate in the keystore.
Encryption Certificate alias	Specifies the provider certificate alias used to find the correct encryption certificate in the keystore.

8. **Click Create.**

   The entity provider, its assigned provider roles, and location will be displayed in the Entity Providers list.

9 **To customize the entity providers' roles behavior, click on the name of the entity provider and choose the tab that corresponds to the role you wish to customize. See Chapter 6, "Federation Attributes for Entity Providers," in *Sun OpenSSO Enterprise 8.0 Administration Reference* for definitions attributes for provider customization.**

WS-Federation Entity Provider

The WS-Federation entity provider type is based on the WS-Federation protocol. The implementation of this protocol allows single sign-on between OpenSSO Enterprise and the Microsoft Active Directory Federation Service. The WS-Federation provider entity allows you to assign and configure the following roles:

- Identity Provider
- Service Provider

▼ To Create a WS-Federation Entity Provider

Use these steps to create to create an entity provider based on the WS-Federation protocol for Federation Services. You can assign the identity provider or service provider (or both) role to the entity, but multiple roles will belong to the same entity provider.

1 **Log in as an administrator.**

2 **Go to the Federation tab in the console and click New in the Entity Provider table.**

3 **When prompted, select WS-FED as the entity provider.**

4 **Select the Realm to which the entity provider will belong.**

5 **Type a name in the Entity Identifier field.**

6 **Choose the entity provider role you wish to assign to the entity provider.**

Entering data in the Meta Alias field will automatically create and assign the entity provider role to the entity provider upon completion.

7 **Enter values for the following attributes for one or more roles:**

Meta Alias	Specifies a `metaAlias` for the provider role 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 coupled with a forward slash and the provider name. For example, `/suncorp/travelprovider`.

Caution – The names used in the `metaAlias` must not contain a `/`.

Signing Certificate Alias	Specifies the provider certificate alias used to find the correct signing certificate in the keystore.
Encryption Certificate alias	Specifies the provider certificate alias used to find the correct encryption certificate in the keystore.

8 **Click Create.**

The entity provider, its assigned provider roles, and location will be displayed in the Entity Providers list.

9 **To customize the entity providers' roles behavior, click on the name of the entity provider and choose the tab that corresponds to the role you wish to customize. See Chapter 6, "Federation Attributes for Entity Providers," in *Sun OpenSSO Enterprise 8.0 Administration Reference* for definitions attributes for provider customization.**

Circle of Trust

A *circle of trust*, previously referred to as 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. To create and populate a circle of trust, you first create an *entity* to hold the *metadata* (configuration information that defines a particular identity service architecture) for each provider that will become a member of the circle of trust. Then, you configure and save the circle of trust. Finally, to add an entity (a configured provider) to the circle of trust, you edit the entity's properties.

The following tasks are associated with circles of trust:

- "To Create a New Circle of Trust" on page 152
- "To Modify a Circle of Trust Profile" on page 153
- "To Add Providers to a Circle of Trust" on page 154
- "To Delete a Circle of Trust Profile" on page 155

▼ To Create a New Circle of Trust

Follow this procedure to create a new circle of trust. The starting point is New Circle of Trust under the Federation interface.

1 **Click New to display the circle of trust attributes.**

The New circle of trust profile page is displayed.

2 **Type a name for the circle of trust.**

3 **Type a description of the circle of trust in the Description field.**

4 **Type a value for the IDFF Writer Service URL.**

The IDFF Writer Service URL specifies the location of the servlet that writes the common domain cookie. Use the format `http://`*common-domain-host* `:`*port*`/deployment_uri/idffwriter`.

5 **Type a value for the IDFF Reader Service URL.**

The IDFF Reader Service URL specifies the location of the servlet that reads the common domain cookie. Use the format `http://`*common-domain-host* `:`*port*`/deployment_uri/idffreader`.

6 **Type a value for the SAML2 Writer Service URL.**

This specifies the location of the SAML2 Writer service that writes the cookie to the common domain. Use the format `http://`*common-domain-host* `:`*port*`/deployment_uri/saml2writer`.

7 **Type a value for the SAML2 Reader Service URL.**

This specifies the location of the SAML2 Reader service that reads the cookie from the common domain. Use the format `http://`*common-domain-host* `:`*port*`/deployment_uri/saml2reader`.

8 **Choose Active or Inactive.**

The default status is Active. Choosing Inactive disables communication within the circle of trust.

9 **Select the Realm in which the circle of trust will be created.**

10 **Choose one or more of the available providers and click the Add arrow to select them.**

The list provided contains the names of entities that have been created and populated with providers. For more information, see "To Add Providers to a Circle of Trust" on page 154.

11 **Click OK to complete the configuration.**

The new circle of trust is displayed in the Circle of Trust list.

▼ To Modify a Circle of Trust Profile

Follow this procedure to edit the configured General attributes of an existing circle of trust, or to add providers to it. The starting point is Circle of Trust under the Federation interface.

1 **Click the name of a configured circle of trust to modify its profile, or to add providers to it.**

The Edit Circle of Trust page is displayed.

2 **Type new values or edit existing values for the circle of trust's General attributes:**

Name	The static value of this attribute is the name provided when you created the circle of trust.
Description	The value of this attribute is a description of the circle of trust. You may modify the description already entered, if applicable.
IDFF Writer Service URL	This attribute specifies the location of the service that writes the common domain cookie. The URL is in the format **`http://`***common-domain-host***`:`***port***`/deployment_uri/idffwriter`** .
IDFF Reader Service URL	This attribute specifies the location of the service that reads the common domain cookie. The URL is in the format **`http://`***common-domain-host***`:`***port***`/deployment_uri//idffreader`** .
SAML2 Writer URL	This attribute specifies the location of the SAML2 Writer service that writes the cookie to the Common Domain. The URL is in the format **`http://`***common-domain-host***`:`***port***`/deployment_uri/saml2writer`**
SAML2 Reader URL	This attribute specifies the location of the SAML2 Writer service that writes the cookie to the Common Domain. The URL is in the format **`http://`***common-domain-host***`:`***port***`/deployment_uri/saml2reader`**
Status	The default status is Active. Selecting Inactive disables communication within the circle of trust.

3 **Choose one or more of the available providers and click the Add arrow to select them.**

The list provided contains the names of entities that have been created and populated with providers. For more information, see "To Add Providers to a Circle of Trust" on page 154.

4 **Click Save to complete the operation.**

▼ To Add Providers to a Circle of Trust

Identity providers and service providers must first be configured within an *entity* before they are available to add to a circle of trust. Once created and populated with providers, the entity (and thus the providers it contains) can be assigned to a circle of trust.

Note – An entity will not be visible in the Available Providers list until it has been populated with providers.

---

1 **Select one or more providers from the Available Providers list and click Add.**

2 **Finish your configurations and click Save to complete the operation.**

▼ To Delete a Circle of Trust Profile

A circle of trust must be empty of providers before you delete it. Follow this procedure to delete an existing circle of trust.

1 **Check the box next to the name of the circle of trust you want to delete.**

2 **Click Delete.**

Deleting a circle of trust does not delete the providers that belong to it.

Managing Federation Using `ssoadm`

The previous sections detailed how to create and configure entities and circles of trust using the OpenSSO Enterprise console. But entities can also be created and configured using the `ssoadm` command-line interface. 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 `ssoadm`.

Caution – The format of the XML file used as input is based on the `sms.dtd`. Alterations to the DTD files may hinder the operation of OpenSSO Enterprise.

This section contains the following information:

- "Managing Entity Metadata using `ssoadm`" on page 155
- "Managing Circles of Trust Using `ssoadm`" on page 160

Managing Entity Metadata using `ssoadm`

`ssoadm` is used to manage the provider metadata. The following table describes the `ssoadm` subcommands specific to metadata management.

TABLE 7–1 ssoadm Subcommands for Managing Metadata

Subcommand	Description
`import-entity`	Loads standard and extended metadata in XML format into a local configuration data store. **Note –** Use the –spec option to specify `saml2`, `idff`, or `wsfed`.
`export-entity`	Exports standard and extended metadata in XML format from a local configuration data store. **Note –** Use the –spec option to specify `saml2`, `idff`, or `wsfed`.
`create-meadata-templ`	Generates a metadata configuration file for any provider type with defined values for default metadata properties. The generated file can be modified for use with `import-entity`. **Note –** Use the –spec option to specify `saml2`, `idff`, or `wsfed`.
`delet-entity`	Removes standard or extended metadata from a local configuration data store. **Note –** Use the –spec option to specify `saml2`, `idff`, or `wsfed`.
`list-entities`	Generates a list of all the entity identifiers on the system. **Note –** Use the –spec option to specify `saml2`, `idff`, or `wsfed`.
`update-entity-key-info`	Update XML signing and encryption key information for a hosted IDP or SP.

There are two types of entity provider metadata (formatted in XML files) that can be used as input to `ssoadm`:

- **Standard metadata** properties are defined in the Liberty ID-FF and SAMLv2 specification.
- **Extended metadata** properties are proprietary and used by features specific to OpenSSO Enterprise.

Information regarding the attributes and possible values of the metadata can be found in Chapter 6, "Federation Attributes for Entity Providers," in *Sun OpenSSO Enterprise 8.0 Administration Reference*. The following sections contain information on loading the metadata.

- "Loading Standard Metadata Using `ssoadm`" on page 156
- "Loading Extended Metadata Using `ssoadm`" on page 158

Loading Standard Metadata Using `ssoadm`

To load metadata compliant with the Liberty ID-FF, SAMLv2, or WS-Federation protocols, use the following command (options in square brackets are optional):

```
ssoadm import-entity --amadmin admin-ID
 --password-file password_filename [--realm]
realm-name[--metadata-file] metadatafilename [--cot] circle_of-trust [--spec] idff_or_saml2_or_wsfed_or_wsfed
```

This option is usually used to load provider metadata sent from a trusted partner in an XML file 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

```
<!--
  Copyright (c) 2005 Sun Microsystems, Inc. All rights reserved
  Use is subject to license terms.
-->

<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
```

EXAMPLE 7–1 Service Provider Standard Metadata XML File *(Continued)*

```
    </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>
   <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 Extended Metadata Using `ssoadm`

OpenSSO Enterprise provides proprietary attributes that are not a specific part of the Liberty ID-FF, WS-Federation, or SAMLv2 protocols. To load OpenSSO Enterprise proprietary metadata use the following command:

```
ssoadm import-entity --amadmin admin-ID --password-file
password_filename [--realm realm-name] [--meta-data-file
 metadatafilename] [--extended-data-file extended_metadata_filename] [--cot circle_of-trust] [--spec]idff_or_saml2_or-wsfed]
```

After loading the metadata, the ssoadm export-entity option can be used to export metadata. 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 Extended Metadata XML File

```
<?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">
      <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>
```

Managing Circles of Trust Using `ssoadm`

The `ssoadm` command line interface creates and manages the circles of trust used by the Federation services. The following table describes the `ssoadm` subcommands specific to circle of trust management.

TABLE 7–2 `ssoadm` Subcommands for Managing Circles of Trust

Subcommand	Description
`create-cot`	Creates a circle of trust.
`delete-cot`	Removes a circle of trust. **Note –** To delete a circle of trust that contains providers, use `remove-cot-members` to remove each provider first, then use `delete-cot` to delete the circle itself.
`add-cot-member`	Adds a trusted provider to an existing circle of trust. **Note –** `add-cot-member` can only add a single entity at a time. Add multiple entities when you first create the circle by using `create-cot` and the `---trustedproviders` option.
`remove-cot-member`	Removes a trusted provider from an existing circle of trust.
`list-cot-members`	Lists the member providers in a particular circle of trust.
`list-cots`	Lists all the circles of trust configured on the system.

The following command example will create a circle of trust:

```
ssoadm create-cot --cot COT-name --adminid
admin-user --password-file password-filename
[--realm realm-name] [--trustedproviders
trusted-providers] [--prefix idp-discovery-URL-prefix]
```

This second command example will add a trusted provider to an existing circle of trust:

```
ssoadm add-cot-member --cot COT-name --enitityid
entitiy_ID --adminid admin-user --password-file
password [--realm realm-name]
[--spec saml2-or-idff]
```

This next command example will remove a trusted provider from an existing circle of trust:

```
ssoadm remove-cot-member --cot COT-name --enitityid
entitiy_ID --adminid admin-user --password-file
password [--realm realm-name]
[--spec saml2-or-idff]
```

This command example will list all the providers belonging to an existing circle of trust:

```
ssoadm list-cot-members --cot COT-name --adminid admin-user
--password-file password [--realm realm-name]
 [--spec saml2-or-idff]
```

This command example will list all the available circles of trust:

```
ssoadm list-cots  --adminid admin-user --password-file password
[--realm realm-name]
```

CHAPTER 8

Federated Operations

This chapter contains procedures illustrating how to use OpenSSO Enterprise to configure interactions based on the ID-FF and SAMLv2 protocols. The sections contained in this chapter are:

- "Finding an Identity Provider for Authentication" on page 163
- "Bulk Federation" on page 167
- "ID-FF Federation Operations" on page 168
- "SAMLv2 Operations" on page 179
- "WS-Federation Operations" on page 218

Finding an Identity Provider for Authentication

If there is only one Identity Provider in a Circle-of-Trust, Service Providers will send users directly to the Identity Provider for authentication. In the case when there are multiple Identity Providers in a Circle-of-trust, a Service Provider requires a way to determine which identity provider is used by a principal requesting authentication. Because Service Providers are configured without regard to their location, this function must work across DNS-defined domains. OpenSSO Enterprise implements the following solutions for this use case:

- Identity Provider Discovery Service in SAMLv2
- Identity Provider Introduction Service in ID-FF
- Home Realm Discovery Service in WS-Federation

When these services are configured, the Service Provider determines and redirects the user agent to the appropriate identity provider for authentication. The following sections contain more information.

- "Configuring the SAMLv2 Identity Provider Discovery Service" on page 164
- "Configuring the ID-FF Identity Provider Introduction Service" on page 164
- "Configuring WS-Federation Home Realm Discovery Service" on page 165
- "Customizing SAMLv2 the Identity Provider Discovery Service and the ID-FF Identity Provider Introduction Service" on page 166

Configuring the SAMLv2 Identity Provider Discovery Service

The SAMLv2 Identity Provider Discovery Service is provided by OpenSSO Enterprise after deployment. Alternatively, the Identity Provider Discovery Service can be configured as a standalone service. After the SAMLv2 Identity Provider Discovery Service is configured, an administrator creates and configures a Circle-of-Trust to use the Identity Provider Discovery service for the IDPs and SPs. In OpenSSO Enterprise, the Identity Provider Discovery Service for SAMLv2 providers is configured using two URLs that point to servlets developed for writing and reading a special cookie called Common Domain cookie. Go to the circle-of-trust entity and configure the following:

- "SAMLv2 Writer Service URL" on page 164
- "SAMLv2 Reader Service URL" on page 164

SAMLv2 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 `_saml_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 SAML2 Writer Service URL attribute when a circle of trust is created. Use the format `http://`*idp-discovery-host:port*`/`*deployment-uri*`/writer` where *idp-discovery-host:port* refers to the machine on which the SAMLv2 Identity Provider Discovery service is installed and *deployment-uri* tells the web container where to look for information specific to the application (such as classes or JARs).

SAMLv2 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 a circle of trust is created. It is formatted as `http://`*idp-discovery-host:port*`/`*deployment-uri*`/transfer` where *idp-discovery-host:port* refers to the machine on which the SAMLv2 IDP Discovery service is installed and *deployment-uri* tells the web container where to look for information specific to the application (such as classes or JARs).

Configuring the ID-FF Identity Provider Introduction Service

OpenSSO Enterprise provides the Liberty ID-FF Identity Provider Introduction Service upon deployment. Alternatively, the Identity Provider Introduction Service could be configured as a standalone service (refer to later section for details).

After the Liberty ID-FF Identity Provider Introduction Service is configured, an administrator needs create and configure a Circle-of-Trust to use the service. In OpenSSO Enterprise, the Identity Provider Introduction Service for ID-FF providers is configured using two URLs that point to servlets developed for writing and reading a special cookie called Common Domain cookie. Go to the circle-of-trust entity and configure the following:

- "ID–FF Writer Service URL" on page 165
- "ID-FF Reader Service URL" on page 165

ID–FF 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 ID-FF Writer Service URL defined for the identity provider. The URL is configured as the value for the ID-FF Writer Service URL attribute when a circle of trust is created. Use the format `http://`*idp-introduction-host:port*`/`*deployment-uri*`/idffwriter` where *idp-introduction-host:port* refers to the machine on which the ID-FF Identity Provider Introduction service is installed and *deployment-uri* tells the web container where to look for information specific to the application (such as classes or JARs).

ID-FF Reader Service URL

The ID-FF 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 ID-FF Reader Service URL attribute when a circle of trust is created. It is formatted as `http://`*idp-introduction-host:port*`/`*deployment-uri*`/transfer` where *idp-intorductoin-host:port* refers to the machine on which the ID-FF Identity Provider Introduction service is installed and *deployment-uri* tells the web container where to look for information specific to the application (such as classes or JARs).

Configuring WS-Federation Home Realm Discovery Service

To configure a WS-Federation service provider to use the Home Realm Discovery Service, click on the WS-Federation entity name in the OpenSSO Enterprise console, select the Service Provider (SP) tab and configure the following:

Home Realm Discovery	Specifies the service so that the service provider can identify the preferred identity provider. The service URL is specified as a contact endpoint by the service provider.

Account Realm Selection	Specifies the identity provider selection mechanism and configuration. Either the cookie or HTTP Request header attribute can be used to locate the identity provider.

Customizing SAMLv2 the Identity Provider Discovery Service and the ID-FF Identity Provider Introduction Service

There are two ways to obtain the SAMLv2 IDP Discovery Service/ID-FF IDP Introduction service:

1. Create and deploy a specialized WAR file used for the SAMLv2 Identity Provider Discovery Service and ID-FF Identity Provider Introduction Service only. See "To Create a Specialized WAR file for the Identity Provider Services" on page 166.
2. Customize the SAMLv2 Identity Provider Discovery Service and ID-FF Identity Provider Introduction Service through the console. See "To Customize the Identity Provider Services Through the Console" on page 167.

▼ To Create a Specialized WAR file for the Identity Provider Services

OpenSSO Enterprise provides a mechanism to create a specialized WAR file for the SAMLv2 Identity Provider Discovery Service and the ID-FF Identity Provider Introduction Service. The WAR file can be deployed as standalone application, independent of Identity Provider and Service Provider domains. See "Creating and Deploying Specialized OpenSSO Enterprise WAR Files" in *Sun OpenSSO Enterprise 8.0 Installation and Configuration Guide*.

1 **After you deploy and run the Configurator for the specialized WAR file, locate the configuration property file named** `libIDPDiscoveryConfig.properties`**.**

This file is created under the web container user's home directory. This file is the same for both the SAMLv2 IDP Discovery service and the ID-FF IDP Introduction service.

2 **Customize the following properties to meet your specific deployment needs:**

`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.

▼ To Customize the Identity Provider Services Through the Console

1 **Login to the console as top level administrator.**

2 **Click the Configuration tab.**

3 **Click the Global sub-configuration tab.**

4 **Select the SAMLv2 Service Configuration service.**

5 **Customize the following attributes. These attributes are applicable for both the SAMLv2 Identity Provider Discovery Service and ID-FF Identity Provider Introduction Service:**

Cookie Domain for IDP Discovery Service	Specifies the cookie domain for the SAMLv2 IDP discovery cookie.
Cookie Type for IDP Discovery Service	Specifies cookie type used in SAMLv2 IDP Discovery Service, either Persistent or Session. Default is Session.
URL Scheme for IDP Discovery Service	Specifies URL scheme used in SAMLv2 IDP Discovery Service.

Bulk Federation

OpenSSO Enterprise handles the federation of user accounts in bulk through the `ssoadm` command line tool. From the service provider, import the metadata to create the bulk federation data. The full command is as follows:

```
ssoadm do-bulk-fed-data --metaalias meta_alias --remote-entity-id
entity_ID--useridmapping id-mapping-filename --name-id-mapping
created_nameid_filename --adminid administrator_ID
 --password-file password_filename --spec idff_or_saml2_or_wsfed
```

As input, the command takes a file that maps the user distinguished name (DN) of the identity provider to the user DN of the service provider. You specify this in the –`useridmapping` option of the `do-bulk-federation` subcommand. The format is the full DN of `local-user-id|remote-user-id`. For example:

```
id=spuser1,ou=user,dc=red,dc=iplanet,dc=com|id=idpuser1,
ou=user,dc=red,dc=iplanet,dc=com
id=spuser2,ou=user,dc=red,dc=iplanet,dc=com|id=idpuser2,
ou=user,dc=red,dc=iplanet,dc=com
id=spuser3,ou=user,dc=red,dc=iplanet,dc=com|id=idpuser3,
ou=user,dc=red,dc=iplanet,dc=com
id=spuser4,ou=user,dc=red,dc=iplanet,dc=com|id=idpuser4,
ou=user,dc=red,dc=iplanet,dc=com
```

Note – The users defined in this file must already exist in the identity provider and service provider.

To load the bulk data into an identity provider, use the following command. The bulk federation file created by the `do-bulk-federation` subcommand is specified in the `bulk-data-file` option:

```
ssoadm import-bulk-fed-data --meta-aslias meta_alias --bulk-data-file
bulk_federation_filename --adminid administrator_ID --password-
file password_filename --spec idff_or_saml2_or_wsfed
```

ID-FF Federation Operations

This section describes Federation operations that are specific to the ID-FF protocol.

- "The Pre-Login URL" on page 168
- "Configuring ID-FF Single Sign-on" on page 171
- "ID-FF Auto-Federation" on page 172
- "Enabling ID-FF XML Signing" on page 174
- "Dynamic Identity Provider Proxying" on page 175

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

OpenSSO Enterprise session token is found. This default process, though, can be modified based on the values of query parameters passed to OpenSSO Enterprise by the service provider within 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 8–1 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 OpenSSO Enterprise 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.
`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.

TABLE 8–1 Pre-login URL Parameters for Federation *(Continued)*

Parameter	Description
`authlevel`	The `authlevel` parameter takes as a value a positive number that maps to an authentication level defined in the OpenSSO Enterprise 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 OpenSSO Enterprise 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, dependent on your deployment.

▼ To Configure for Pre-login

In a federation setup, OpenSSO Enterprise 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 OpenSSO Enterprise.**

For example: `com.sun.am.policy.loginURL = http://www.sp1.com:58080/opensso/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 OpenSSO Enterprise acting as the service provider.**

For example: `com.sun.am.policy.am.library.loginURL = http://www.sp1.com:58080/opensso/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/opensso/liberty-logout?metaAlias=www.sp1.com`.

Configuring ID-FF Single Sign-on

To setup single sign-on between two OpenSSO Enterprise instances for ID-FF protocol, one as an identity provider and one as a service provider, follow these steps:

▼ To Configure ID-FF Single Sign-on

1. **Create an ID-FF identity provider. For instructions, see "To Create an ID-FF Entity Provider" on page 150.**

2. **Export the standard identity provider metadata into an XML file using the `ssoadm` command's `export-entity` subcommand. The example filename for these instructions is `IDP.xml`.**

3. **Create an ID-FF service provider. For instructions, see "To Create an ID-FF Entity Provider" on page 150**

   Note – If the identity provider and service provider reside in the same domain, you need to modify the cookie name of one instance to be different from the other. To do so, log in to the OpenSSO Enterprise and go to Configuration>Servers and Sites, then choose the server instance. Click the Security>Inheritance Settings, and uncheck the Cookie Name field. Click Save.

   Click Back to Server Profile, go to the Cookie section, and modify the value for Cookie Name. Click Save. Restart the web container.

   ---

4. **Export the standard identity provider metadata into an XML file using the `ssoadm` command's `export-entity` subcommand. The example filename for these instructions is `SP.xml`.**

5 **Load the remote service provider metadata to the OpenSSO Enterprise instance of the identity provider. To do so:**

   a. **Copy the `SP.xml` file to the identity provider instance.**

   b. **Log in to the console on the identity provider instance and click on the Federation tab.**

   c. **Click the Import Entity button.**

   d. **Choose the realm to which the identity provider resides.**

   e. **Select the File option and click upload to locate the `SP.xml` file. You can leave the extended metadata field blank.**

6 **In the Federation tab, create a circle of trust and add the identity provider and service provider. For instructions, see "To Create a New Circle of Trust" on page 152.**

7 **Load the identity provider metadata to the OpenSSO Enterprise instance of the service provider. To do so:**

   a. **Copy the `IDP.xml` file to the identity provider instance.**

   b. **Log in to the console on the identity provider instance and click on the Federation tab.**

   c. **Click the Import Entity button.**

   d. **Choose the realm to which the identity provider resides.**

   e. **Select the File option and click upload to locate the `IDP.xml` file. You can leave the extended metadata field blank.**

8 **In the Federation tab, create a circle of trust and add the identity provider and service provider. For instructions, see "To Create a New Circle of Trust" on page 152.**

Single Sign-on is now established between the OpenSSO Enterprise instances.

ID-FF Auto-Federation

The auto-federation feature in OpenSSO Enterprise 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 on the service provider side (that is, without login on the service provider side).

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 ID-FF Auto Federation

Ensure that single sign-on is properly configured. For more information, see "To Configure ID-FF Single Sign-on" on page 171. Remote providers would not be configured in your deployment.

1. **In the OpenSSO Enterprise Console, click the Federation tab.**
2. **Select the name of the identity provider to edit its profile.**
3. **Click on the Auto Federation link at the top of the page, or scroll down to the Auto Federation subsection.**
4. **Enable Auto Federation by checking the box.**
5. **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.
6. **Click the Identity Provider Attribute Mapper link, or scroll down to the Identity Provider Attribute Mapper subsection. Enter the following attribute values:**

Attribute Mapper Class	`com.sun.identity.federation.services.FSDefaultRealmA`
Identity Provider Attribute Mapping	Enter the mapping for the Auto-Federation attribute name.
7. **Click the Plug-ins link, or scroll down to the Plug-ins subsection. Enter the following attribute value:**

Attribute Statement Plug-in	`com.sun.identity.federation.services.FSDefaultRealmAttribut`
8. **Click Save to complete the identity provider configuration.**
9. **Go back to the Federation tab and select the service provider you wish to edit.**
10. **Click on the Auto Federation link at the top of the page, or scroll down to the Auto Federation subsection.**

11 **Enable Auto Federation by checking the box.**

12 **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.

13 **Click the Service Provider Attribute Mapper link, or scroll down to the Service Provider Attribute Mapper subsection. Enter the following attribute values:**

Attribute Mapper Class	`com.sun.identity.federation.services.FSDefaultRealmAttrib`
Identity Provider Attribute Mapping	Enter the mapping for the Auto-Federation attribute name.

14 **Click Save to complete the configuration.**

Enabling ID-FF XML Signing

By default, ID-FF singing is turned off. To enable ID-FF XML signing on OpenSSO Enterprise server, see the following section:

▼ To Enable ID-FF XML Signing

1 **Login to the console as the top-level administrator.**

2 **Select the Configuration tab, select Global, and then Liberty ID-FF Service Configuration.**

3 **Select YES for the XML Signing On attribute and save the configuration.**

4 **Go to the Federation tab and select the service provider and/or identity provider that needs to be enabled.**

5 **Under the Common Attributes section, make sure a signing certificate alias is chosen for the provider. Otherwise, you must enter your certification alias.**

If the certificate alias is added or changed, you need to send the new metadata (to be exported using `ssoadm` CLI) to the remote party to update its metadata.

6 **Click Save.**

7 **Restart the server.**

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 (referred as Proxying 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. After receiving the Assertion from the authenticating identity provider, the proxying identity provider generates a new Assertion based on the information from the original Assertion, and sends to the requesting service 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 OpenSSO Enterprise 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

1 **Install and configure three OpenSSO instances.**

If those three instances resides on the same domain, you need to modify the cookie name of all three instances to be different from one another. To do so:

a. **Login to administration console and click the Configuration tab.**

b. **Click Servers and Sites and choose your server instance.**

c. **Click the Security tab, then click Inheritance Settings.**

d. **Uncheck the box for the Cookie Name attribute.**

e. **Click Save.**

f. **Click the Back to Server Profile button.**

g. **Modify the value for the Cookie Name attribute under the Cookie section.**

h. **Click Save.**

i. **Restart the web container on which the server instance is deployed.**

2 **Create hosted ID-FF metadata on the requesting service provider instance on machine 3.**

For more information, see "ID-FF Entity Provider" on page 149. When creating the provider:

a. **Enter the service provider entity ID in the Entity Identifier field**

b. **Under the Service Provider section, specify meta alias, signing/encryption cert alias if needed.**

3 **Export the requesting Service Provider's standard metadata to a XML file. This could be done using the `export-entity` option in ssoadm CLI or the `ssoadm.jsp`. The rest of these steps use the file name `sp.xml` as an example.**

4 **Create hosted metadata on the proxying Identity provider instance, o machine 2.**

a. **Login to machine 2 as the top-level administrator, click Federation, then click New under Entity Providers table.**

b. **Select IDFF in the pop up window**

c. **Save the configuration.**

d. **Enter the entity ID in the Entity Identifier field.**

e. **Under the Identity Provider section, specify a meta alias, and enter signing/encryption cert alias if needed.**

f. **Under the Service Provider section, specify a meta alias different from that entered for Identity Provider section, and enter signing/encryption cert alias if needed.**

g. **Click create.**

5 **Export the proxying identity provider's standard metadata to a XML file. This is done using the `export-entity` option in `ssoadm` CLI or `ssoadm.jsp`. The rest of these steps use the file name `proxy.xml` as an example.**

6 **Create the hosted metadata on the authenticating Identity provider instance, machine 1.**

For more information, see "ID-FF Entity Provider" on page 149. When creating the provider:

a. **Enter the service provider entity ID in the Entity Identifier field**

b. **Under the Service Provider section, specify meta alias, signing/encryption cert alias if needed.**

7 **Export the authenticating Identity provider's standard metadata to a XML file using the `export-entity` option in the `ssoadm` CLI or by using the `ssoadm.jsp`. The rest of these steps use the file name `idp.xml` as an example.**

8 **Load the remote proxying identity provider metadata on the requesting service provider instance.**

a. **Copy the `proxy.xml` to machine 3.**

b. **In the console of machine 3, click the Federation tab and then the Import Entity button.**

c. **Choose the realm to which the requesting service provider belongs.**

d. **Under Where Does the Meta Data File Reside field, choose File and click Upload.**

e. **Choose `proxy.xml`. The Where Does the Extended Data File Reside can be left blank.**

f. **Click Ok.**

9 **Create a circle of trust on the requesting service provider instance to include the proxying IDP and requesting SP entity. For information, see "Circle of Trust" on page 152.**

10 **Load the remote authenticating identity provider and requesting service provider metadata to the proxying identity provider instance.**

a. **Copy the `idp.xml` and `sp.xml` files to machine 2.**

b. **In the console of machine 2, click the Federation tab and then the Import Entity button.**

c. **Choose the realm to which the requesting service provider belongs.**

d. **Under Where Does the Meta Data File Reside field, choose File and click Upload.**

e. **Choose `idp.xml`. The Where Does the Extended Data File Reside field can be left blank.**

f. **Click OK.**

g. **Repeat the steps for loading the requesting service provider meta data (`sp.xml`).**

11 **Create circles of trust on the proxying identity provider instance, machine 2. For information, see "Circle of Trust" on page 152**

Add the requesting service provider and proxying identity provider to the circle of trust. Repeat this step to create a circle of trust for both the authenticating identity provider and the proxying identity provider. Make sure the circles of trust have different names.

12 **Load the remote proxying identity provider metadata on the authenticating identity provider instance, machine 1.**

a. **Copy the `proxy.xml` to machine 1.**

b. **In the console of machine 1, click the Federation tab and then the Import Entity button.**

c. **Choose the realm to which the requesting service provider belongs.**

d. **Under Where Does the Extended Data File Reside field, choose File and click Upload.**

e. **Choose `proxy.xml`. The Where Does the Extended Data File Reside can be left blank.**

f. **Click Ok.**

13 **Create a create a circle of trust on the authenticating identity provider instance to include the proxying identity provider and authenticating identity provider entities.**

14 **Enable Dynamic Identity provider proxying on the requesting service provider instance.**

a. **In the console of machine 3, click the Federation tab.**

b. **Select the hosted requesting service provider.**

c. **Go to Proxy Authentication Configuration.**

d. **Check the box for Proxy Authentication to enable it.**

e. **Enter 10 for the value in the Maximum Number of Proxies attribute.**

f. **Click Save.**

15 **Enable dynamic identity provider proxying on the proxying identity provider instance.**

a. **In the console of machine 2, click the Federation tab.**

b. Select the hosted proxying identity provider.

c. Go to Proxy Authentication Configuration.

d. Check the box for Proxy Authentication to enable it.

e. Enter 10 for the value in the Maximum Number of Proxies attribute.

f. Add the authenticating identity provider's entity ID in the Proxy Identity Providers List file.

g. Click Save.

16 Federate a user between machine 3 (acting as a service provider) and machine 2 (acting as an identity provider).

17 Federate a user between machine 2 (acting as a service provider) and machine 1 (acting as an identity provider).

18 Close the browser and attempt to perform a single sign-on again from machine 3. You will be redirected to machine 1 rather than machine 2 for authentication.

Enter the username and password used on machine 1. You will have a successful single sign-on to machine 3.

SAMLv2 Operations

The SAMLv2 specification defines the assertion security token format as well as profiles that standardize the HTTP exchanges required to transfer XML requests and responses between an asserting authority and a trusted partner. OpenSSO Enterprise supports a number of the defined profiles. This section describes how to configure for the operations defined by the SAMLv2 profiles using OpenSSO Enterprise. It covers the following topics.

- "POST Binding with Single Sign-on and Single Logout" on page 180
- "Creating Affiliations" on page 181
- "Requesting Attribute Values Using a SAMLv2 Assertion" on page 182
- "Requesting a SAMLv2 Assertion" on page 185
- "Requesting a SAMLv2 Assertion for Authentication Context" on page 188
- "Encoding Artifacts" on page 189
- "Managing SAMLv2 Name Identifiers" on page 189
- "Mapping SAMLv2 Name Identifiers" on page 191
- "Enhanced Client and Proxy" on page 192
- "Formatting Name Identifiers" on page 194
- "Configuring SAMLv2 Single Sign-on without Service Provider User Accounts" on page 195
- "Auto-creation of User Accounts" on page 198

- "Using Non-Default Federation Attributes" on page 199
- "Enabling XML Signing and Encryption" on page 200
- "Securing SOAP Binding" on page 200
- "Load Balancing" on page 201
- "Access Control" on page 203
- "Certificate Revocation List Checking" on page 206
- "SAMLv2 IDP Discovery Service" on page 211
- "Bootstrapping the Liberty ID-WSF with SAML v2" on page 212

POST Binding with Single Sign-on and Single Logout

HTTP POST binding is used for an identity provider response to a request from a service provider. To configure for POST binding, the following tags must be present in the identity provider standard metadata.

```
<IDPSSODescriptor
WantAuthnRequestsSigned="false"
protocolSupportEnumeration="urn:oasis:names:tc:
SAML:2.0:protocol">.
  <SingleLogoutService
   Binding="urn:oasis:names:tc:SAML:2.0:
   bindings:HTTP-POST"
   Location="http://isdev-3.red.com:
   58080/fam/IDPSloPOST/metaAlias/idp"
   ResponseLocation="http://isdev-3.red.com:
   58080/opensso/IDPSloPOST/metaAlias/idp"/>
  <SingleSignOnService
   Binding="urn:oasis:names:tc:SAML:2.0:bindings:
   HTTP-POST"
   Location="http://isdev-3.red.iplanet.com:58080/opensso/
   SSOPOST/metaAlias/idp"/>
</IDPSSODescriptor>
```

To configure on the service provider side the standard metadata must include the following tags.

```
<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-POST"
 Location="http://mach1.red.com:58080/opensso/
 SPSloPOST/metaAlias/sp"
 ResponseLocation="http://mach1.red.com:58080/
 opensso/SPSloPOST/metaAlias/sp"/>
</SPSSODescriptor>
```

`idpSSOInit.jsp`, `spSSOInit.jsp`, `spSingleLogoutInit.jsp` and `idpSingleLogoutInit.jsp` will initiate single sign-on or single logout using the proper binding. Supported values are `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect` and `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST`. An example URL for service provider initiated single logout might be `http://mach1.red.com:58080/opensso/saml2/jsp/spSingleLogoutInit.jsp ?metaAlias=/sp&idpEntityID=isdev-3.red.com&binding=urn:oasis:names:tc: SAML:2.0:bindings:HTTP-POST`

Creating Affiliations

To configure for affiliations, the following tags must be in the identity provider standard metadata.

```
<AffiliationDescriptor
 affiliationOwnerID="affi.red.com">
  <AffiliateMember>mach1.red.com</AffiliateMember>
  <AffiliateMember>mach2.red.com</AffiliateMember>
</AffiliationDescriptor>
```

The following tags must also be present in the identity provider extended metadata.

```
<AffiliationConfig metaAlias="/ff">
  <Attribute name="signingCertAlias">
   <Value>test</Value>
  </Attribute>
  <Attribute name="encryptionCertAlias">
   <Value>test</Value>
  </Attribute>
</AffiliationConfig>
```

The `ssoadm` command line interface can be used to create and import the identity provider metadata. Use the following options to create the appropriate tags in the metadata. See Part I, “Command Line Interface Reference,” in *Sun OpenSSO Enterprise 8.0 Administration Reference* for more information.

--affiliation, -F	Specify a metaAlias for the hosted affiliation. The format must be *realm name*/*identifier*.
--affiscertalias, -J	Specify a signing certificate alias for the hosted affiliation.
--affiecertalias, -K	Specify an encryption certificate alias for the hosted affiliation.
--affimembers, -M	Specify affiliation members.

--affiownerid, -N	Specify the identifier for the Affiliation Owner.

An example illustrating how the command line might be used to create the metadata:

```
ssoadm create-metadata-templ -u amadmin
-f /tmp/pw -m /home/tmp/affimm -x /home/tmp/affixx
-F /ff -y affi.red.com -K test -J test -M sp1.red.com
sp2.red.com -N affiownerID
```

Note – See Chapter 1, "ssoadm Command Line Interface Reference," in *Sun OpenSSO Enterprise 8.0 Administration Reference* for information on the other options.

`idpMNIRequestInit.jsp`, `idpSSOInit.jsp`, `spMNIRequestInit.jsp` and `spSSOInit.jsp` can initiate single sign-on based on a configured affiliation. The `affiliationID` parameter should match the value of the entity ID for the affiliation in the standard metadata. A URL to initiate single sign-on from the service provider might be:

```
http://mach1.red.com:58080/opensso/saml2/jsp/
spSSOInit.jsp?metaAlias=/sp&idpEntityID=isdev-3.red.com&reqBinding=
urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST&affiliationID=affi.red.com
```

Requesting Attribute Values Using a SAMLv2 Assertion

Providers may request attributes (and the corresponding values) from a specific identity profile. A successful response is the return of an assertion containing the requested information. The identity provider acting as the attribute authority uses an implementation of the `com.sun.identity.saml2.plugins.AttributeAuthorityMapper` interface to process queries. The implementation uses the attribute map table configured in the identity provider's extended metadata which maps attributes in the SAMLv2 assertion to attributes in the local user data store. (If an attribute map is not configured, no attributes will be returned.)

OpenSSO Enterprise contains two custom mappers:

`com.sun.identity.saml2.plugins.DefaultAttributeAuthorityMapper`

`com.sun.identity.saml2.plugins.DefaultAttributeAuthorityMapper` maps using the NameID from a single sign-on interaction. To set OpenSSO Enterprise to use a different attribute mapper implementation, modify the value of the `default_attributeAuthorityMapper` property in the extended metadata of the provider defined as the attribute authority. The mapper value of `default_attributeAuthorityMapper` is used for a standard attribute queries

`com.sun.identity.saml2.plugins.X509SubjectAttributeAuthorityMapper`
: `com.sun.identity.saml2.plugins.X509SubjectAttributeAuthorityMapper` maps using the value of the X.509 Subject in the certificate in the NameID. To set OpenSSO Enterprise to use a different attribute mapper implementation, modify the value of the `x509Subject_attributeAuthorityMapper` property in the extended metadata of the provider defined as the attribute authority. The mapper value of `x509Subject_attributeAuthorityMapper` is used for attribute queries with an X509 certificate. The X509 mapper maps an X509 subject to a user by searching the identity data store for the attribute defined as the value of the `x509SubjectDataStoreAttrName` property in the identity provider extended metadata of the attribute authority. If the user has the specified attribute and the attribute's value is the same as that of the X509 subject in the attribute query, the user will be used.

Only SOAP binding is supported for these communications. Signing is required so make sure the Signing Certificate Alias attribute of any provider acting as the attribute requester and the attribute authority is configured. The `ssoadm` command line interface can be used to create and import the service provider metadata. The following tags must be in the standard metadata of the service provider (querying provider).

```
<RoleDescriptor
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:query="urn:oasis:names:tc:SAML:metadata:ext:query"
  xsi:type="query:AttributeQueryDescriptorType"
  protocolSupportEnumeration=
  "urn:oasis:names:tc:SAML:2.0:protocol">
</RoleDescriptor>
```

The following tags must be in the extended metadata of the service provider (querying provider).

```
<AttributeQueryConfig metaAlias="/attrq">
  <Attribute name="signingCertAlias">
   <Value>test2</Value>
  </Attribute>
  <Attribute name="encryptionCertAlias">
    <Value>test2</Value>
  </Attribute>
</AttributeQueryConfig>
```

Use the following options to create the appropriate tags in the service provider's metadata. See Part I, "Command Line Interface Reference," in *Sun OpenSSO Enterprise 8.0 Administration Reference* for more information.

`--attrqueryprovider, -S`	Specify a metaAlias for the hosted querying provider. The format must be *realm name/identifier*.
`--attrqscertalias, -A`	Specify a signing certificate alias.

`--attrqecertalias`, `-R`	Specify an encryption certificate alias.

The `ssoadm` command line interface can also be used to create and import the identity provider metadata. The following tags must be in the standard metadata of the identity provider (attribute authority).

```
<AttributeAuthorityDescriptor
  protocolSupportEnumeration=
  "urn:oasis:names:tc:SAML:2.0:protocol">
</AttributeAuthorityDescriptor>
```

The following tags must be in the extended metadata of the identity provider (attribute authority). Note the presence of the `x509SubjectDataStoreAttrName` attribute.

```
<AttributeAuthorityConfig metaAlias="/attra">
  <Attribute name="signingCertAlias">
   <Value>test2</Value>
  </Attribute>
  <Attribute name="encryptionCertAlias">
   <Value>test2</Value>
  </Attribute>
  <Attribute name="default_attributeAuthorityMapper">
   <Value>com.sun.identity.saml2.plugins.DefaultAttributeAuthorityMapper</Value>
  </Attribute>
  <Attribute name="x509Subject_attributeAuthorityMapper">
   <Value>com.sun.identity.saml2.plugins.X509SubjectAttributeAuthorityMapper</Value>
  </Attribute>
  <Attribute name="x509SubjectDataStoreAttrName">
   <Value></Value>
  </Attribute>
</AttributeAuthorityConfig>
```

Use the following options to create the appropriate tags in the identity provider's metadata. See Part I, "Command Line Interface Reference," in *Sun OpenSSO Enterprise 8.0 Administration Reference* for more information.

`--attrauthority`, `-I`	Specify a metaAlias for the hosted attribute authority. The format must be *realm name/identifier*.
`--attrascertalias`, `-B`	Specify a signing certificate alias.
`--attraecertalias`, `-G`	Specify an encryption certificate alias.

To initiate this query, create and import the standard and extended metadata for both the service provider and identity provider. Add the mapped values to the `attributeMap` property in the extended identity provider metadata in the following format:

attribute in SAML assertion=local attribute

Tip – You can specify the attributes to be returned in the `Attribute` tag of the `AttributeAuthorityDescriptor` element of the identity provider standard metadata. If this attribute has no value, all requested attributes will be returned.

To send an attribute query from the provider use the method of `com.sun.identity.saml2.profile.AttributeQueryUtil`.

```
public static Response sendAttributeQuery(
  AttributeQuery attrQuery,
  String attrAuthorityEntityID,
  String realm,
  String attrQueryProfile,
  String attrProfile, String binding) throws SAML2Exception;
```

To construct an `AttributeQuery` object, use the `com.sun.identity.saml2.assertion.*` and `com.sun.identity.saml2.protocol.*` packages.

Requesting a SAMLv2 Assertion

The Assertion Query/Request profile specifies a means for requesting existing assertions using a unique identifier. The requester initiates the profile by sending an assertion request, referenced by the identifier, to a SAMLv2 authority. The SAMLv2 authority processes the request, checks the assertion cache for the identifier, and issues a response to the requester.

Note – To store assertions generated during single sign-on, add the following attribute to the metadata file of the identity provider acting as the SAMLv2 authority.

```
<IDPSSOConfig metaAlias="/idp">
<Attribute name="assertionCacheEnabled">
<Value>true</Value>
</Attribute>
</IDPSSOConfig>
```

To configure for assertion queries, the following tags must be defined in the identity provider standard metadata.

```
<IDPSSODescriptor WantAuthnRequestsSigned=
"false" protocolSupportEnumeration="urn:oasis:names:tc:
SAML:2.0:protocol">

  <AssertionIDRequestService Binding="urn:oasis:names:tc:
   SAML:2.0: bindings:SOAP" Location=
   "http://isdev-3.red.iplanet.com:58080/
   fam/AIDReqSoap/IDPRole/metaAlias/idp"/>
    <AssertionIDRequestService Binding=
     "urn:oasis:names:tc:SAML:
     2.0:bindings:URI" Location=
     "http://isdev-3.red.iplanet.com:
     58080/fam/AIDReqUri/IDPRole/metaAlias/idp"/>
</IDPSSODescriptor>

<AttributeAuthorityDescriptor protocolSupportEnumeration=
"urn:oasis:names:tc:SAML:2.0:protocol">
  <AssertionIDRequestService Binding=
   "urn:oasis:names:tc:SAML:
   2.0:bindings:SOAP" Location=
   "http://isdev-3.red.iplanet.com:
   58080/fam/AIDReqSoap/AttrAuthRole/metaAlias/attra"/>
    <AssertionIDRequestService Binding=
     "urn:oasis:names:tc:SAML:
     2.0:bindings:URI" Location=
     "http://isdev-3.red.iplanet.com:
     58080/fam/AIDReqUri/AttrAuthRole/
     metaAlias/attra"/>
</AttributeAuthorityDescriptor>

<AuthnAuthorityDescriptor protocolSupportEnumeration=
"urn:oasis:names:tc:SAML:2.0:protocol">
..<AssertionIDRequestService Binding="urn:oasis:names:tc:SAML:
   2.0:bindings:SOAP" Location="http://isdev-3.red.iplanet.com:
   58080/fam/AIDReqSoap/AuthnAuthRole/metaAlias/authna"/>
  <AssertionIDRequestService Binding="urn:oasis:names:tc:SAML:
   2.0:bindings:URI" Location="http://isdev-3.red.iplanet.com:
   58080/fam/AIDReqUri/AuthnAuthRole/metaAlias/authna"/>
..</AuthnAuthorityDescriptor>
```

The following tags must be defined in the identity provider extended metadata.

```
<IDPSSOConfig metaAlias="/idp">
..<Attribute name="assertionIDRequestMapper">
   <Value>com.sun.identity.saml2.plugins.
    DefaultAssertionIDRequestMapper</Value>
   </Attribute>
</IDPSSOConfig>

<AttributeAuthorityConfig metaAlias="/attra">
..<Attribute name="assertionIDRequestMapper">
   <Value>com.sun.identity.saml2.plugins.
```

```
    DefaultAssertionIDRequestMapper</Value>
  </Attribute>
</AttributeAuthorityConfig>

<AuthnAuthorityConfig metaAlias="/authna">
..<Attribute name="assertionIDRequestMapper">
   <Value>com.sun.identity.saml2.plugins.
    DefaultAssertionIDRequestMapper</Value>
  </Attribute>
</AuthnAuthorityConfig>
```

`com.sun.identity.saml2.plugins.DefaultAssertionIDRequestMapper` is the default implementation used to process the assertion request. (See `com.sun.identity.saml2.plugins.AssertionIDRequestMapper` in the *Sun OpenSSO Enterprise 8.0 Java API Reference*.) To define a customized mapper, change the value of the `assertionIDRequestMapper` property in the IDP, attribute authority or authentication authority extended metadata.

Supported bindings are SOAP and URI however in order to implement URI binding, you must do the following.

1. Write an implementation of `com.sun.identity.saml2.plugins.AssertionIDRequestMapper`.

   The method `authenticateRequesterURI()` should be returned without throwing an exception.
2. Modify the value of the `assertionIDRequestMapper` element in the identity provider metadata to match the name of the custom implementation.

- To send a request for an assertion from a service provider use either of the methods of `com.sun.identity.saml2.profile.AssertionIDRequestUtil` as below.

  ```
  public static Response sendAssertionIDRequest(
    AssertionIDRequest assertionIDRequest, String samlAuthorityEntityID,
    String role, String realm, String binding) throws SAML2Exception;

  public static Assertion sendAssertionIDRequestURI(
    String assertionID, String samlAuthorityEntityID,
    String role, String realm) throws SAML2Exception;
  ```
- To construct an `AssertionIDRequest` object, use `com.sun.identity.saml2.assertion.*` and `com.sun.identity.saml2.protocol.*`.

Requesting a SAMLv2 Assertion for Authentication Context

A SAMLv2 assertion contains information regarding the context of a principal's authentication. The requesting party may require this additional information (for example, the authenticating technology or protocol used) in order to assess the level of confidence they can place in the assertion. To retrieve authentication context information, the service provider issues a query to the authentication authority. Only SOAP binding is supported for this request And signing is required so make sure the Signing Certificate Alias attribute of the service provider and the authentication authority is configured.

▼ To Configure for Authentication Context Queries

1 **Create and load the metadata for the service provider.**

2 **Create the metadata for the identity provider using `ssoadm` and define these additional options for it's role as an authentication authority.**

-C	Defines the meta Alias for the hosted authentication authority to be created. The format must be *realm name/identifier*.
-D	Defines the authentication authority signing certificate alias.
-E	Defines the authentication authority encryption certificate alias.

For example:

```
ssoadm create-metadata-templ -u amadmin -f /tmp/pw -m /home/user1/tmp/mm -x
/home/usr1/tmp/xx -s /idp -a test -r test -C /authna -D test2 -E test2 -y
example.com
```

3 **Add the following attribute to the identity provider metadata file just created.**

This allows the identity provider to store assertions generated during the SAMLv2 Single Sign-on process.

```
<IDPSSOConfig metaAlias="/idp">
<Attribute name="assertionCacheEnabled">
<Value>true</Value>
</Attribute>
</IDPSSOConfig>
```

4 **Configure for SAMLv2 single sign-on as documented in "Configuring SAMLv2 Single Sign-on without Service Provider User Accounts" on page 195.**

5 **Do either of the following:**

- **To send an authentication query from the service provider use the method of** `com.sun.identity.saml2.profile.AuthnQueryUtil`**.**

  ```
  public static Response sendAuthnQuery(AuthnQuery authnQuery,
    String authnAuthorityEntityID, String realm, String binding)
    throws SAML2Exception;
  ```

- **To construct an** `AuthnQuery` **object, use** `com.sun.identity.saml2.assertion.*` **and** `com.sun.identity.saml2.protocol.*`**.**

Encoding Artifacts

When an identity provider sends a response to a service provider using artifact binding, OpenSSO Enterprise supports either URI or form encoding. To specify the encoding, toggle the value of the `responseArtifactMessageEncoding` tag in the service provider's extended metadata. Possible values are URI and FORM as in:

```
<Attribute name="responseArtifactMessageEncoding">
  <Value>FORM</Value>
</Attribute>
```

Managing SAMLv2 Name Identifiers

With this release, OpenSSO Enterprise enhances its implementation of the Name Identifier Management Profile to include the termination of the association of a name identifier between a service provider and an identity provider (including the accompanying federation) and the issuance of a new name identifier. When metadata is created using OpenSSO Enterprise, XML is defined to support HTTP-Redirect, SOAP and HTTP-POST bindings. Following is the code for an identity provider.

```
<IDPSSODescriptor
  <ManageNameIDService
   Binding="urn:oasis:names:tc:SAML:2.0:bindings:
   HTTP-Redirect" Location="http://isdev-3.red.iplanet.com:
   58080/fam/IDPMniRedirect/metaAlias/idp" ResponseLocation=
   "http://isdev-3.red.iplanet.com:58080/fam/IDPMniRedirect/
   metaAlias/idp"/>
  <ManageNameIDService
   Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
   Location="http://isdev-3.red.iplanet.com:58080/fam/
   IDPMniPOST/metaAlias/idp" ResponseLocation=
   "http://isdev-3.red.iplanet.com:58080/fam/IDPMniPOST/
```

```
   metaAlias/idp"/>
  <ManageNameIDService
   Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
   Location="http://isdev-3.red.iplanet.com:58080/fam/
   IDPMniSoap/metaAlias/idp"/>
</IDPSSODescriptor>
```

The *ManageNameID* (MNI) JSP provide a way to initiate name identifier changes or terminations. For example, after establishing a name identifier for use when referring to a principal, the identity provider may want to change its value and/or format. Additionally, an identity provider might want to indicate that a name identifier will no longer be used to refer to the principal. The identity provider will notify service providers of the change by sending them a `ManageNameIDRequest`. A service provider also uses this message type to register or change the `SPProvidedID` value (included when the underlying name identifier is used to communicate with it) or to terminate the use of a name identifier between itself and the identity provider. To initiate termination of a name identifier or creation of a new identifier, access the appropriate JSP using the URL and URL parameter information in the following sections.

- "`idpMNIRequestInit.jsp`" on page 190
- "`spMNIRequestInit.jsp`" on page 191

The JSP are located in `/`*OpenSSO-Deploy-base*`/opensso/saml2/jsp/`. `idpMNIRedirect.jsp`, `spMNIRedirect.jsp`, `idpMNIPOST.jsp`, and `spMNIPOST.jsp`, also in that directory, are process pages served as endpoints.

idpMNIRequestInit.jsp

`idpMNIRequestInit.jsp` initiates name identifier modifications or termination from the identity provider. The URL for this JSP is *protocol*`://`*host*`:`*port*`/`*service-deploy-uri*`/saml2/jsp/idpMNIRequestInit.jsp`. The following URL parameters are appended to it.

- `metaAlias`: The value of the `metaAlias` property set in the identity provider's extended metadata configuration file. If the `metaAlias` attribute is not present, an error is returned.
- `spEntityID`: The entity identifier of the service provider to which the response is sent.
- `requestType`: The type of `ManageNameIDRequest`. Accepted values include `Terminate` and `NewID`.
- `binding`: A URI specifying the binding to use for the communications. The supported values are:
  - `urn:oasis:names:tc:SAML:2.0:bindings:SOAP`
  - `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect`
  - `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST`
- `RelayState`: The target URL of the request.

An example URL for using HTTP-POST communication might be:

```
http://dev-3.sun.com:58080/opensso/saml2/
  jsp/idpMNIRequestInit.jsp?metaAlias=/idp&spEntityID=
  mach1.sun.com&requestType=Terminate&binding=
  urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST
```

`spMNIRequestInit.jsp`

`spMNIRequestInit.jsp` initiates name identifier modifications or termination from the service provider. The URL for this JSP is *protocol*`://`*host*`:`*port*`/`*service-deploy-uri*`/saml2/jsp/spMNIRequestInit.jsp`. The following URL parameters are appended to it.

- `metaAlias`: This parameter takes as a value the `metaAlias` set in the identity provider's extended metadata configuration file. If the `metaAlias` attribute is not present, an error is returned.
- `idpEntityID`: The entity identifier of the identity provider to which the request is sent.
- `requestType`: The type of `ManageNameIDRequest`. Accepted values include `Terminate` and `NewID`.
- `binding`: A URI specifying the protocol binding to use for the `Request`. The supported values are:
  - `urn:oasis:names:tc:SAML:2.0:bindings:SOAP`
  - `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect`
  - `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST`
- `RelayState`: The target URL of the request.

An example URL for using SOAP communication might be:

```
http://dev-3.sun.com:58080/opensso/saml2/
  jsp/idpMNIRequestInit.jsp?metaAlias=/sp&idpEntityID=
  mach1.sun.com&requestType=NewID&binding=
  urn:oasis:names:tc:SAML:2.0:bindings:SOAP
```

Mapping SAMLv2 Name Identifiers

The NameID Mapping Protocol allows a service provider that shares an identifier for a principal with an identity provider to obtain a name identifier for said principal in another format or that is in another federation namespace (for example, is shared between the identity provider and another service provider). The requester initiates the profile by sending a `NameIDMappingRequest` message to the identity provider. After processing the request, the identity provider issues a `NameIdMappingResponse` message to the requester.

Only SOAP binding is supported. Signing is required so make sure the Signing Certificate Alias attribute of the identity provider and the service provider is configured.

To send a `NameIDMappingRequest` message from the service provider, use the method of the `com.sun.identity.saml2.profile.NameIDMapping`.

```
public static NameIDMappingResponse initiateNameIDMappingRequest(
  Object session,
  String realm,
  String spEntityID,
  String idpEntityID,
  String targetSPEntityID,
  String targetNameIDFormat,
  Map paramsMap) throws SAML2Exception;
```

Enhanced Client and Proxy

The Enhanced Client and Proxy (ECP) profile assumes the client can contact an appropriate provider using the reverse SOAP (PAOS) binding. The ECP process flow is as follows:

1. The principal, through the ECP, makes an HTTP request for a secured resource at a service provider, which does not have an established security context for the ECP or principal.
2. The service provider issues an authentication request to the ECP using PAOS binding.
3. The ECP obtains the location of an endpoint at an identity provider for the authentication request protocol that supports its binding.
4. The ECP conveys the authentication request to the identity provider.
5. The identity provider identifies the principal.
6. The identity provider issues a response to the ECP that is to be delivered to the service provider.
7. The ECP conveys a response message to the service provider.
8. The service provider grants or denies access to the principal.

See the following procedures for configuration information.

- "To Configure for ECP on the Identity Provider Side" on page 193
- "To Configure for ECP on the Service Provider Side (Optional)" on page 193

After completing the configuration, use the following URL format to access a resource on the service provider.

SP protocol`://`*SP host*`:`*SP port*`/`*SP deploy URI*`/SPECP?metaAlias=`*sp metaAlias*`&RelayState=`*resource url*

▼ To Configure for ECP on the Identity Provider Side

1. **Click the Federation tab and select the hosted SAMLv2 identity provider you wish to edit.**
2. **Click on the IDP tab.**
3. **Click the Advanced tab.**
4. **Edit the IDP Session Mapper attribute for you deployment.**

   The session mapper SPI `com.sun.identity.saml2.plugins.IDPECPSessionMapper` finds a valid session from the HTTP servlet request on the identity provider side with the ECP profile. The default implementation will construct a session object from the OpenSSO Enterprise server cookie. To construct a session from other cookies or HTTP headers, you need to implement this SPI and set your implementation here.
5. **Click Save.**

▼ To Configure for ECP on the Service Provider Side (Optional)

1. **Click the Federation tab and select the hosted SAMLv2 identity provider you wish to edit.**
2. **Click on the SP tab.**
3. **Click the Advanced tab.**
4. **Click ECP Configuration.**
5. **Edit Request IDP List Finder Implementation the IDP Finder SPI.**

   `com.sun.identity.saml2.plugins.SAML2IDPFinder` finds a list of preferred identity providers. You can write your own implementation of this interface and set it here. The default implementation will read Request IDP List attribute. Request IDP List Get Complete Specifies an URI reference that can be used to retrieve the complete IDP list if the IDPList element is not complete. Request IDP List Defines a list of IDPs for the ECP to contact. This is used by the default implementation of the IDP Finder.
6. **Click Save.**

Formatting Name Identifiers

Name identifiers are used by the identity provider and the service provider to communicate with each other regarding a principal. As of this release, OpenSSO Enterprise supports the following name identifier formats.

- `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified` (enabled by default)
- `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress` (enabled by default)
- `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName`
- `urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName`
- `urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos`
- `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`
- `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent` (enabled by default)
- `urn:oasis:names:tc:SAML:2.0:nameid-format:transient` (enabled by default)

OpenSSO Enterprise defines these name identifiers in the identity provider's standard metadata. If no specific name identifier format is requested by the service provider, a default format must be used in the authentication response. To enable one or more of the supported formats you must add a name identifier format/user attribute map to the identity provider extended metadata to generate the name identifier based on the specified user profile attribute. The value is formatted as *name ID format=user profile attribute* as in the following XML sample

```
<Attribute name="nameIDFormatMap">
  <Value>urn:oasis:names:tc:SAML:1.1:nameid-format:
   emailAddress=mail</Value>
  <Value>urn:oasis:names:tc:SAML:1.1:nameid-format:
   X509SubjectName=</Value>
  <Value>urn:oasis:names:tc:SAML:1.1:nameid-format:
   WindowsDomainQualifiedName=</Value>
  <Value>urn:oasis:names:tc:SAML:2.0:nameid-format:
   kerberos=</Value>
</Attribute>
```

If a user profile attribute is specified, the name ID value will be the value of the user profile attribute. If no user profile attribute is specified an exception will be thrown. If the name ID format is persistent or transient, a random string will be generated. For more information on persistent and transient identifiers, see "Configuring SAMLv2 Single Sign-on without Service Provider User Accounts" on page 195.

Note – To disable one or more name ID formats, the format XML tags can be removed from the identity provider's standard metadata.

Configuring SAMLv2 Single Sign-on without Service Provider User Accounts

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. OpenSSO Enterprise supports persistent identifiers by default and transient identifiers with configuration. NameID formats other than persistent and transient are supported by mapping the NameID format to a user profile attribute.

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.

- "To Use the Transient Name Identifier" on page 195
- "To Federate Disparate Accounts with Auto Federation" on page 195
- "To Map Attributes to `anonymous` User Account" on page 196
- "To Achieve Single Sign-on Without Data Store Writes" on page 197

▼ To Use the Transient Name Identifier

1 **Append the `NameIDFormat=urn:oasis:names:tc:SAML:2.0:nameid-format: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 *Sun OpenSSO Enterprise 8.0 Developer's Guide*.

▼ To Federate Disparate Accounts with Auto Federation

The auto-federation feature in OpenSSO Enterprise 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.

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 OpenSSO Enterprise Console, click the Federation tab.**
2. **Select the name of the hosted identity provider to edit its profile.**
3. **Click the Assertion Processing tab.**
4. **In the Attribute Map attribute, add the `autofedAttribute=local-attribute` value. For example, `employeeNumber=employeeID`.**
5. **Click Save.**
6. **Go back to the Federation tab and select the name of the hosted service provider to edit its profile.**
7. **If the Auto Federation Common Attribute Name is the same as local attribute name, skip to next step. If not, enter the `autofedAttribute=local-attribute` value in the New Value field under Attribute Map. For example:**

   ```
   employeeNumber=employeeID
   ```
8. **Click on the Auto Federation link at the top of the page, or scroll down to the Auto Federation subsection.**
9. **Enable Auto Federation by checking the box.**
10. **Click Save to complete the configuration.**

▼ To Map Attributes to `anonymous` 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 `Attribute Statement` 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. **In the console, select the identity provider you wish to configure.**

2 **Edit `Attribute Map` attribute.**

`attribute Map` 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:

```
mail=mail
employeeNumber=employeeNumber
```

3 **From the console, select the service provider you wish to configure.**

4 **Edit the following attributes for the service provider.**

- `transient User` will take a value of one of the existing transient user identifiers on the service provider side, for example, `anonymous`.
- `attribut eMap` 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:

  ```
  >mail=mail
  employeeNumber=employeeNumber
  ```

5 **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.

▼ To Achieve 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 **Edit the following attributes in the OpenSSO Enterprise console for the identity provider.**

- `Auto Federation` – enable this attribute.
- `Auto Federation Attribute` defines the common attribute on the identity provider side. For example, `mail`.
- `Attribute Map` 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>
  ```

2 **Edit the following attributes in the OpenSSO Enterprise console for the service provider.**

- `Transient User` takes a null value.
- `Auto Federation` enable this attribute.
- `Auto Federation Attribute` defines the common attribute. For example, `mail`.
- `Attribute Map` 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:

  ```
  mail=mail
  ```

3 **Invoke the single sign-on URL with the `NameIDFormat=transient` query parameter appended to it to test.**

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 OpenSSO Enterprise 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 **Enable auto Federation for the Identity Provider. For more information, see "To Federate Disparate Accounts with Auto Federation" on page 195.**

2 **Click Save.**

3 **Repeat the above steps to modify the service provider's extended metadata.**

4 **Enable Dynamic Profile Creation using the OpenSSO Enterprise console.**

a. **Log in to the OpenSSO Enterprise 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 OpenSSO Enterprise.**

5 **To test, invoke single sign-on from the service provider.**

Using Non-Default Federation Attributes

If OpenSSO Enterprise 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 **In the OpenSSO Enterprise console, go to Configuration>Global>SAMLv2 Service Configuration.**

2 **Modify the following attributes:**

- Attribute name for Name ID information
- Attribute name for Name ID information key

See “SAMLv2 Service Configuration” in *Sun OpenSSO Enterprise 8.0 Administration Reference* for more information.

3 **Restart the web container.**

Federation information will now be written to the specified attributes.

Enabling XML Signing and Encryption

XML signing and encryption is most easily configured through the OpenSSO Enterprise console. To enable request/response signing and encryption, click on the name of the entity provider you wish to edit in the Federation tab. For both service and identity providers, the signing attributes are located under the Assertion Content sub tab.

For definitions for service provider attributes, see "SAMLv2 Service Provider Customization " in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

For definitions of identity provider attributes, see "SAMLv2 Identity Provider Customization" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

Securing SOAP Binding

SOAP binding supports the following authentication methods to protect interactions between SAML v2 entities:

- "Basic Authentication" on page 200
- "Secure Socket Layer/Transport Layer Security" on page 200

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. These attributes are located in the Assertion Content tab of the SAMLv2 entity provider profile.

TABLE 8–2 Securing SOAP Endpoint with Basic Authentication

Attribute	Description
`Basic Authenticaion Enabled`	Establishes that the SOAP endpoint is using basic authentication. Takes a value of `true` or `false`.
`User Name`	Defines the user allowed access to the protected SOAP endpoint in the original SAML v2 entity.
`Password`	Defines an encrypted password for the user. The password is encrypted using `ampassword` on the partner side. .

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™ platform (JVM™) running the SSL/TLS client:

`-Djavax.net.ssl.trustStore`	Defines the full path to the file containing the server's CA certificates.
`-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 Server Certificate Authenticaion..
`-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 OpenSSO Enterprise 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 OpenSSO Enterprise and follow the documentation to set up a load balancer.**

Load balancing information for OpenSSO Enterprise can be found in "Configuring the Directory Server Load Balancer" in *Deployment Example: SAML v2 Using Sun OpenSSO Enterprise 8.0*.

2 **Create an identity provider or a service provider through the OpenSSO Enterprise Console or with the `ssoadm` CLI.**

- Use the load balancer URL, not the server URL, to access the console. For information on creating an entity provider, see "To Create a SAMLv2 Entity Provider" on page 147.
- To create the provider through the command line, see "Managing Federation Using `ssoadm`" on page 155.

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 protocol 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 protocol 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

- 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 **Create a SAMLv2 identity provider. For more information, see "To Create a SAMLv2 Entity Provider" on page 147.**

2 **Create a SAMLv2 service provider.**

3 **Install the Sun Policy Agents 3.0 to protect the service provider configured on the instance of OpenSSO Enterprise**

For more information, see the *Sun OpenSSO Enterprise Policy Agent 3.0 User's Guide for Web Agents*.

4 **In the OpenSSO Enterprise console, go to Access Control>Realms>Agents and select the web policy agent profile you wish to configure.**

5 Go to the OpenSSO Services sub tab and edit the OpenSSO Login URL attribute .

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.

SP-protocol`://`*SP-host*`:`*SP-port*`/`*service-deploy-uri*`/`
`saml2/jsp/spSSOInit.jsp?NameIDFormat=transient&metaAlias=`*SP-metaAlias*`&`
`idpEntityID=`*IDP_EntityID*

For example:

```
http://example1.com:58080/
opensso/saml2/jsp/spSSOInit.jsp?NameIDFormat=transient&metaAlias=/sp&
idpEntityID=sample.com
```

6 (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 OpenSSO Enterprise. An example value might be **`http://`***host***`:`***port***`/opensso/UI/Login`**.

7 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
```

8 Set up single sign-on without requiring writes to the data store by following the procedure described in "To Achieve Single Sign-on Without Data Store Writes" on page 197.

To test, 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).

9 **Create a policy with the `Session Property` condition on the service provider instance of OpenSSO Enterprise.**

a. **Log in to the OpenSSO Enterprise 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 Session Property 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, use `employeenumber`.

p. **Add the match value to the Values field and click Add.**

   To test, 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 "Creating Policies and Referrals" on page 107.

10 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 reasons 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 OpenSSO Enterprise 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 the Advanced properties of the Sites and Server tab. For more information, see "Advanced" in *Sun OpenSSO Enterprise 8.0 Administration Reference*. 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.

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.

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 OpenSSO Enterprise 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 OpenSSO Enterprise 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 "To Manually Populate a Directory Server with a Certificate Revocation List" on page 209 for information on manual population.

3 **Configure OpenSSO Enterprise in the console to point to the instance of Directory Server designated as the CRL repository.**

a. **In the OpenSSO Enterprise Console, click the Configuration tab.**

b. **Click Servers and Sites tab.**

c. **Click the Server Name.**

d. **Click Security tab.**

e. **Click Inheritance Settings.**

f. **Uncheck the following properties:**
   - LDAP Search Base DN
   - LDAP Server Bind Password
   - LDAP Server Bind Username
   - LDAP Server Host Name
   - LDAP server port number
   - Search Attributes
   - SSL Enabled

g. **Click Save and then Back to Server Profile.**

h. **Click Certificate Revocation List Caching.**

i. **Configure the following attributes. See "Certificate Revocation List Caching" in** *Sun OpenSSO Enterprise 8.0 Administration Reference* **for definitions of the properties:**
   - LDAP Server Host Name
   - LDAP Server Port Number
   - SSL Enabled
   - LDAP Server Bind User Name
   - LDAP Server Bind Password
   - LDAP Search Base DN
   - Search Attributes

j. **Click Save.**

k. **Restart the web container.**

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* **`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
  ```

▼ To Enable Certificate Revocation List Checking for SAMLv2

1 In the OpenSSO Enterprise Console, click the Configuration tab.

2 Click the Global sub tab.

3 Click SAMLv2 Service Configuration.

4 Check the box for XML Signing Certificate Validation.

5 Check the box for CA Certificate Validation. This step is optional.

6 Click Save.

7 Restart the web container.

SAMLv2 IDP Discovery Service

In deployments having more than one identity provider, service providers need to determine which identity provider 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.

The Reader and Writer URLs, used by the SAML v2 IDP Discovery Service, are defined when configuring the circle of trust. 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. The following section describes the procedure for setting up and testing the Identity Provider Discovery Service.

For steps to deploy the IDP Discovery Service, see Chapter 10, “Deploying the Identity Provider (IDP) Discovery Service,” in *Sun OpenSSO Enterprise 8.0 Installation and Configuration Guide*.

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 SAMLv2 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://example.com</saml:Issuer>
<saml:Subject>
<saml:NameID NameQualifier="http://example.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/
  opensso/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://example.iplanet.com
  /aWQ9aWRwLG91PXVzZXIsZGM9aXBsYW5ldCxkYz1jb20sYW1zZGtkbj11aWQ9aWRwLG91PXBlb3BsZ
  SxkYz1pcGxhbmV0LGRjPWNvbQ%3D%3D</ResourceID>
<ServiceInstance xmlns="urn:liberty:disco:2003-08">
<ServiceType>urn:liberty:disco:2003-08</ServiceType>
<ProviderID>http://mach1.red.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://mach1.red.com:58080/opensso/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://mach1.red.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.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://example.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.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.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://example.com" Format="urn:oasis:names:tc:SAML:2.0:
  nameid-format:persistent"  >HuCJIy9v5MdrjJQOgsuT4NWmVUl3
  </lib:IDPProvidedNameIdentifier>
  </sec:SessionSubject>
<sec:ProviderID>http://mach1.red.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.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.

▼ To Enable an Identity Provider for SAML v2 Bootstrapping of Liberty ID-WSF

1. **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 for the existing identity provider metadata, create the identity provider in the OpenSSO Enterprise console or with the** `ssoadm` **command line utility. See "Creating an Entity" on page 146.**

2. **In the OpenSSO Enterprise Console, click the Federation tab.**

3. **Click the hosted Identity Provider you wish to edit.**

4. **Check Discovery Bootstrapping Enabled check box.**

5. **Click Save.**

6. **Go to the Configuration tab.**

7. **Click the Servers and Sites tab.**

8. **Click Default Server Settings.**

9. **Click the Advanced tab.**

10. **If the** `com.sun.identity.liberty.ws.util.providerManagerClass` **property does not exist in the Advanced attribute table, add it and define the following value for it:**

    ```
    com.sun.identity.liberty.ws.util.providerManagerClass =
    com.sun.identity.saml2.plugins.SAML2ProviderManager
    ```

Note – By default, this property has no value. In this case, Liberty ID-WSF 1.1 works with ID-FF providers. After you change this value, ID-WSF 1.1 works with SAMLv2 providers but not ID-FF providers. To switch back to ID-FF providers, delete the attribute from the Advanced attribute table.

---

11 **Click Save.**

12 **Restart the web container.**

▼ To Enable a Service Provider for SAML v2 Bootstrapping of Liberty ID-WSF

1 **In the OpenSSO Enterprise Console, click the Federation tab.**

2 **Click the hosted Service Provider you wish to configure.**

3 **Click the Assertion Processing tab.**

4 **In the Attribute Map attribute, add the following value:**

```
urn:liberty:disco:2003-08:DiscoveryResourceOffering=urn:liberty:disco:2003-08:Discover
```

5 **Click Save.**

Retrieving SAMLv2 Bootstrapping of Liberty ID-WSF from the WSC

You can use the following API to retrieve the Discovery Service bootstrap resource offering and included security token from the web services client:

```
ResourceOffering SAML2SDKUtil.getDiscoveryBootStrapResourceOffering(
HttpServletRequest request)

List SAML2SDKUtil.getDiscoveryBootStrapCredentials(
HttpServletRequest request)
```

For more information, see the *Sun OpenSSO Enterprise 8.0 Java API Reference*.

WS-Federation Operations

This section provides steps for configuring OpenSSO Enterprise's implementation of WS-Federation so that single sign-on can work among OpenSSO and ADFS (Microsoft Active Directory Federation Service)-based environments. For a detailed description of deployment considerations, use cases, and configuration overviews, see Chapter 9, “Enabling Web Services Federation Between Active Directory Federation Service and OpenSSO Enterprise,” in *Sun OpenSSO Enterprise 8.0 Deployment Planning Guide*. For a detailed overview of OpenSSO Enterprise's implementation, see “Using WS-Federation” in *Sun OpenSSO Enterprise 8.0 Technical Overview*.

Note – The steps provided here are based on the assumption that you are proficient in setting up ADFS-based environment as described in

---

Enabling WS-Federation between an ADFS environment and an OpenSSO Enterprise environment involves exchanging metadata to enable a trust relationship. The steps in this section use host names consistent with those outlined in the Microsoft Active Directory Federation Services (ADFS) Step-by-Step Guide (`http://go.microsoft.com/fwlink/?linkid=49531`). You can, however, use any host names you wish.

Prior to this, the following requirements must be met:

- All communications between WS-Federation components must be made over SSL.

  ADFS does not perform an HTTP POST of a WS-Federation RequestSecurityTokenResponse (RSTR) to a non-HTTPS URL.
- The ADFS environment can rely on DNS.
- All servers must be time-synchronized. This is essential to proper token validation.
- Token signing certificates must be created and imported for both ADFS and OpenSSO Enterprise endpoints. This process is automated in ADFS, but requires the use of the `keytool` command for OpenSSO Enterprise.
- You have set up ADFS based on the instructions contained in theMicrosoft Active Directory Federation Services (ADFS) Step-by-Step Guide (`http://go.microsoft.com/fwlink/?linkid=49531`).

  Make sure that Single Sign-On is configured for the ADFS from `adfsaccount` to `adfsweb` through the `adfsresource`.
- You have created a new user in the `adatum.com` Active Directory domain. The login name must be the same as the UID you created in OpenSSO. To do so, go to Start>Administrative Tools>Active Directory Users and Computers. Right click Users and click on New>User.

Proceed to the following sections to see the configuration steps.

- “To Configure OpenSSO Enterprise as a Service Provider” on page 219

- "To Configure OpenSSO Enterprise as an Identity Provider" on page 222

▼ To Configure OpenSSO Enterprise as a Service Provider

1 In the ADFS environment, Add a new Resource Partner to `adfsaccount.adatum.com` and configure the following attributes:

Display Name	Enter a name, for example `OpenSSO SP`.
Federation Service URI	This must be the same as the `TokenIssuerName` in the service provider metadata file that you will create. For example: `urn:federation:mywsfedsp`

Federation Service endpoint URL	The last path component of this URL must the match `metaAlias` in the service provider extended meta data file that you will create. For example:
	
	`https://amhost(:amsecureport)/fam/WSFederationServlet/m`
	
	`/mywsfedsp`

2 Convert the Active Directory machine's token signing certificate file (for example, `adfsaccount_ts.cer`) to PEM format. You use OpenSSL for this conversion. For example:

```
openssl x509 in adfsaccount_ts.cer inform DER -out adfsaccount_ts.pem outform
PEM
```

3 Create the metadata and extended metadata for an identity provider using the `ssoadm` command line utility. For example purposes, the files are named `adatum.xml` and `adatumx.xml`..

For example:

create-meadata-templ –u amadmin –f *password_file* –m adatum.xml –x adtumx.xml –i /*metaalias* –y *entity_id* –c wsfed

4 Create the metadata and extended metadata for a service provider using the `ssoadm` command line utility. For example purposes, the files are named `wsfedsp.xml` and `wsfedspx.xml`.

Note – You can also use the OpenSSO Enterprise console to create a hosted service provider or identity provider. For more information, see "WS-Federation Entity Provider" on page 151.

For example:

```
create-metadata-templ –u amadmin –f password_file –m wsfedsp.xml –x wsfedspx.xml
–s /metaalias –y entity_id –c wsfed
```

5 **In** `adatum.xml`**, paste the PEM-encoded certificate from** `adfsaccount_ts.pem` **into the** `<ns2:X509Certificate>` **element.**

6 **In the hosted service provider (**`wsfedsp.xml`**), change the hostname and port in the** `<ns3:Address>` **element to match your configuration. For example:**

```
<?xml version="1.0" encoding="UTF8"
standalone="yes"?>
<Federation FederationID="mywsfedsp"
xmlns="http://schemas.xmlsoap.org/ws/2006/12/federation">
<TokenIssuerName>urn:federation:mywsfedsp</TokenIssuerName>
<TokenIssuerEndpoint>
<ns3:Address
xmlns:ns3="http://www.w3.org/2005/08/addressing">https://patlinux.red.ip
lanet.com:8443/fam/WSFederationServlet/metaAlias/mywsfedsp</ns3:Address>
</TokenIssuerEndpoint>
</Federation>
```

7 **In the hosted service provider (**`adatumx.xml`**), change the hostname and port in the** `<HomeRealmDiscoveryService>` **attribute to match your configuration. For example:**

```
<FederationConfig xmlns="urn:sun:fm:wsfederation:1.0:federationconfig"
xmlns:fm="urn:sun:fm:wsfederation:1.0:federationconfig"
hosted="1" FederationID="mywsfedsp">
<SPSSOConfig metaAlias="/mywsfedsp">
<Attribute name="displayName">
<Value>My Open Federation Service Provider</Value>
</Attribute>
<Attribute name="AccountRealmSelection">
<Value>cookie</Value>
</Attribute>
<Attribute name="AccountRealmCookieName">
<Value>amWSFederationAccountRealm</Value>
</Attribute>
<Attribute name="HomeRealmDiscoveryService">
<Value>http://patlinux.red.com:8180/fam/RealmSelectio
n/metaAlias/mywsfedsp</Value>
</Attribute>
<Attribute name="spAccountMapper">
<Value>com.sun.identity.wsfederation.plugins.DefaultADFSPartn
erAccountMapper</Value>
</Attribute>
<Attribute name="spAttributeMapper">
<Value>com.sun.identity.wsfederation.plugins.DefaultSPAttribu
teMapper</Value>
```

```
</Attribute>
</SPSSOConfig>
</FederationConfig>
```

8. **Load the identity provider and service provider metadata to OpenSSO Enterprise. From the console:**

   a. **Log in to the console and click the Federation tab and then the Import Entity button.**

   b. **Choose the realm to which the requesting service provider belongs.**

   c. **In the Where Does the Meta Data File Reside field, choose File and click Upload.**

   d. **Choose** `adatum.xml`**.**

   e. **Click Ok.**

   f. **In the Where Does the Extended Meta Data File Reside field, choose File and click Upload.**

   g. **Choose** `adtumx.xml`**.**

   h. **Click Ok.**

   i. **Repeat the steps for loading the service provider meta data (**`wsfedsp.xml` and `wsfedspx.xml`**).**

9. **Create a circle of trust and add the identity provider and service provider. For instructions, see "Circle of Trust" on page 152.**

10. **On the OpenSSO Enterprise instance, go to**
    `https://`*opensssohost*`(:`*openssosecureport*`)/opensso`
    `WSFederationServlet/`*metaAlias*`/mywsfedsp?goto=https://`*openssohost*`(:`*openssosecureport*`)/opens`

    You should be forwarded to the realm selection page. Click 'Proceed. You may see a few redirections in the browser's address bar before reaching the user's profile page in OpenSSO Enterprise.

    If you do this from outside the Window domain, you will get an HTTPBasic authentication username/password dialog. Enter the user's Active Directory credentials to gain access.

    The realm selection process sets a persistent cookie. If you enter the same URL a second time, you should not be prompted for a realm and should be redirected to the OpenSSO Enterprise user page.

11 **Configure your installed policy agent profile with the WS-Federation servlet as its login URL.**

For the J2EE policy agent profile:

- Log in to the console and go to Access Control>*realm*>Agents
- Click the name of the J2EE policy agent you wish to edit.
- In the OpenSSO Login URL attribute, enter the WS-Federation servlet, for example:

  `https://`*openssohost*`(:`*openssosecureport*`)/opensso/WSFederationServlet/`*metaAlias*`/mywsfedsp`

For the web policy agent profile:

- Log in to the console and go to Access Control>realm>Agents
- Click the name of the web policy agent you wish to edit.
- In the OpenSSO Login URL attribute, enter the WS-Federation servlet, for example:

  `https://`*openssohost*`(:`*openssosecureport*`)/opensso/WSFederationServlet/`*metaAlias*`/mywsfedsp`

When accessing the resource protected by the policy agent, you should be authenticated through WS-Federation.

▼ To Configure OpenSSO Enterprise as an Identity Provider

1 **Create a token signing certificate in a Java Keystore on the OpenSSO Enterprise machine. For example:**

```
keytool keystore keystore.jks genkey dname "CN=amhost" alias mywsfedidp
```

Specify the same password for the keystore and key. Put the keystore in any location, since you will need to specify the full path

2 **Encrypt the keystore/key password. The easiest method is to use the OpenSSO Enterprise** `encode.jsp`**:**

a. **Go to** `https://`*host*`:`*port*`/opensso/encode.jsp`**.**

b. **Enter the password.**

c. **Create two files,** `.storepass` **and** `.keypass`**, whose only content is the encrypted password.**

3 **Set the path to** `keystore.jks` **and the two files containing encrypted the passwords. To do so:**

a. **Log into the OpenSSO Enterprise console.**

b. Go to Configuration>Sites and Servers.

c. Click the Default Server Settings button and click the Security tab.

d. Configure the following attributes:

Keystore File	Set to /*path*/keystore.jks
Keystore Password File	Set to /*path*/.storepass
Private Key Password File	Set to /*path*/.keypass

e. You must restart the web container for the changes to take effect.

4 Export the token signing certificate in DER format. For example:

```
keytool keystore keystore.jks export alias mywsfedidp file cert.der
```

5 Copy `cert.der` to the `adfsresource` machine.

6 Create the metadata and extended metadata for a remote service provider using the `ssoadm` command line utility.

For example:

```
create-meadata-templ –u amadmin –f password_file –m treyresearrch.xml.xml –x
treyresearch.xmlx.xml –s /metaalias –y entity_id –c wsfed
```

For this example, the files are named `treyresearch.xml` and `treyresearchx.xml`.

7 Create the metadata and extended metadata for a hosted identity provider using the `ssoadm` command line utility.

Note – You can also use the OpenSSO Enterprise console to create a hosted service provider or identity provider. For more information, see "WS-Federation Entity Provider" on page 151.

For example:

```
create-meadata-templ –u amadmin –f password_file –m wsfedidp.xml –x wsfedidpx.xml
–i /metaalias –y entity_id –c wsfed
```

For this example, the files are named `wsfedidp.xml` and `wsfedidpx.xml`.

8 In the remote service provider (`treyresearch.xml`), change the hostname and port in the `<ns3:Address>` element to match your configuration.

9 **In the remote service provider (**`wsfedidpx.xml`**), change the hostname and port in the** `<HomeRealmDiscoveryService>` **attribute to match your configuration. For example:**

```
<FederationConfig xmlns="urn:sun:fm:wsfederation:1.0:federationconfig"
xmlns:fm="urn:sun:fm:wsfederation:1.0:federationconfig"
hosted="1" FederationID="mywsfedidp">
<IDPSSOConfig metaAlias="/mywsfedidp">
<Attribute name="displayName">
<Value>My Open Federation Identity Provider</Value>
</Attribute>
<Attribute name="upnDomain">
<Value>red.com</Value>
</Attribute>
<Attribute name="signingCertAlias">
<Value>mywsfedidp</Value>
</Attribute>
<Attribute name="assertionEffectiveTime">
<Value>600</Value>
</Attribute>
<Attribute name="idpAccountMapper">
<Value>com.sun.identity.wsfederation.plugins.DefaultIDPAccoun
tMapper</Value>
</Attribute>
<Attribute name="idpAttributeMapper">
<Value>com.sun.identity.wsfederation.plugins.DefaultIDPAttrib
uteMapper</Value>
</Attribute>
</IDPSSOConfig>
</FederationConfig>
```

10 **Load the identity provider and service provider metadata to OpenSSO Enterprise. From the console:**

a. **Log in to the console and click the Federation tab and then the Import Entity button.**

b. **Choose the realm to which the requesting service provider belongs.**

c. **In the Where Does the Meta Data File Reside field, choose File and click Upload.**

d. **Choose** `wsfedidp.xml`**.**

e. **Click OK.**

f. **In the Where Does the Extended Meta Data File Reside field, choose File and click Upload.**

g. **Choose** `wsfedidpx.xml`**.**

h. Click Ok.

i. Repeat the steps for loading the service provider meta data (`treyresearch.xml` and `treyresearchx.xml`).

11 Create a circle of trust and add the identity provider and service provider. For instructions, see "Circle of Trust" on page 152.

12 In the ADFS environment, add a new Account Partner to `adfsresource.treyresearch.net` and configure the following attributes:

Display Name	Enter a name, for example `OpenSSO IDP`.
Federation Service URI	This must be the same as the `TokenIssuerName` in the identity provider metadata. For example: `urn:federation:mywsfedidp`

Federation Service endpoint URL	The last path component of this URL must the match `metaAlias` in the identity provider extended metadata. For example:
	
	`https://amhost(:amsecureport)/fam/WSFederationSer`
	
	`/metaAlias/mywsfedidp`

Account Partner Verification Certificate	Import the OpenSSO token signing certificate that you copied to the `adfsresource` machine.

13 Delete all cookies in your browser and go to the sample claims-aware application at `https://adfsweb.treyresearch.net:8081/claimapp/`.

You should see the OpenSSO Enterprise identity provider listed in the drop down list. Select the OpenSSO identity provider. You will be redirected to the standard OpenSSO Enterprise login screen. After logging in, you will be redirected back to the sample application

14 Click the Sign Out link to do a single logout.

Check that you are logged out by trying the `https://adfsweb.treyresearch.net:8081/claimapp/` URL again. You should be redirected to the OpenSSO login page, demonstrating that neither ADFS or OpenSSO have an active session for the browser.

The realm choice is stored in a persistent cookie. If you close and restart the browser, return to `https://adfsweb.treyresearch.net:8081/claimapp/`. You should directly proceed to the OpenSSO Enterprise login page.

CHAPTER 9

Identity Web Services

OpenSSO Enterprise 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. OpenSSO Enterprise 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 OpenSSO Enterprise Authentication Service discussed in "About Identity Web Services" in *Sun OpenSSO Enterprise 8.0 Technical Overview*.

Authentication Web Service Attribute

The Authentication Web Service attributes are *global* attributes. The value is carried across the OpenSSO Enterprise configuration and inherited by every organization.

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 9–1 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.

Challenge Cleanup Interval

Specifies cleanup interval (in seconds) in the default CRAM-MD5 mechanism handler implementation class `com.sun.identity.liberty.ws.authnsvc.mechanism.CramMD5MechanismHandler`. The internal thread will start to cleanup the challenge map based on this value.

Transform Classes

Specifies the transform name to the implementation class mapping. Values are comma separated with a pipe (`|`) as delimiter for the transform name and implementation class name. For example, `name1|class1, name2|class2`.

PLAIN Mechanism Handler Authentication Module

Specifies the default authentication module used for the PLAIN mechanism handler. The default value is the Data Store authentication module.

CRAM-MD5 Mechanism Handler Authentication Module

This specifies the default authentication module used for the CRAM MD5 mechanism handler. The default value is the Data Store authentication module

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 OpenSSO Enterprise configuration and inherited by each configured organization.

The attributes are:

- "ResourceID Mapper" on page 230
- "Authorizer" on page 230
- "Attribute Mapper" on page 231
- "Provider ID" on page 231
- "Name Scheme" on page 231
- "Namespace Prefix" on page 231
- "Supported Containers" on page 231
- "PPLDAP Attribute Map List" on page 231
- "Require Query PolicyEval" on page 232
- "Require Modify PolicyEval" on page 232
- "Extension Container Attributes" on page 232
- "Extension Attributes Namespace Prefix" on page 233

- “Service Update” on page 233
- “Service Instance Update Class” on page 233
- “Alternate Endpoint” on page 233

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, OpenSSO Enterprise 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, OpenSSO Enterprise 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:

- Require Query PolicyEval
- Require Modify PolicyEval

Attribute Mapper

The value of this attribute defines the class for mapping a Liberty Personal Profile Service attribute to an OpenSSO Enterprise 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 OpenSSO Enterprise 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.

PPLDAP Attribute Map List

Each identity attribute defined in the Liberty Personal Profile Service maps one-to-one with a OpenSSO Enterprise LDAP attribute. For example, `JobTitle=sunIdentityServerPPEmploymentIdentityJobTitle` maps the Liberty `JobTitle` attribute to the OpenSSO Enterprise `sunIdentityServerPPEmploymentIdentityJobTitle` attribute.

The value of this attribute is a list that specifies the mappings. The list is used by the attribute mapper defined in "Attribute Mapper" on page 231, 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.

Require Query PolicyEval

If selected, this option requires that a policy evaluation be performed for Liberty Personal Profile Service queries. For more information, see "Authorizer" on page 230.

Require Modify PolicyEval

If selected, this option requires that a policy evaluation be performed for Liberty Personal Profile Service modifications. For more information, see "Authorizer" on page 230.

Extension Container Attributes

The Liberty Personal Profile Service allows you to specify extension attributes that are not defined in the Liberty Alliance Project specifications. The values of this attribute specify a list of extension container attributes. All extensions should be defined as:

`/PP/Extension/PPISExtension [@name='`*extensionattribute*`']`

The following sample illustrates an extension query expression for `creditcard`, an extension attribute.

EXAMPLE 9–1 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"
  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">
```

EXAMPLE 9–1 Extension Query for `creditcard` *(Continued)*

```
              <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 "Extension Container Attributes" on page 232. 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.

Discovery Service

The Discovery Service is a framework for describing and discovering identity web services. It allows a requesting entity, such as a web service client, 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 SAMLv2, 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 OpenSSO Enterprise configuration and inherited by every configured organization. The Discovery Service attributes are:

- "Provider ID" on page 234
- "Supported Authentication Mechanisms" on page 234
- "Supported Directives" on page 234
- "Policy Evaluation for Discovery Lookup" on page 235
- "Policy Evaluation for Discovery Update" on page 235
- "`Authorizer` Plug-in Class" on page 235
- "Entry Handler Plug-in Class" on page 235
- "Classes For `ResourceIDMapper` Plug-in" on page 236
- "Authenticate Response Message" on page 236
- "`SessionContextStatement` for Bootstrapping" on page 236
- "Encrypt `NameIdentifier` in Session Context for Bootstrapping" on page 236
- "Implied Resource" on page 236
- "Resource Offerings for Bootstrapping" on page 237

Provider ID

This attribute takes a URI that points to the Discovery Service. Use the format *protocol*`://`*host*`:`*port*`/opensso/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.

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. .

TABLE 9–2 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.
`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`.

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`.

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*`/opensso/Liberty/disco`. See "Provider ID" on page 231.
- *ID Mapper* takes as a value the class name and path of the implementing class.

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.

Name Identifier Mapper

Defines the class and path that implements the `NameIdentifierMapper` interface. It is used to map user's Name Identifier from one provider to another.

Global Entry Handler Plug-in Class

Defines the class and path that implements the `DiscoEntryHandler` interface. It is used to get and set Disco Entries for a user stored in a realm. When an implied resource is used in a discovery service request, this implementation is used to perform the operation.

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.

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 OpenSSO Enterprise, 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 supports only the `DiscoveryLookup` operation using the discovery protocol. (Updates can still be done using the OpenSSO Enterprise Console.) More information is provided in the following sections:

- "Storing Resource Offerings as User Attributes" on page 237
- "Storing Resource Offerings as Dynamic Attributes" on page 240
- "Storing a Resource Offering for Discovery Service Bootstrapping" on page 242

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 OpenSSO Enterprise 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 Discovery Service.**

7 **Click Add.**

8 **(Optional) Type a value for the Resource ID Attribute.**

This field defines an identifier for the resource offering.

9 **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.

---

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.

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`.

---

12 **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.

---

13 **Click New 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.

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 Save to save the configuration.**

Storing Resource Offerings as Dynamic Attributes

Due to the repetition inherent in storing discovery entries as user attributes, OpenSSO Enterprise has established the option of storing a discovery entry as a dynamic attribute within a realm. The 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.

▼ 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 OpenSSO Enterprise 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.**

6 **Click Next.**

7 **Click Discovery Service to add a resource offering to the service.**

8 **Click Add to add a resource offering.**

9 **(Optional) Enter a description of the resource offering in the Description field.**

10 **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`.

11 **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.

---

12 **Click New 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.**

13 **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.

14 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.

15 Click OK.

16 Click Close to close the Discovery Resource Offering window.

17 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 "Resource Offerings for Bootstrapping" on page 237.

▼ To Store a Resource Offering for Discovery Service Bootstrapping

1 In the OpenSSO Enterprise 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.

---

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. .

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 OpenSSO Enterprise configuration and inherited by every organization.

The SOAP Binding Service attributes are as follows:

- "Request Handler List" on page 245
- "Web Service Authenticator" on page 245
- "Supported Authentication Mechanisms" on page 246

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 OpenSSO Enterpriseis 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*`/opensso/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`

Enforce Only Known Providers

If enabled, this property will enforce the `ProviderID` header sent in a SOAP message to ensure that the provider exists in the system. If it does not, the request will be rejected. If this attribute is not enabled, the check will be skipped.

Certification Alias For SSL Client Authentication

Value is set during installation. Client certificate alias that will be used in SSL connection for Liberty SOAP Binding.

Time Limit for Stale Message

Default value is 300000. Determines if a message is stale and thus no longer trustworthy. If the message timestamp is earlier than the current timestamp by the specified number of milliseconds, the message the considered to be stale.

Message ID Cache Cleanup Interval

Default value is 60000. Specifies the number of milliseconds to elapse before cache cleanup events begin. Each message is stored in a cache with its own message ID to avoid duplicate messages. When a message's current time less the received time exceeds thestaleTimeLimit value, the message is removed from the cache.

Supported SOAP Actors

Default value is `http://schemas.xmlsoap.org/soap/actor/next`. Specifies supported SOAP actors. Each actor must be separated by a pipe character (|).

Namespace Prefix Mapping

Default value is:

```
=S=http://schemas.xmlsoap.org/soap/envelope/|sb=urn:liberty:sb:2003-08
|pp=urn:liberty:id-sis-pp:2003-08|ispp=http://www.sun.com/identity/
liberty/pp|is=urn:liberty:is:2003-08
```

Specifies the namespace prefix mapping used when marshalling a JAXB content tree to a DOM tree. The syntax is `prefix=namespace|prefix=namespace|...`

JAXB Package List

Specifies JAXB package list used when constructing `JAXBContext`. Each package must be separated by a colon (:).

Liberty Identity Web Service Version

This property determines the Liberty ID-WSF framework when the framework cannot determine from the in-bound message or from the resource offering when OpenSSO is acting as the WSC. Values can be 1.0 or 1.1. The default is 1.1.

Liberty Interaction Service

The Liberty Interaction Service allows the user to interact during web services communication for any authorization. .

The Liberty Interaction Service is configurable through the OpenSSO Enterprise console at Configuration>Global>Liberty ID-WSF Interaction Service. See “Liberty Interaction Service” in *Sun OpenSSO Enterprise 8.0 Administration Reference* for attribute definitions.

Liberty ID-WSF Security Service

The Liberty ID-WSF Security Service is configurable through the OpenSSO Enterprise console at Configuration>Global>Liberty ID-WSF Interaction Service. See "Liberty ID-WSF Security Service" in *Sun OpenSSO Enterprise 8.0 Administration Reference* for attribute definitions.

CHAPTER 10

SAML 1.x Administration

Security Assertion Markup Language (SAML) is an open-standard protocol that uses a L framework to exchange security information between an *authority* and a *trusted* partner site. The security information concerns itself with a subject's authentication status, access authorization and attribute information. (A person identified by an email address is a subject as might be a printer.) A SAML authority (also referred to as the *asserting party*) is a platform or application that has been integrated with SAML API, allowing it to relay security information. For example, asserting that a subject has been authenticated into its system by passing the required attributes. Trusted partner sites receive the security information and rely on its authenticity.

SAML Attributes

All attributes associated with SAML1.x can be configured and defined in the OpenSSO Enterprise console. This section describes how to create and configure the SAML1.x service.

Note – By default character escaping is enabled so that you can use the following special characters in SAML 1.x attributes in the OpenSSO Enterprise console:

- &
- <
- >
- “
- '
- /

To disable character escaping:

1. In the OpenSSO Enterprise Console, go to Configuration>Servers and Sites>Default Server Setting>Advanced.
2. Enter the `com.sun.identity.saml.escapeattributevalue` as the key, and `false` as the value.
3. Restart the server.

If you wish to enable character escaping, change the value to `true` and restart the server.

The following SAML attributes can be configured for your implementation by clicking the Local Site Properties button:

- “Target Specifier” on page 250
- “Site Identifiers” on page 251
- “Trusted Partners” on page 251
- “Target URLs” on page 255
- “Assertion Timeout” on page 256
- “Assertion Skew Factor for `notBefore` Time” on page 256
- “Artifact Timeout” on page 257
- “SAML Artifact Name” on page 257
- “Sign SAML Assertion” on page 257
- “Sign SAML Request” on page 257
- “Sign SAML Response” on page 257
- “Attribute Query” on page 257

Target Specifier

This attribute assigns a name to the destination site URL used in SAML redirects. The default is `TARGET`. Only sites configured in the Trusted Partners attribute can be specified as a `TARGET`.

Site Identifiers

For information about site identifiers and to see the procedure for configuring a site identifier, see the following section.

▼ To Configure a Site Identifier

The Site Identifier defines any site hosted by the server on which OpenSSO Enterprise is installed. A default value and an automatically generated Site ID are defined for the host during installation. Multiple entries are possible. For example, load balancing or multiple instances of OpenSSO Enterprise sharing the same data store would all need to be defined. The starting point is the Site Identifiers attribute on the SAML screen under the Federation interface. Site IDs are defined in the Servers and Sites configuration screen. For more information, see "Servers and Sites" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

1. **Click New to add a new site identifier or click on the name of a configured site identifier to modify its profile.**

   The Site Identifier attributes are displayed.

2. **Provide values for the Site Identifier attributes based on the following information:**

Instance ID	The value of this property is *protocol* `://`*host*`:` *port*. If configuring SAML for SSL (in both the source and destination site), ensure that the protocol defined here is `https//`.
Site ID	The site ID is an identifier generated for each site (although the value will be the same for multiple servers behind a load balancer). There is a class in the `com.sun.identity.saml.common` package that can be used to generate this identifier manually, if needed. Type the following at the command line: `% java -classpath` *FM-classpath* `com.sun.identity.saml.common.SAMLSiteID` *protocol*`://`*host*`:`*port*
Issuer Name	The default value of this property is *host* `:`*port*, but it could be any URI.

3. **Click OK to complete the Site Identifier configuration.**

4. **Click Save on the Local Site Properties page to complete the SAML configuration.**

Trusted Partners

For information on trusted partners and to see the procedure for configuring a new Trusted Partner, see the following section.

▼ Trusted Partners: Selecting Partner Type and Profile

This attribute defines any trusted partner (remote to the server on which OpenSSO Enterprise is installed) that will be communicating with OpenSSO Enterprise.

Note – The trusted partner site must have a prearranged trust relationship with one or more of the sites configured in the Site Identifiers attribute.

The first step in configuring a trusted partner is to determine the partner's role in the trust relationship. A trusted partner can be a source site (one that generates a single sign-on assertion) or a destination site (one that consumes a single sign-on assertion). For example, if the partner is the source site, this attribute is configured based on how it will send assertions. If the partner is the destination site, this attribute is configured based on the profile in which it will be receiving assertions. Following is the first part of the procedure for configuring a trusted partner. The starting point is the SAML screen under Federation.

Note – To edit or duplicate the attributes of a trusted partner profile, click the appropriate button in the Actions column next to the configured trusted partner name.

---

1. **Select the role (Destination or Source) of the partner site you are configuring by checking the appropriate profile that will be used to communicate with it.**

   You may choose *Web Browser Artifact Profile* or *Web Browser Post Profile* for either Destination, Source or both, or *SOAP Query* for Destination only. The choices made dictate which of the attributes in the following steps need to be configured.

   Note – Click Edit to change the role of the partner site if you are modifying an existing trusted partner.

   ---

2. **Click Next.**

▼ Trusted Partners: Configuring Trusted Partner Attributes

Following is the second part of the procedure for configuring a trusted partner. Based on the roles selected in the first part, any of the sub-attributes listed in the following sections may need to be defined.

Note – If you reached this page by clicking Edit or Duplicate on the SAML configuration screen under Federation, modify the trusted partner profile based on the steps below and click Save to change the values. Click Save on the SAML Profile page to complete the modification.

---

1 **Type in values for the Common Settings sub-attributes.**

Name	Can be any string, such as an organization name.
Source ID	This is a 20 byte sequence (encoded using the Base64 format) that comes from the partner site. It is generally the same value as that used for the Site ID attribute when configuring the Site Identifiers attribute.
Target	This is the domain of the partner site (with or without a port number). If you want to contact a web page that is hosted in this domain, the redirect URL is picked up from the values defined in the Trusted Partner attribute. **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.
SAML URL	The URL that points to the servlet that implements the Web Browser Artifact Profile.
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 `PartnerSiteAttributeMapper` interface. If no class is defined, no attributes will be included in the assertion.
Name Identifier Mapper	The class that defines how the subject of an assertion is related to an identity at the destination site. An account mapper needs to be implemented from the `com.sun.identity.saml.plugins.PartnerAccountMapper` interface. The default is `com.sun.identity.saml.plugins.DefaultAccountMapper`. If no class is defined, no attributes will be included in the assertion.
Version	The SAML version used (`1.0` or `1.1`) to send SAML requests. To change the version or protocol, click on the Local Site Properties button and change the Protocol and Assertion attributes as necessary.

Signing Certificate Alias	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.

2 **Type in values for the Destination sub-attributes.**

Artifact: SAML URL	The URL that points to the servlet that implements the Web Browser Artifact Profile.
Post: Post URL	The URL that points to the servlet that implements the Web Browser POST Profile.
Host List	A list of the IP addresses, the DNS host name, or the alias of the client authentication certificate used by the partner. This is configured for all hosts within the partner site that can send requests to this authority. This list helps to ensure that the requestor is indeed the intended receiver of the artifact. If the requester is defined in this list, the interaction will continue. If the requester's information does not match any hosts defined in the host list, the request will be rejected.

3 **Type in values for the Source sub-attributes.**

Artifact: SOAP URL	The URL to the SAML SOAP Receiver.
Authentication Type	Authentication types that can be used with SAML:

- Specify `None` if the URL to the SAML SOAP receiver is accessed using HTTP, and the SAML SOAP receiver is not protected by HTTP basic authentication and/or SSL.
- Specify `Basic` if the URL to the SAML SOAP receiver is accessed using HTTP, and the SAML SOAP receiver is protected by HTTP basic authentication.
- Specify `SSL` if the URL to the SAML SOAP receiver is accessed using HTTPS, and the SAML SOAP receiver is not protected by HTTP SSL.
- Specify `SSL with Basic` if the URL to the SAML SOAP receiver is accessed using HTTPS, and the SAML SOAP receiver is not protected by `BASIC AUTH WITH SSL`.

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 OpenSSO Enterprise configuration. You do, however, supply the HTTP basic authentication user ID and password in the OpenSSO Enterprise 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`.
User	When `BASICAUTH` is chosen as the Authentication Type, the value of this attribute defines the user identifier of the partner being used to protect the partner's SOAP receiver.
User's Password	When `BASICAUTH` is chosen as the Authentication Type, the value of this attribute defines the password for the user identifier of the partner being used to protect the partner's SOAP receiver.
User's Password (reenter)	Reenter the password defined previously.

4 **Type a value for the Post sub attribute.**

Issuer	The creator of a generated assertion. The default syntax is *hostname*`:` *port*.

5 **Click Finish.**

6 **Click Save on the SAML Profile page to complete the configuration.**

Target URLs

If the Target URL received using either profile is listed as a value of this attribute, the received assertions will be sent to the Target URL using an HTTP FORM POST.

Note – To edit or duplicate the attributes of a trusted partner profile, click the Local Site Properties button and click New in the Target URL table.

▼ To Configure a Target URL

The following sub attributes can be defined (or modified) for each Target URL that will receive assertions using POST.

1 **Click New to add a new target URL.**

The Add New Post to Target URL page is displayed. You can also reach this page by selecting the Edit or Duplicate button of a defined Target URL.

2 **Type values for the attributes.**

Protocol	Choose either `http` or `https`.
Server Name	The name of the server on which the TARGET URL resides, such as `www.sun.com`.

Port	This attribute contains the port number such as `58080`.
Path	This attribute contains the URI such as `/opensso/console`.

3 **Click OK to complete the Target URL configuration.**

4 **Click Save on the SAML Profile page to complete the SAML configuration.**

Default Protocol Version

This attribute specifies the default SAML protocol version this entity uses to communicate with remote partners. Default value is 1.1.

Default Assertion Version

This attribute specifies the default SAML assertion version this entity uses to communicate with remote partners.

Remove Assertion

If enabled (true), the SAML entity will remove assertions from memory once they are used. Otherwise, all assertions remain in memory until expiration. It is enabled by default.

Assertion Timeout

This attribute specifies the number of seconds before a timeout occurs on an assertion. The default is 420.

Assertion Skew Factor for `notBefore` Time

This attribute is used to calculate the `notBefore` time of an assertion. For example, if `IssueInstant` is `2002-09024T21:39:49Z`, and `Assertion Skew Factor For notBefore Time` is set to 300 seconds (180 is the default value), the `notBefore` attribute of the conditions element for the assertion would be `2002-09-24T21:34:49Z`.

Note – The total valid duration of an assertion is defined by the values set in both the `Assertion Timeout` and `Assertion Skew Factor For notBefore Time` attributes.

Artifact Timeout

This attribute specifies the period of time an assertion that is created for an artifact will be valid. The default is 400.

SAML Artifact Name

This attribute assigns a variable name to a SAML artifact. The artifact is bounded-size data that identifies an assertion and a source site. It is carried as part of a URL query string and conveyed by redirection to the destination site. The default name is `SAMLart` . Using the default `SAMLart`, the redirect query string could be `http://`*host:port* `/`*deploy-URI*`/`
`SamlAwareServlet?TARGET=`*target-URL* `/&SAMLart=artifact123`.

Sign SAML Assertion

This attribute specifies whether all SAML assertions will be digitally signed (`XML DSIG`) before being delivered. Selecting the check box enables this feature.

Sign SAML Request

This attribute specifies whether all SAML requests will be digitally signed (`XML DSIG`) before being delivered. Selecting the check box enables this feature.

Sign SAML Response

This attribute specifies whether all SAML responses will be digitally signed (`XML DSIG`) before being delivered. Selecting the check box enables this feature.

Note – All SAML responses used by the Web Browser POST Profile will be digitally signed whether this option is enabled or not enabled.

Attribute Query

Defines the list of the Attribute Query Profile for X. 509 subjects only.

SAML Operations

This section contains procedures illustrating how to use the OpenSSO Enterprise SAML Service.

Setting Up SAML Single Sign-on

The following procedures explain how to configure and access instances of OpenSSO Enterprise 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 `Coopkie Name` property in Configuration>Servers and Sites>Security, located in the OpenSSO Enterprise console.

This section contains the following procedures:

- "To Set Up SAML Single Sign-on" on page 258
- "To Verify the SAML Single Sign-on Configurations" on page 261

▼ To Set Up SAML Single Sign-on

This procedure assumes the following values:

Deployment URI	`opensso`
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 console running at `exampleB.com` as the default administrator, `amadmin`.

b. Click the Federation tab.

c. Click the SAML button.

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 OpenSSO Enterprise.**

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 console running at `exampleA.com` as the default administrator, `amadmin`.**

b. **Click the Federation tab.**

c. **Click New under Trusted Partners.**

This takes you to the *Select trusted partner type and profile* page.

d. **Check Artifact and Post under Destination and click Next.**

This takes you to the *Add New Trusted Partner* page.

e. **Set the values of the following attributes to configure machine B as a trusted partner of machine A:**

name	Type the name of the trusted partner. The name will be displayed in the trusted partner table.
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/opensso/SAMLAwareServlet`
HOST LIST	`exampleB.com`
POST URL	`http://exampleB.com:58080/opensso/SAMLPOSTProfileServlet`

f. **Click Finish.**

g. **Click Save.**

h. **Click the sole entry listed under Site Identifiers.**

This takes you to the *Edit site identifier* page.

i. **Write down or copy the value of the Site ID attribute.**

j. **Click Cancel to go to previous page.**

k. **Log out of OpenSSO Enterprise.**

3 **Configure the destination site (machine B) to trust the source site (machine A).**

a. **Login to the OpenSSO Enterprise console running at** `exampleB.com` **as the default administrator,** `amadmin`**.**

b. **Click the Federation tab.**

c. **Click New under Trusted Partners.**

This takes you to the *Select trusted partner type and profile* page.

d. **Check Artifact and Post under Source and click Next.**

This takes you to the *Add New Trusted Partner* page.

e. **Set the values of the following attributes to configure machine A as a trusted partner of machine B:**

Name	Type the name of the trusted partner. This will appear in the Trusted Partners table.
Source ID	Type the Site ID you copied from the source site, machine A, in the previous step.
SOAP URL	`http://exampleA.com:58080/opensso/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.

f. **Click Finish.**

g. **Click Save.**

h. **Log out of OpenSSO Enterprise.**

▼ To Verify the SAML Single Sign-on Configurations

1 **Login to the OpenSSO Enterprise 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/opensso/SAMLAwareServlet?TARGET=`*exampleB.com_Target_URL*

- **Access the following URL to use the SAML POST profile:**

  `http://exampleA.com:58080/opensso/SAMPOSTProfileServlet?TARGET=`*exampleB.com_Target_*

Note – XML signing must be enabled before running the SAML POST profile. .

---

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/opensso/UI/Login`. If the administrator successfully accesses the OpenSSO Enterprise console on the destination site without manual authentication, an `SSOtoken` has been created for the principal on the destination site and single sign-on has been properly established.

PART III

Directory Management and Default Services

This is part three of the *Sun OpenSSO Enterprise 8.0 Administration Guide*. The Directory Management chapter describes how to manage LDAP objects when OpenSSO Enterprise is deployed in Legacy Mode. The other chapters describe how to configure and use some of OpenSSO Enterprise's default services. This part contains the following chapters:

- Chapter 11, "Directory Management"
- Chapter 12, "Current Sessions"
- Chapter 13, "Password Reset Service"
- Chapter 14, "Logging Service"
- Chapter 15, "Backing Up and Restoring Configuration Data"

CHAPTER 11

Directory Management

The Directory Management tab is only displayed when you install OpenSSO Enterprise in Legacy mode. This directory management feature provides an identity management solution for Sun Directory Server-enabled OpenSSO Enterprise deployments.

For more information on the Legacy Mode installation option, see the *Sun Java Enterprise System 5 Update 1 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 OpenSSO Enterprise 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 OpenSSO Enterprise. The Directory Server objects you can manage are:

- “Organizations” on page 265
- “Containers” on page 268
- “Group Containers” on page 269
- “Groups” on page 270
- “People Containers” on page 273
- “Users” on page 274
- “Roles” on page 277

Organizations

An *Organization* represents the top-level of a hierarchical structure used by an enterprise to manage its departments and resources. Upon installation, OpenSSO Enterprise dynamically creates a top-level organization (defined during installation) to manage the OpenSSO 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/opensso/UI/Login?org=exampleorg`
	
	`http://machine.example.com/opensso/UI/Login?org=abc`
	
	`http://machine.example.com/opensso/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/opensso/UI/`

	`Login?org=exampleorg`
	
	`http://machine.example1.com/opensso/`
	
	`UI/Login?org=exampleorg`
	
	`http://machine.example2.com/opensso/`
	
	`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 `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 `http://www.example.com`. 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.

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

OpenSSO Enterprise 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 "Modifying Policies and Referrals" on page 114.

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 OpenSSO Enterprise container entry and the OpenSSO Enterprise 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.

OpenSSO Enterprise 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 – OpenSSO Enterprise 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 OpenSSO Enterprise 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 asterisk 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 members 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 members 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, OpenSSO Enterprise 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 asterisk 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 members you wish to remove and choose Remove Members.
Delete Members	This action will permanently delete the member you select. Select the members you wish to delete and choose Delete Members.

To Add a Group to a Policy

OpenSSO Enterprise 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 "Modifying Policies and Referrals" on page 114.

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 OpenSSO Enterprise 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 OpenSSO Enterprise. 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 OpenSSO Enterprise. 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 OpenSSO Enterprise, 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 OpenSSO Enterprise 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 OpenSSO Enterprise. Only active users can authenticate through OpenSSO Enterprise. The default value is Active. Either of the following can be selected from the pull-down menu: . ▪ Active: The user can authenticate through OpenSSO Enterprise. ▪ Inactive: The user cannot authenticate through OpenSSO Enterprise, but the user profile remains stored in the directory.

Note – Changing the user status to Inactive only affects authentication through OpenSSO Enterprise. The Directory Server uses the *nsAccountLock* attribute to determine user account status. User accounts inactivated for OpenSSO Enterprise authentication can still perform tasks that do not require OpenSSO Enterprise. To inactivate a user account in the directory, and not just for OpenSSO Enterprise authentication, set the value of *nsAccountLock* to true. If delegated administrators at your site will be inactivating users on a regular basis, consider adding the *nsAccountLock* attribute to the OpenSSO Enterprise User Profile page. See the *Sun OpenSSO Enterprise 8.0 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

OpenSSO Enterprise 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 “Modifying Policies and Referrals” on page 114.

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 OpenSSO Enterprise 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 OpenSSO Enterprise console. Other Directory Server types can be used in a policy's subject definition. For more information on policy subjects, see “Creating Policies and Referrals” on page 107.

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.

OpenSSO Enterprise uses roles to apply access control instructions. When first installed, OpenSSO Enterprise 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 – OpenSSO Enterprise 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 OpenSSO Enterprise, 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 OpenSSO Enterprise 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 OpenSSO Enterprise 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 OpenSSO Enterprise 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 OpenSSO Enterprise, 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 OpenSSO Enterprise 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 OpenSSO Enterprise 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 OpenSSO Enterprise 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

OpenSSO Enterprise 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 "Modifying Policies and Referrals" on page 114.

CHAPTER 12

Current Sessions

This chapter describes the session management features of OpenSSO Enterprise. 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 OpenSSO Enterprise.

Session Management

The Session Management frame displays the name of the OpenSSO Enterprise that is currently being managed.

Session Information

The Session Information window displays all of the users who are currently logged into OpenSSO Enterprise, 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 Invalidate Session.**

   To terminate multiple sessions, select the individual sessions and click the Invalidate Sessions button.

CHAPTER 13

Password Reset Service

OpenSSO Enterprise provides a Password Reset service to allow users to reset their password for access to a given service or application protected by OpenSSO Enterprise. 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.

Your configuration must meet the following prerequisites in order for the Password Reset service to work:

- Valid email address for the users who want their password to be reset through this service.
- Valid SMTP service host and port configured in the Access Manager Server.
- The user data store must be a supported instance of Sun Directory Server Enterprise Edition or OpenDS For more information see the *Sun OpenSSO Enterprise 8.0 Installation and Configuration Guide*.

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 Password Reset and click Next**

The Password Reset service attributes will be displayed. For attribute definitions, see the online help or "Password Reset" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

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.

5 **Enable Force Change Password After Reset.**

This optional step is the key part for the password reset service to force the user to change their password after a password reset. If this is not enabled then password reset service will ignore the `pwdreset` control from the Directory Server. This particular option is meaningful only if the password policy in the Directory Server is enabled to force the users to change the password upon an administrator-controlled password reset occurrence, so you must make a configuration change for the Directory Server.

You can enable Force Change Password After Reset globally by selecting it in the global Password Reset attributes, or you can select if for individual users by selecting a User profile, clicking on Password Reset Options, and enabling the attribute.

6 **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 OpenSSO Enterprise, 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`**. 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 or "Password Reset" in *Sun OpenSSO Enterprise 8.0 Administration Reference* 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, or "Password Reset" in *Sun OpenSSO Enterprise 8.0 Administration Reference* for a description of the service attributes.

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 the *Sun Java System 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 OpenSSO Enterprise's administration accounts, such as `amldap`, `dsame`, and `puser`. When the OpenSSO Enterprise administrator's account password expires, and an end-user is logged in, the user will receive the password change screen. However, OpenSSO Enterprise 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.

▼ Example: To Create a Password Policy in Directory Server for Force Password Change After Reset

The following example shows how to configure the Directory Server to work with the Force Password Change After Reset attribute. This involves creating a password policy and assigning to it to a range of user identities.

This sample password policy forces users to change their password after an administrator reset (Any password change that is not done by the self modify is considered as password reset, meaning that the attribute `pwdreset` will be true.)

1. **Type the following text in a file called** `passwdPolicy.ldif`**.**

   Change *dc=red,dc=sun,dc=com* in the text to the actual root suffix of the deployed Directory Server.

   ```
   dn: cn=AMUsersPasswordPolicy,dc=red,dc=sun,dc=com
   objectClass: top
   objectClass: pwdPolicy
   objectClass: LDAPsubentry
   cn: AMUsersPasswordPolicy
   pwdMustChange: TRUE
   pwdattribute: userPassword
   ```

2. **Execute the following command:**

   ```
   ldapmodify -D"cn=directory manager" -w admin123 -c -a -f passwdPolicy.ldif
   ```

   This will add the password policy to the Directory Server.

3. **Assign this policy to user identities. For example, enter the following text in to a file called** `AddPwdPolicy.ldif`**:**

   ```
   dn:uid=example_user,ou=people,dc=red,dc=iplanet,dc=com
   changetype:modify
   add: pwdPolicySubentry
   pwdPolicySubentry:cn=AMUsersPasswordPolicy,dc=red,dc=iplanet,dc=com
   ```

4. **Execute the following command:**

   ```
   ldapmodify -D"cn=directory manager" -w admin123 -c -a -f AddPwdPolicy.ldif
   ```

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 OpenSSO Enterprise console in order to customize their secret questions.

▼ To Customize Password Reset

1. **The user logs into the OpenSSO Enterprise 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, OpenSSO Enterprise 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* /*uri*/`password` (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.

CHAPTER 14

Logging Service

OpenSSO Enterprise 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" on page 295
- "OpenSSO Enterprise Logs" on page 296
- "Logging Features" on page 298
- "Debug Files" on page 301

Log Files

The log files record a number of events for each of the services. These files should be checked by the administrator on a regular basis. The default directory for the log files is *ConfigurationDirectory*`/`*uri*`/log/`, where *ConfigurationDirectory* is the configuration directory, and *uri* is the OpenSSO deployment URI specified during OpenSSO configuration and deployment time. These tags are interpreted at run time, such that each deployed OpenSSO instance has its own logging directory. This is particularly useful when there are multiple OpenSSO instances per system. The log file directory can be configured in the Logging Service by using the OpenSSO Enterprise console or `ssoadm` command-line utility. An absolute path may also be configured as the log file directory.

See Chapter 15, "Recording Events with the Logging Service," in *Sun OpenSSO Enterprise 8.0 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 OpenSSO Enterprise Console or "Logging" in *Sun OpenSSO Enterprise 8.0 Administration Reference*.

OpenSSO Enterprise 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 OpenSSO Enterprise services. Flat log files are appended with the `.error` or `.access` extension. Database column names end with `_ERROR` or `_ACCESS` for an Oracle database, 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" on page 296
- "Console Logs" on page 296
- "Authentication Logs" on page 297
- "Federation Logs" on page 297
- "Policy Logs" on page 297
- "Agent Logs" on page 297
- "SAML Logs" on page 297
- "`ssoadm` Logs" on page 297

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 OpenSSO Enterprise console logs record the creation, deletion, and modification of identity-related entities, policies, and services including, among others, realms, users, policies, and groups. It also records modifications of user attributes including passwords and the addition of users to or removal from groups. Additionally, the console logs record 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 a circle of trust 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 OpenSSO Enterprise 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`.

`ssoadm` Logs

The `ssoadm` logs record events that occur during operations using the `ssoadm` command line tool. These include operations that have OpenSSO administration console equivalents. The `ssoadm` command line logs are located in the logging directory specified when running the setup script for the administration tools unzipped from `ssoAdminTools.zip`. The main logs are prefixed with `ssoadm`; other task-related log files are also have `access` and `error` suffixes.

Logging Features

The Logging Service has a number of special features which can be enabled for additional functionality.

- "Secure Logging" on page 298
- "Logging Level Attributes and Properties" on page 299
- "Database Logging" on page 300
- "Remote Logging" on page 300

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 preregistered certificate configured by the system administrator. A 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 two methods to enable secure logging; through a through a Java Cryptography Extension (JCE) provider and through a Java Security Server (JSS) provider.

Note – Secure logging can only be used for flat files. This option does not work for Database (DB) logging.

▼ To Enable Secure Logging through a JSS Provider

1 **Create a certificate with the name `Logger` and install it in the key store specified by the Logging Service configuration's Logging Certificate Store Location.**

The key store's password is expected to be the same as the top-level administrator password. The default location set during OpenSSO Enterprise configuration time is *ConfigurationDirectory*`/`*uri*`/Logger.jks/`, where *ConfigurationDirectory* is the configuration directory, and *uri* is the OpenSSO deployment URI specified during OpenSSO configuration and deployment time. These tags are interpreted at run time, such that each deployed OpenSSO instance has its own key store. It is particularly useful when there are multiple OpenSSO instances per system. Information on getting certificates can be found in "Obtaining Secure Socket Layer Certificates" in *Deployment Example: Single Sign-On, Load Balancing and Failover Using Sun OpenSSO Enterprise 8.0*.

2 **Turn on Secure Logging in the Logging Service configuration using the OpenSSO Enterprise administration console and save the change. The administrator can also modify the default values for the other Logging Service attributes.**

If the logging directory is changed from the default `/log` directory, make sure that the directory is writable by the user ID and that the OpenSSO Enterprise's web application is running. Also set the directory's permissions to 0700, as the logging service will create the directory, if it does not exist, with permissions set to 0755.

3 **Verify Secure Log Archives.**

To detect unauthorized changes or tampering of the secure logs, look for error messages that are written by the Logging Service's periodic verification process to *ConfigurationDirectory*`/`*uri*`/debug/amLog`. To manually check for tampering, run the `amverifyarchive` command-line utility, which is included in the `ssoAdminTools.zip` file.

4 **Changing from a JCE Provider to a JSS Provider**

The default secure log helper provider is the JCE provider, `com.sun.identity.log.secure.impl.SecureLogHelperJCEImpl`, as specified by the `iplanet-am-logging-secure-log-helper` attribute in the iPlanetAMLoggingService's schema. Refer to the `opensso/xml/amLogging.xml` file from the `opensso.zip` file.

To change to the JSS provider, use the `ssoadm` command-line utility:

```
./ssoadm set-attr-defs --servicename iPlanetAMLoggingService --schematype
global --attributevalues iplanet-am-logging-secure-log-helper-class-name=
com.sun.identity.log.secure.SecureLogHelperJSSImpl --adminid amadmin
--password-file amadminpass
```

To verify the change:

```
./ssoadm get-attr-defs --servicename iPlanetAMLoggingService --attributenames
iplanet-am-logging-secure-log-helper-class-name --schematype global --adminid
amadmin --password-file amadminpass
```

Logging Level Attributes and Properties

There are attributes in the Logging Service configuration that affect logging output. The Log Status can be set to Inactive to disable all logging output. The Logging Level can be set to one of the `java.util.logging.Level` values other than the default INFO to get more or less detailed logging output.

Additionally, an individual log file's logging level can be specified in the Configuration > Servers and Sites > *server-name* > Advanced page. For a given log file, for exampleam`SAML.access`, add a property name, `iplanet-am-logging.amSAML.access.level` with Property Value FINER (or any of the `java.util.logging.Level`values). The logging level specified here for the log file will take precedence over the Logging Service's Logging Level setting.

Database Logging

This feature provides logging to Oracle or MySQL databases. No special coding is required to enable this feature. In the Logging Service configuration (Configuration > System > Logging), set the Logging Type to DB, set the Database User Name, Database User Password, and Database Driver Name. `oracle.jdbc.driver.OracleDriver` is the default driver name set for Oracle. For MySQL, it is typically `com.mysql.jdbc.Driver`. Be sure to put the JDBC driver's `.zip` or `.jar` file in the OpenSSO Enterprise web application's classpath (for example, `WEB-INF/lib` or `jre/lib/ext`).

The DB Failure Memory Buffer Size specifies how many records per table to buffer if the connection to the database fails. If more records are queued before the connection is reestablished, older records will be discarded.

Note – The ssoadm command line interface cannot log to the database directly. In addition to adding the JDBC driver to the web application's classpath, remove `-D"com.sun.identity.log.dir=<the_specified_log_dir>`.

Remote Logging

OpenSSO Enterprise supports remote logging. This allows a remote client application using the OpenSSO client SDK, or another OpenSSO Enterprise server (in the same Site) to use an OpenSSO Enterprise server's Logging Services.

Remote Client Logging

A remote client using the OpenSSO Enterprise client SDK may log to an OpenSSO Enterprise server. For example, the OpenSSO Enterprise client sdk samples refer to the `samples/sdk/resources/AMConfig.properties` set up by the `samples/sdk/scripts/setup.sh` script. In particular, the `com.iplanet.am.naming.url` property's value points to the target OpenSSO Enterprise server's Naming service, which provides the location of the Logging Service. In order for the remote client to successfully log to the target OpenSSO Enterprise server, the entity making the logging request must have Log Writing permission on the target OpenSSO Enterprise server.

Remote OpenSSO Enterprise Server Logging

Another OpenSSO Enterprise server may use an OpenSSO Enterprise server's Logging Services if both are in the same Site. The remote OpenSSO Enterprise server sets its Logging Service URL in the administration console (Configuration > System > Naming) to the target OpenSSO Enterprise server's Logging Service, by changing the attribute's protocol, host, port, and uri values accordingly. Logginservice does not usually need to be changed.

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 `ConfigurationDirectory/uri/debug`. This location, along with the level of the debug information, is configurable through the OpenSSO Enterprise Console. Go to Configuration > Sites and Servers >*server-name* > General and edit the Debug Directory attribute.

Debug Levels

There are several levels of information that can be recorded to the debug files. The debug level is set using the `Debug Level` attribute located at Configuration > Sites and Servers >*server-name* > General.

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: This level allows Error and Warning debug messages to be written.
4. Message: This level allows detailed code tracing.

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

By default there are separate debug files, corresponding to the major service components of OpenSSO Enterprise:

- Authentication
- CoreSystem
- amAuthContextLocal
- WebServices
- IDRepo
- Policy
- Configuration
- Session
- Federation

Debug output can be combined into one file through the administration console (Configuration > Sites and Servers > *server-name* > General. Turn the Merge Debug Files attribute to ON.

CHAPTER 15

Backing Up and Restoring Configuration Data

OpenSSO Enterprise creates and manages configuration and service management data in the embedded configuration datastore. If this data becomes corrupt or if some of it is missing, OpenSSO Enterprise will not function properly. Because of this, it is recommended that you backup your configuration datastore on a regular basis. Thus, in the event of a machine crash or other corrupting influence, you can restore the configuration data to its previous, non-corrupt state. This chapter describes the backup and restore procedures for the OpenSSO Enterprise server configuration data and contains the following sections.

- "Understanding Backup and Restore" on page 303
- "Backing Up the Configuration Datastore" on page 304
- "Restoring the Configuration Data Store" on page 305

Understanding Backup and Restore

OpenSSO Enterprise supports the following types of configuration datastores:

Caution – The procedures in this chapter do not apply to the user datastore.

- **Embedded configuration datastore**: Configuration data is stored on the server local to the instance of OpenSSO Enterprise with an exposed LDAP port. This is the default datastore installed during initial configuration of OpenSSO Enterprise.
- **Sun Directory Server configuration datastore**: Configuration data is stored in an instance of Sun Directory Server, which can be selected and configured during initial configuration of OpenSSO Enterprise.

The backup and restore procedures are dependent on the following:

- The OpenSSO Enterprise bits are not corrupted.

- The backup and restore procedures described in this document pertain only to the service configuration information stored in the defined configuration datastore. All other product files (including the bootstrap file, debug/log files, and key store files) are in the configuration directory defined during deployment (for example, `/opensso`) and are NOT in the scope of the these procedures.
- All of the restore options provided require the OpenSSO Enterprise web application to be re-configured thus, it is assumed that some configuration parameters will have to be used during the product reconfiguration. As long as the original system-generated configuration file `.configParam` (created as the result of a successful OpenSSO configuration and located in the configuration directory defined during OpenSSO Enterprise deployment; by default `/opensso`) is backed up, the information in it can be used to create a configuration file for use as input to the command-line configurator. For more information, see Chapter 5, "Configuring OpenSSO Enterprise Using the Command-Line Configurator," in *Sun OpenSSO Enterprise 8.0 Installation and Configuration Guide*.
- After OpenSSO Enterprise is successfully configured, it is assumed that no OpenDS interface or any other utility is used directly to manipulate the configuration data store.

Backing Up the Configuration Datastore

This procedure describes how to back up the data contained the configuration datastore.

▼ To Backup the Configuration Datastore

Before You Begin Make sure that the configuration datastore is running, but there are no write procedures being sent to the configuration datastore.

1 **Export the service configuration data to an XML file using the** `ssoadm` **command line utility option** `export-svc-cfg`**. For example:**

`$ cd` *sso_tools_dir*

`$ ./ssoadm export-svc-cfg –u` *username* `–f` *password file location* `–e` *key to encyrpt password* `–o` *XML-backup-file*

Note – If multiple servers are configured to share the same configuration store, the step is only required to be executed once on one of the servers.

---

2 **Move** *XML-backup-file* **(from the previous step) to a secure location.**

It is recommended to also create an MD5 hash of this file and to store it in a secure location. Use the hash file for future verification.

Restoring the Configuration Data Store

This section contains instructions to restore saved configuration data to the OpenSSO Enterprise configuration data store or the Directory Server configuration data store. Restoration of the configuration data can be done by loading an XML file or through directory replication. There are two methods to restore the configuration data for the OpenSSO configuration data store:

Loading XML	Use this option if there is only one OpenSSO Enterprise instance and it is corrupted or, multiple servers are configured to share the same configuration datastore and all instances are corrupted.
Directory Replication	Use this option in the case where multiple OpenSSO Enterprise instances are configured to share the same configuration datastore and at least one of the instances is uncorrupted.

This section contains the following procedures.

- "To Restore the Embedded Configuration Datastore by Loading XML" on page 305
- "To Restore by Replication of the OpenSSO Configuration Data store" on page 306
- "To Restore the Directory Server Configuration Datastore by Loading XML" on page 307
- "To Restore by Replication of the Directory Server Configuration Datastore" on page 308

Tip – In cases where the default OpenSSO Enterprise configuration data store is used, check its status by running `ssoadm embedded-status` on the command line. This will help to determine the proper restoration procedure to use. See Chapter 1, "ssoadm Command Line Interface Reference," in *Sun OpenSSO Enterprise 8.0 Administration Reference* for more information.

▼ To Restore the Embedded Configuration Datastore by Loading XML

Use this option if there is only one OpenSSO Enterprise instance and it is corrupted or, multiple servers are configured to share the same configuration datastore and all instances are corrupted. If multiple instances of OpenSSO Enterprise are configured to share the same configuration datastore, repeat steps 1 through 4 on each instance first and then do step 5 and step 6.

1 **Stop all instances of OpenSSO Enterprise.**

2 **Remove all files and directories from the existing configuration directory.**

`$ rm -rf` *configuration_directory*

3 **Restart all instances of OpenSSO Enterprise.**

4 **Reconfigure the OpenSSO Enterprise web application by accessing the OpenSSO Enterprise configurator.**

All configuration attributes must be redefined as they were originally defined. For the configuration of the second and all succeeding OpenSSO Enterprise instances, choose the Add to Existing Deployment option during configuration and point it to the first instance.

5 **Import the saved service configuration data to the configuration datastore using the** `ssoadmin` **command line utility option** `import-svc-cfg`**.**

`./ssoadm import-svc-cfg -u` *username* **`-f`** *password_file_location* **`-e`** *key_to_enctrypt_password* **`-X`** *backup_xml_file*

In the case of the multiple server configuration, this step only needs to be done once.

6 **Restart all OpenSSO Enterprise instances.**

To Restore by Replication of the OpenSSO Configuration Data store

Before You Begin Use this option in the case where multiple OpenSSO Enterprise instances are configured to share the same configuration datastore and at least one of the instances is uncorrupted.

1 **Log in to the console of an uncorrupted instance of OpenSSO Enterprise as administrator.**

2 **Remove the corrupted OpenSSO Enterprise instance(s) from the platform server list.**

The de-provisioning of the OpenSSO configuration datastore node will take effect after all the OpenSSO servers are restarted.

3 **Remove all files and directories from the existing configuration directory for all corrupted instances of OpenSSO Enterprise.**

`$ rm -rf` *configuration_directory*

4 **Restart all instances of OpenSSO Enterprise including those that are corrupted.**

5 **Reconfigure the OpenSSO Enterprise web application on the corrupted OpenSSO Enterprise instance by accessing the OpenSSO Enterprise configurator.**

All configuration attributes must be redefined as they were originally defined.

6 **Import the saved service configuration data to the configuration datastore using the** `ssoadm` **command line utility option** `import-svc-cfg`**.**

`./ssoadm import-svc-cfg -u` *username* **`-f`** *password_file_location* **`-e`** *key_to_enctrypt_password* **`-X`** *backup_xml_file*

In the case of the multiple server configuration, this step only needs to be done once.

7 **Restart all OpenSSO Enterprise instances.**

▼ To Restore the Directory Server Configuration Datastore by Loading XML

Use this option if there is only one OpenSSO Enterprise instance and it is corrupted or, multiple servers are configured to share the same configuration datastore and all instances are corrupted. If multiple instances of OpenSSO Enterprise are configured to share the same configuration datastore, repeat steps 1 through 4 on each instance first and then do step 5 and step 6.

1 **Stop all OpenSSO Enterprise instances.**

2 **Remove all files and directories from the existing configuration directory.**

`$ rm -rf` *configuration_directory*

3 **Confirm that the Directory Server configuration datastore is up and running with no OpenSSO Enterprise service configuration.**

4 **Reconfigure the OpenSSO Enterprise web application by accessing the OpenSSO Enterprise configurator.**

All configuration attributes must be redefined as they were originally defined. For the configuration of the second and all succeeding OpenSSO Enterprise instances, choose the Add to Existing Deployment option during configuration and point it to the first instance.

5 **(Optional) Repeat these steps on each instance of OpenSSO Enterprise that is configured to share the same Directory Server configuration datastore.**

6 **Import the saved service configuration data to the configuration datastore using the `ssoadmin` command line utility option `import-svc-cfg`.**

`./ssoadm import-svc-cfg —u` *username* **`-f`** *password_file_location* **`—e`** *key_to_enctrypt_password* **`-X`** *backup_xml_file*

In the case of the multi-server configuration, this step only needs to be done once.

7 **Restart all OpenSSO Enterprise instances.**

▼ To Restore by Replication of the Directory Server Configuration Datastore

Before You Begin Use this option in the case where multiple OpenSSO Enterprise instances are configured to share the same configuration datastore and at least one of the instances is uncorrupted.

1. **Log in to the console of an uncorrupted instance of OpenSSO Enterprise as administrator.**

2. **Remove the corrupted OpenSSO Enterprise instance(s) from the platform server list.**

   The de-provisioning of the OpenSSO configuration datastore node will take effect after all the OpenSSO servers are restarted.

3. **Remove all files and directories from the existing configuration directory for all corrupted instances of OpenSSO Enterprise.**

   `$ rm -rf` *configuration_directory*

4. **Restart all of the OpenSSO Enterprise servers including those that are corrupted.**

5. **Reconfigure the OpenSSO Enterprise web application by accessing the OpenSSO Enterprise configurator.**

   All configuration attributes must be redefined as they were originally defined.

6. **Import the saved service configuration data to the configuration datastore using the `ssoadm` command line utility option `import-svc-cfg`.**

   `./ssoadm import-svc-cfg -u` *username* **`-f`** *password_file_location* **`-e`** *key_to_enctrypt_password* **`-X`** *backup_xml_file*

   In the case of the multi-server configuration, this step only needs to be done once.

7. **Restart all OpenSSO Enterprise instances.**

APPENDIX A

Changing the Host Name of an OpenSSO Instance

After `opensso.war` is deployed in a web container, the installed OpenSSO instance is uniquely identified by a URL defined with a protocol (http/https), a host name, a port and a deployment URI; for example, `http://ipg-test2.sun.com:8080/opensso`. This URL is defined in the OpenSSO `bootstrap` file as well as in various places in the service configuration data store. When the web container on which OpenSSO is deployed is restarted, OpenSSO uses the `bootstrap` URL to locate its system properties in the service configuration data store and start itself. Additionally, almost all federation and web services endpoints contain this URL. Thus, to change the host name on which the instance of OpenSSO has been installed, use the following procedure.

Note – For this procedure, assume the current OpenSSO URL is `http://current.example.com:58080/opensso`, and the new OpenSSO URL will be `http://new.example1.com:8080/opensso1`.

1. Login to the OpenSSO console as administrator; by default, `amadmin`.
2. Click the Access Control tab.
3. Click `/ Top Level Realm`.
4. Add the new host name as a value for the Realm/DNS Aliases attribute.

   For example, `new.example1.com`.
5. Export the service configuration data to a file named `export.xml`.

   See Chapter 15, "Backing Up and Restoring Configuration Data," for information.
6. Copy `export.xml` to `new.xml`.
7. Open `new.xml` and make the following changes.
   a. Search for

      ```
      <SubConfiguration name=
      "http://current.example.com:58080/opensso" id="server">
      ```

and:

- Change

  ```
  <Value>com.iplanet.am.services.deploymentDescriptor=/opensso</Value>
  ```

  to

  ```
  <Value>com.iplanet.am.services.deploymentDescriptor=/opensso1<Value>
  ```
- Change

  ```
  <Value>com.iplanet.am.server.port=58080</Value>
  ```

  to

  ```
  <Value>com.iplanet.am.server.port=8080</Value>
  ```
- Change

  ```
  <Value>com.iplanet.am.server.host=current.example.com</Value>
  ```

  to

  ```
  <Value>com.iplanet.am.server.host=new.example1.com</Value>
  ```

b. Search for

   ```
   <Service name= "iPlanetAMAuthConfiguration" version="1.0">
   <Schema i18nFileName="amAuthConfig"
   i18nKey="iplanet-am-auth-config-service-description"
   propertiesViewBeanURL="opensso/auth/ACServiceInstanceList">
   ```

   and change `opensso` to `opensso1`.

c. Search for

   ```
   <SubSchema inheritance= "multiple"
   maintainPriority="no" name="NamedConfiguration"
   supportsApplicableOrganization="no" validate="yes">
   <AttributeSchema cosQualifier="default" i18nKey="a101"
   isSearchable="no" name="iplanet-am-auth-configuration"
   propertiesViewBeanURL="opensso/auth/ACModuleList">
   ```

   and change `opensso` to `opensso1`.

d. Search for

   ```
   <AttributeSchema cosQualifier= "default"
   i18nKey="a133" isSearchable="no"
   name="iplanet-am-auth-login-success-url" syntax="string"
   type="list"><DefaultValues><Value>/opensso/console</Value>
   ```

   and change `opensso` to `opensso1`.

e. Search for

```
<AttributeValuePair>
<Attribute name= "sunOrganizationAliases"/>
<Value>opensso</Value>
```

and change `opensso` to `opensso1`.

f. Search for

```
<AttributeSchema cosQualifier= "default"
i18nKey="a103" isSearchable="no"
name="iplanet-am-platform-cookie-domains" syntax="string"
type="list"><DefaultValues><Value>.example.com</Value>
```

and change the cookie domain from `.example.com` to `.example1.com`.

g. Substitute the following strings:

- `http://new.example1.com:8080/opensso1` for `http://current.example.com:58080/opensso`
- `new.example1.com:8080` for `current.example.com:58080`

8. Save `new.xml`.
9. Backup the OpenSSO configuration data.

   This backup can be used to restore a previous configuration. If the OpenSSO configuration data store is the default embedded OpenDS, backup the contents of *OpenSSO-ConfigDir*. *OpenSSO-ConfigDir* represents the name of the directory specified as the configuration directory during initial configuration of OpenSSO. By default, an opensso directory would be created in the home directory of the user configuring the instance. Thus, if root is configuring the instance, *OpenSSO-ConfigDir* is `/opensso`. If any other directory server is used, work with the administrator to back up the OpenSSO configuration data before proceeding.
10. Import `new.xml` back into OpenSSO.

    See Chapter 15, "Backing Up and Restoring Configuration Data," for information.
11. Stop the web container.
12. Replace `http%3A%2F%2Fcurrent.example.com%3A58080%2Fopensso` with `http%3A%2F%2Fnew.example1.com%3A8080%2Fopensso1` in the *OpenSSO-ConfigDir*`/bootstrap` file.

    During OpenSSO deployment, a setup servlet creates a file named `bootstrap` in the OpenSSO configuration directory. This file contains the information that points to a location from which OpenSSO can retrieve configuration data to bootstrap itself.
13. Change the deploy context on the OpenSSO web container to `opensso1`.

    Check the your web container's documentation for instructions.
14. Move *OpenSSO-ConfigDir*`/opensso` to *OpenSSO-ConfigDir*`/opensso1`.

    Be sure to backup this directory first.

15. Change to the *user-home*/`.openssocfg` directory.

    A file named with the prefix `AMConfig` is in this directory; for example, `AMConfig_usr_local_tomcat_webapps_opensso` or `AMConfig_opt_jboss-4.2.2.GA_server_fam2_._deploy_opensso.war_`. *user-home* is the home directory of the user who configured the instance of OpenSSO.
16. Change `opensso` in the `AMConfig*` file to `opensso1`.
17. Start the web container.
18. Log in to OpenSSO as the administrator using the new URL (and host name); by default, `amadmin`.
19. Click the Access Control tab.
20. Click `/ Top Level Realm`.
21. Remove `current.example.com`, the old host name, from the Realm/DNS Aliases attribute.

This second procedure documents how to restore the previous host name. It is based on the examples and information used in the previous procedure.

1. Edit *OpenSSO-ConfigDir*/`bootstrap` by changing the new encoded URL back to the old encoded URL.
2. Import `export.xml` back into OpenSSO.
3. Change the deploy context on the OpenSSO web container back to `opensso`.
4. Move *OpenSSO-ConfigDir*/`opensso1` to *OpenSSO-ConfigDir*/`opensso`.
5. Change `opensso1` in the `AMConfig*` file (located in the *user-home*/`.openssocfg` directory) back to `opensso`.
6. Restart the web container.

Index

A

access control, 203-206
account lockout
  memory, 87-89
  physical, 87-89
Administration service, 26-28
administrator interface, 21
advice message, 105
affiliate entity, 146-152
agent profiles, and realm, 33
agents, 203-206
`amadmin`, 131-132
`AMAgent.properties`, 203-206
`amldapuser`, 132
`arg` login URL parameter, 84
attribute federation, *See* auto-federation
Attribute Mapper, 231
attributes
  Authentication Web Service, 228-229
  Discovery Service, 234-237
  Liberty Personal Profile Service, 229-233
  non-default federation, 199
  SOAP Binding Service, 244-247
audience for this guide, 13
authentication
  account lockout
    memory, 87-89
    physical, 87-89
  and realm, 26
  authentication types, 64-80
  FQDN mapping, 89-90
  login URLs, 36

authentication, login URLs *(Continued)*
    authentication level, 73
    realm, 65-66
    service, 68
    user, 70-71
  methods
    resource, 123-125
  multiple LDAP configurations, 93-95
  persistent cookies, 90-91
  realm
    redirection URLs, 66-67
  realm values, 61-62
  redirection URLs
    authentication level-based, 74-75
    service, 68-69
    user, 71-72
  session upgrade, 91-92
  types
    authentication level, 72-75
    module, 75-77
    realm, 64-67
    role, 77-80
    service, 67-69
    user, 69-72
  user interface
    login URL, 80-87
authentication chains, 55-64
  create, 62-64
authentication level authentication login URLs, 73
authentication level authentication type, 72-75
  configuring, 73
authentication level-based redirection URLs, 74-75

authentication modules, 55-64
  configure, 60-62
  Core, 43-55
  global values, 61
authentication properties, Core, 44-45
authentication types, 64-80
  authentication level, 72-75
    configuring, 73
  module, 75-77
    configuring, 75
  realm, 64-67
    configuring, 65
  role, 77-80
  role-based
    configuring, 77-78
  service, 67-69
    configuring, 67-68
  user, 69-72
    configuring, 70
Authentication Web Service, attribute, 228-229
`authlevel` login URL parameter, 84
Authorizer, 230
auto-creation, 198-199
auto-federation, 197-198
  ID-FF, 172-174
  SAMLv2, 195-196

B

backup, configuration data, 303-304
basic authentication, 200
bootstrapping discovery service, 237
bootstrapping Discovery Service, 242-244
bulk federation, 167-168

C

change host name, 309-312
circle of trust, 152-155
  add providers, 154-155
  create, 146-155
  delete, 155
  modify, 153-154
conditions, 103
  advice messages, 105
  Authentication by Module Chain, 104
  Authentication by Module Instance, 104
  Authentication Level, 104
  IP Address/DNS Name, 104
Conditions
  LDAP Filter, 105
conditions
  session property, 104
Conditions
  Time, 105
configuration data, backup and restore, 303-304
console, 21-22
  administrator interface, 21
  legacy support, 22
  user interface, 22
    login URL, 80-87
containers, 231
Containers, 268-269
  Creating, 268
  Deleting, 268-269
Core authentication module, 43-55
create entities, with `ssoadm`, 155-161
create policy, 99-105
Current Sessions
  Interface, 285-286
  Session Management
    Terminating a Session, 286
  Session Management Window, 285

D

data stores, and realm, 28-30
debug files, 301
Directory Management, 265
Discovery Service, 26-28
  attributes, 234-237
discovery service, bootstrapping, 237
Discovery Service
  bootstrapping, 242-244
  resource offerings, 237-244
documentation
  collections, 15-16

documentation *(Continued)*
- OpenSSO Enterprise, 14-15
- related product, 15-16

`domain` login URL parameter, 87
dynamic identity provider proxying, 175-179

E

enable auto-creation, 198-199
entities
- create, 146-155
- creating with `ssoadm`, 155-161

entity
- affiliate, 146-152
- provider, 146-152

F

federation
- auto-federation, 172-174
- bulk federation, 167-168
- configure global logout, 171
- configure pre-login, 170
- dynamic identity provider proxying, 175-179
- entities
  - creating with `ssoadm`, 155-161
- entities and circles of trust, 146-155
- identity provider metadata sample, 159
- metadata, 146-155
- non-default attributes, 199
- pre-login URL, 168-171
- service provider metadata sample, 157-158

Federation Operations, Finding an Identity Provider for Authentication, 163-167
`forceAuth` login URL parameter, 85
FQDN mapping, and authentication, 89-90

G

global logout, configure, 171
Globalization Settings, 26-28
`goto` login URL parameter, 81
`gotoOnFail` login URL parameter, 81-82
group, subjects, 130
Group Containers, 269
- Creating, 269
- Deleting, 269

Groups, 270-273
- Adding to a Policy, 273
- Create a Managed Group, 271
- Membership by Filter, 270
- Membership by Subscription, 270

H

host name, change, 309-312

I

ID-FF, auto-federation, 172-174
ID-FF writer service URL, 165
ID—FF Identity Provider Introduction service, configuring, 164-165
identities, 127
Identity Management, 265-284
- Containers, 268-269
  - Creating, 268
  - Deleting, 268-269
- Group Containers, 269
  - Creating, 269
  - Deleting, 269
- Groups, 270-273
  - Adding to a Policy, 273
  - Create a Managed Group, 271
  - Membership by Filter, 270
  - Membership by Subscription, 270
- Organizations, 265-268
  - Adding to a Policy, 268
  - Creating, 266-267
  - Deleting, 267-268
- People Containers, 273-274
  - Creating, 273
  - Deleting, 273-274
- Roles, 277-284
  - Adding to a Policy, 284

Identity Management, Roles *(Continued)*
Adding Users to, 280-281
Creating, 278-280
Removing Users from, 283
Users, 274-277
Adding to a Policy, 277
Adding to Services, Roles and Groups, 277
Creating, 274
identity provider, metadata sample, 159
IDP Discovery Server, SAMLv2, 211-212
`idpMNIPOST.jsp`, 190
`idpMNIRedirect.jsp`, 190
`idpMNIRequestInit.jsp`, 190-191
`IDToken`*N* login URL parameter, 85-86
interfaces
Authorizer, 230
ResourceIDMapper, 230
`iPSPCookie` login URL parameter, 86

J
JSP
`idpMNIPOST.jsp`, 190
`idpMNIRedirect.jsp`, 190
`idpMNIRequestInit.jsp`, 190-191
`spMNIPOST.jsp`, 190
`spMNIRedirect.jsp`, 190
`spMNIRequestInit.jsp`, 191

L
LDAP authentication, multiple configurations, 93-95
LDAPv3–compliant directory, 199
legacy support, 22
Liberty Personal Profile Service, attributes, 229-233
`libIDPDiscoveryConfig.properties`, 166-167
load balancing, 201-203
`locale` login URL parameter, 83
login URL parameters, 80-87
login URLs
and authentication, 36
authentication level, 73
realm authentication, 65-66
login URLs *(Continued)*
service, 68
user, 70-71

M
Managing OpenSSO Enterprise Objects, 265-284
memory account lockout, 87-89
metadata
federation, 146-155
identity provider sample, 159
managing with `ssoadm`, 155-159
service provider sample, 157-158
methods
authentication
resource, 123-125
module authentication type, 75-77
configuring, 75
`module` login URL parameter, 83-84

N
name identifiers, 195-198
naming service, and policy, 98
non-default federation attributes, 199

O
`org` login URL parameter, 87
Organizations, 265-268
Adding to a Policy, 268
Creating, 266-267
Deleting, 267-268
overview
authentication
login URL, 80-87
auto-creation, 198-199
auto-federation, 197-198
bulk federation, 167-168
dynamic identity provider proxying, 175-179
policy agents, 97-99
policy process, 97-99

overview *(Continued)*
  pre-login URL, 168-171

P
parameters, pre-login URL, 169-170
password, change, 129
Password Reset, 26-28
People Containers, 273-274
  Creating, 273
  Deleting, 273-274
`PersistAMCookie` login URL parameter, 86
persistent cookies, and authentication, 90-91
persistent name identifier, 195-198
physical account lockout, 87-89
policies
  conditions, 103
    advice messages, 105
policy, 97-125
  and naming service, 98
  and realm, 32
  create, 107-114
    console, 111-112
    referral, 112-114
    `ssoadm`, 107-111
  creating, 99-105
  modify, 114-117
  policy types, 99-106
  referral, 106
    modify, 114-117, 117-119
    referrals, 106
    rules, 106
  resource authentication, 123-125
  rules, 99
  subjects, 101
policy agents
  overview, 97-99
  process, 97-99
Policy Configuration service, 26-28
policy types, 99-106
pre-login, configure, 170
pre-login URL, 168-171
  configure, 170
  parameters, 169-170
prerequisites for this guide, 13
privileges
  and realm, 30
  upgrade, 32
procedures
  store resource offerings, 237-240, 240-242, 242-244
provider entity, 146-152

Q
query parameter, 169

R
reader service URL, 165
realm authentication login URLs, 65-66
realm authentication redirection URLs, 66-67
realm authentication type, 64-67
  configuring, 65
`realm` login URL parameter, 82
realms, 23-33
  adding service, 28
  agent profiles, 33
  authentication, 26
  creating, 24-25
  data stores, 28-30
  general properties, 24-25
  modifying, 24-25
  modifying service, 28
  policy, 32
  privileges, 30
    upgrade, 32
  services, 26-28
  subject, 32-33
  subjects, 127
    group, 130
    user, 127
redirection URLs
  authentication level-based, 74-75
  realm, 66-67
  service, 68-69
  user, 71-72
referral, create, 112-114

referral policy, 106
referrals, create, 107-114
related guides, 14-16
request handler, 245
resource authentication, 123-125
resource offering, for bootstrapping, 242-244
resource offerings
  as dynamic attributes, 240-242
  as user attributes, 237-240
  storing, 237-244
resource offerings for bootstrapping, 237
ResourceID Mapper, 230
restore, configuration data, 303-304
role authentication type, 77-80
role-based authentication type, configuring, 77-78
`role` login URL parameter, 86
Roles, 277-284
  Adding to a Policy, 284
  Adding Users to, 280-281
  Creating, 278-280
  Removing Users from, 283
rules, 99

S
SAML, 249-261
  Attributes, 249-257
  site identifiers
    configure, 251
  target URL, 255-256
  trusted partner
    configure step 1, 252
    configure step 2, 252-255
SAML v2 Plug-in for Federation Services, and `AMAgent.properties`, 203-206
SAMLv2
  auto-federation, 195-196
  IDP Discovery Service, 211-212
SAMLv2 IDP Discovery service
  configuring
    URLs, 164
SAMLv2 reader service URL, 164
SAMLv2 writer service URL, 164
Secure Socket Layer/Transport Layer Security, *See* SSL/TLS
security
  SOAP binding, 200-201
  XML encryption, 200
  XML signing, 200
service authentication login URLs, 68
service authentication redirection URLs, 68-69
service authentication type, 67-69
  configuring, 67-68
`service` login URL parameter, 84
service provider, metadata sample, 157-158
services
  adding to realm, 28
  and realm, 26-28
  Discovery Service, 26-28
  Globalization Settings, 26-28
  modifying properties, 28
  Password Reset, 26-28
  Policy Configuration, 26-28
  Session, 26-28
  User, 26-28
Session service, 26-28
session upgrade, and authentication, 91-92
single sign-on, *See* SSO
single sign-on with transient name identifier, 195
site identifiers, 251
SOAP binding, 200-201
  basic authentication, 200
  SSL/TLS, 200-201
  SSL/TLS client authentication, 201
  SSL/TLS server authentication, 201
SOAP Binding Service
  attributes, 244-247
  request handler, 245
special users
  `amadmin`, 131-132
  `amldapuser`, 132
  `UrlAccessAgent`, 132-133
`spMNIPOST.jsp`, 190
`spMNIRedirect.jsp`, 190
`spMNIRequestInit.jsp`, 191
SSL/TLS, 200-201
  client authentication, 201

SSL/TLS *(Continued)*
  server authentication, 201
SSO, use cases, 195-198
SSO without service provider user account, 196-197
ssoadm, *See* do-bulk-fed-data
`ssoadm`
  and metadata, 155-159
  create entities, 155-161
subject, and realm, 32-33
subjects, 101, 127
  group, 130
  user, 127

T
target URLs, 255-256
Terminating a Session, 286
transient name identifier, 195-198
trusted partners, 252

U
`UrlAccessAgent`, 132-133
use cases
  access control, 203-206
  agents, 203-206
  basic authentication, 200
  enable auto-creation, 198-199
  load balancing, 202-203
  single sign-on with transient name identifier, 195
  single sign-on without service provider user account, 196-197
  SSL/TLS, 200-201
  using non-default federation attributes, 199
user, subjects, 127
user authentication login URLs, 70-71
user authentication redirection URLs, 71-72
user authentication type, 69-72
  configuring, 70
user interface, 22
user interface login URL, 80-87
`user` login URL parameter, 82
User service, 26-28
Users, 274-277
  Adding to a Policy, 277
  Adding to Services, Roles, and Groups, 277
  Creating, 274

X
XML encryption, 200
XML signing, 200