

CA Identity Manager

Configuration Guide

r12



Second Edition

This documentation and any related computer software help programs (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of the Documentation for their own internal use, and may make one copy of the related software as reasonably required for back-up and disaster recovery purposes, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the provisions of the license for the Product are permitted to have access to such copies.

The right to print copies of the Documentation and to make a copy of the related software is limited to the period during which the applicable license for the Product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

EXCEPT AS OTHERWISE STATED IN THE APPLICABLE LICENSE AGREEMENT, TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

The use of any product referenced in the Documentation is governed by the end user's applicable license agreement.

The manufacturer of this Documentation is CA.

Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Copyright © 2009 CA. All rights reserved.

CA Product References

This document references the following CA products:

- CA Identity Manager
- CA SiteMinder® Web Access Manager
- CA Security Command Center (SCC)
- CA Audit
- eTrust® Directory, also known as CA Directory

Contact CA

Contact Technical Support

For your convenience, CA provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA products. At <http://ca.com/support>, you can access the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Provide Feedback

If you have comments or questions about CA product documentation, you can send a message to techpubs@ca.com.

If you would like to provide feedback about CA product documentation, complete our short [customer survey](#), which is also available on the CA support website, found at <http://ca.com/support>.

Contents

Chapter 1: Introduction to Identity Manager Environments 13

Identity Manager Environment Components	13
Multiple Identity Manager Environments.....	14
Identity Manager Management Console	15
How to Access the Identity Manager Management Console	16
Provisioning in an Identity Manager Environment	16

Chapter 2: Protecting Identity Manager 19

Management Console Security	19
Use SiteMinder to Secure the Management Console	19
Identity Manager User Console Security	20

Chapter 3: Sample Identity Manager Environment 21

Overview of Sample Identity Manager Environment	21
How to Configure the NeteAuto Sample with Organization Support	21
LDAP Directory Structure for NeteAuto	22
Relational Database for NeteAuto	23
Prerequisite Software for NeteAuto	23
Installation Files for the NeteAuto Environment	24
Install the NeteAuto Environment	24
Configure an LDAP User Directory	25
Configure a Relational Database	26
Create the Identity Manager Directory	27
Create the NeteAuto Identity Manager Environment	29
How to Configure the NeteAuto Sample without Organization Support	31
Sample Identity Manager Environment Description	31
Installation Files for the Neteauto Environment	32
Install the NeteAuto Environment—No Organization Support	33
Prerequisite Software	34
Configure a Relational Database	34
Create the Identity Manager Directory	35
Create the NeteAuto Identity Manager Environment	36
How to Use the NeteAuto Identity Manager Environment	38
Self-Service Task Management	38
User Management	42
How to Configure Additional Features	46

Chapter 4: Creating an Identity Manager Directory for an LDAP User Store 47

Identity Manager Directories	47
How to Create an Identity Manager Directory	48
Directory Structure	48
Directory Configuration File	50
How to Select a Directory Configuration Template	51
How to Describe a User Directory to Identity Manager	53
How to Modify the Directory Configuration File	53
Connection to the User Directory	54
Provider Element	55
Directory Search Parameters	58
User, Group and Organization Managed Object Descriptions	59
Managed Object Descriptions	60
Attribute Descriptions	63
Managing Sensitive Attributes	68
eTrust Directory Considerations	73
Microsoft Active Directory Considerations	74
IBM Directory Server Considerations	74
Oracle Internet Directory Considerations	75
Well-Known Attributes for an LDAP User Store	75
User Well-Known Attributes	76
Group Well-Known Attributes	79
Organization Well-Known Attributes	80
%Admin_Role_Constraint% Attribute	81
Configure Well-Known Attributes	81
Describe the User Directory Structure	82
How to Describe a Hierarchical Directory Structure	82
How to Describe a Flat User Directory Structure	82
How to Describe a Flat Directory Structure	82
How to Describe a User Directory that Does Not Support Organizations	83
How to Configure Groups	83
Configure Self-Subscribing Groups	83
Configure Dynamic and Nested Groups	84
Add Support for Groups as Administrators of Groups	85
Validation Rules	86
Additional Identity Manager Directory Properties	86
Configure Sort Order	87
Search across Objectclasses	88
Specify Replication Wait Time	89
Specify LDAP Connection Settings	90
How to Improve Directory Search Performance	91

Chapter 5: Creating an Identity Manager Directory for a Relational Database **93**

Identity Manager Directories	93
Important Notes When Configuring Identity Manager for Relational Databases	95
Create an Oracle Datasource for WebSphere	96
How to Create an Identity Manager Directory	97
How To Create a JDBC Datasource	97
Create a JDBC Datasource for JBoss Application Servers	98
WebLogic Data Sources	99
WebSphere Datasources	103
How to Create an ODBC Data Source For Use with SiteMinder	105
How to Describe a Database in a Directory Configuration File	105
Modify the Directory Configuration File	107
Managed Object Descriptions	108
Connection to the User Directory	119
Description of a Database Connection	120
SQL Query Schemes	123
Well-Known Attributes for a Relational Database	125
User Well-Known Attributes	125
Group Well-Known Attributes	127
%Admin_Role_Constraint% Attribute	128
Configure Well-Known Attributes	129
How to Configure Self-Subscribing Groups	130
Validation Rules	131
Organization Management	131
How to Set Up Organization Support	131
Configure Organization Support in the Database	132
Root Organization Specification	132
Well-Known Attributes for Organizations	133
How to Define the Organizational Hierarchy	134
How to Improve Directory Search Performance	134

Chapter 6: Managing Identity Manager Directories **135**

Prerequisites to Creating an Identity Manager Directory	135
Create an Identity Manager Directory in the Management Console	136
View an Identity Manager Directory	138
Identity Manager Directory Properties	138
Identity Manager Directory Properties Window	139
How to View Managed Objects Properties and Attributes	140
Validation Rule Sets	144
How to Update Settings for an Identity Manager Directory	145

Export an Identity Manager Directory	146
Update an Identity Manager Directory	146
Delete an Identity Manager Directory	147

Chapter 7: Managing Identity Manager Environments 149

Identity Manager Environments	149
Prerequisites to Creating an Identity Manager Environment	150
Create an Identity Manager Environment	151
How to Access an Identity Manager Environment	155
Modify Identity Manager Environment Properties	156
Export an Identity Manager Environment	158
Import an Identity Manager Environment	159
Restart an Identity Manager Environment	159
Delete an Identity Manager Environment	160
Optimize Policy Rule Evaluation	161
Role and Task Settings	162
Export Role and Task Settings	162
Import Role and Task Settings	162
Modify the System Manager Account	163
Access the Status of an Identity Manager Environment	165
Troubleshooting Identity Manager Environments	165

Chapter 8: Advanced Settings 169

Auditing Logs	169
Business Logic Task Handlers	170
Email Notifications	170
Event Listeners	171
Identity Policies	171
Logical Attributes Handlers	172
Miscellaneous Properties	172
Notification Rules	173
Organization Selectors	173
Provisioning	174
Provisioning Directory	175
Enable Session Pooling	175
Enable Password Synchronization	176
Attribute Mappings	176
Inbound Mappings	176
Outbound Mappings	176
User Console	177
Web Services	178

Workflow Properties	179
Global Process to Event Mapping	180
Work Item Delegation.....	181
Workflow Participant Resolvers	181
Import/Export Custom Settings.....	182

Chapter 9: Customizing the User Console 183

Skins	183
Components of a Skin	184
Identity Manager Skins.....	185
Skin Associations	188
How to Create a Skin.....	188
Recompile the .jsp Files and Restart the Application Server	192
How to Use Multiple Skins	192
How to Edit Console Pages	194
Logout Pages.....	197
Configuring Logout Pages	198
Custom Help	199
How Customized Help Works	199
How Help Determines Which Link To Use	201
How to Customize the Help	202
Examples of How to Use Custom Help	202

Chapter 10: Validation Rules 205

Introduction	205
About Validation Rules	205
Types of Validation Rules	206
Validation Rule Sets.....	207
Basics of Validation Rule Definition	208
Using Default Validation Rules	209
Default Data Validations	209
Predefined Validation Rules	212
How to Implement Custom Validation Rules	212
Regular Expression Implementation	212
JavaScript Implementation	213
Java Implementation.....	216
Exceptions	219
How to Configure Validation Rules	221
How to Configure Task-Level Validation	222
How to Configure Directory-Level Validation	223
How to Initiate Validation	228

Sample Implementations	228

Chapter 11: Auditing an Identity Manager Environment 229

Audit Data	229
How to Configure Auditing	229
Configure Audit Settings	230
Audit Settings File	231
How to Enable Auditing For a Task	242
How to View Audit Data in eTrust Security Command Center	243
Clean Up the Audit Database	243

Chapter 12: Identity Manager Localization 245

Introduction	245
How to Configure the System for Localization	246
Prerequisites for Localization Support	246
Configure the PATH Variable	247
Configure Perl Tools for Localization	247
How to Plan the Localization	247
Elements that Can Be Translated	248
Resource Bundles	248
How to Localize Resource Bundles	250
Tokenize the RoleDefinitions.xml File	251
Translate the RoleDefinitions.xml file	252
Translate Resource Bundles	253
Install the Resource Bundles	255
Install the Translated RoleDefintions.xml Files	256
Translate Email Templates	257
Configure Multibyte Character Support (SQL Server)	258
How to Localize Branding Skins	258
Modify the Change My Account Task	259
Localizing Custom Code	259
How to Display Identity Manager in a Different Language	260
Sample Localized Identity Manager Environments	261
Install the Sample Resource Bundles	262
Import a Sample RoleDefinitions.xml File	263
Install the Online Help	264
Set the imlanguage HTTP Header in a SiteMinder Response	264

Chapter 13: Identity Manager in a Production Environment 267

To migrate Admin roles and task definitions	267

To export Admin role and task definitions.....	268
To import Admin role and task definitions	268
To verify the role and task import	269
To migrate Identity Manager skins	269
Update Identity Manager in a Production Environment	269
To migrate an Identity Manager environment	270
To export an Identity Manager environment	271
To import an Identity Manager environment	271
To verify the Identity Manager environment migration.....	272
Migrate the IdentityMinder EAR for JBoss.....	272
Migrate the IdentityMinder EAR for WebLogic	273
Migrate the IdentityMinder EAR for WebSphere	273

Chapter 14: Identity Manager Logs 277

How to Track Problems in Identity Manager	277
How to Trace Components and Data Fields	279

Chapter 15: Integrating SiteMinder with Identity Manager 283

SiteMinder Integration	283
SiteMinder Components	285
SiteMinder Configuration	286
Collect User Credentials Using a Custom Authentication Schemes	287
Configure the LogOff URI	288
Aliases in SiteMinder Realms	290
Update an Alias in SiteMinder Realms	290
Modify a SiteMinder Password or Shared Secret	292
How to Configure Identity Manager Agent Settings.....	293
Configure an Identity Manager Environment to Use Different Directories for Authentication and Authorization	294
How to Improve the Performance of LDAP Directory Operations.....	296

Appendix A: Global Database Task Persistence 297

Modify the Task Persistence EJB Descriptor for WebLogic	297
Modify the Task Persistence EJB Descriptor for JBoss.....	298
Modify the Task Persistence EJB Descriptor for WebSphere	298

Index 301

Chapter 1: Introduction to Identity Manager Environments

This section contains the following topics:

[Identity Manager Environment Components](#) (see page 13)

[Multiple Identity Manager Environments](#) (see page 14)

[Identity Manager Management Console](#) (see page 15)

[How to Access the Identity Manager Management Console](#) (see page 16)

[Provisioning in an Identity Manager Environment](#) (see page 16)

Identity Manager Environment Components

An *Identity Manager Environment* is a view of a management namespace that lets Identity Manager administrators manage objects such as users, groups, and organizations, with a set of associated roles and tasks. The Identity Manager environment controls the management and graphical presentation of a directory.

A single user store may be associated with [multiple Identity Manager environments](#) (see page 14) to define different views of the directory. However, an Identity Manager environment is associated with only one user store.

Identity Manager environments may contain the following elements:

Directory

Describes a user store to Identity Manager. It includes the following:

- A pointer to a user store, which stores managed objects such as users, groups, and organizations.
- Metadata that describes how managed objects are stored in the directory and represented in Identity Manager.

Provisioning Directory (optional)

Stores information required by the Provisioning Server to manage additional accounts in managed endpoints.

Note: For more information about the Provisioning Server or the Provisioning Directory, see the *Provisioning Guide*.

User Console

Enables Identity Manager administrators to perform tasks in an Identity Manager environment.

Task and role definitions

Determine user privileges in Identity Manager and other applications. These task and role definitions are initially available in the Identity Manager environment where they can be assigned to users.

You can customize the default roles and tasks using the User Console.

Self-service

Lets users create and maintain their own accounts for accessing resources, such as a customer web site. It also lets users request a temporary password if they forget their current password.

Workflow definitions

Identity Manager includes default workflow definitions that automate approval and notification tasks associated with user management tasks, such as creating user profiles or assigning users to roles or groups. You can modify the default workflow processes in Identity Manager to support each organization's requirements.

Skins

Determine the appearance of the Identity Manager user interface.

Custom features

You can extend Identity Manager to suit your company's requirements using the Identity Manager APIs. See the *Programming Guide for Java*.

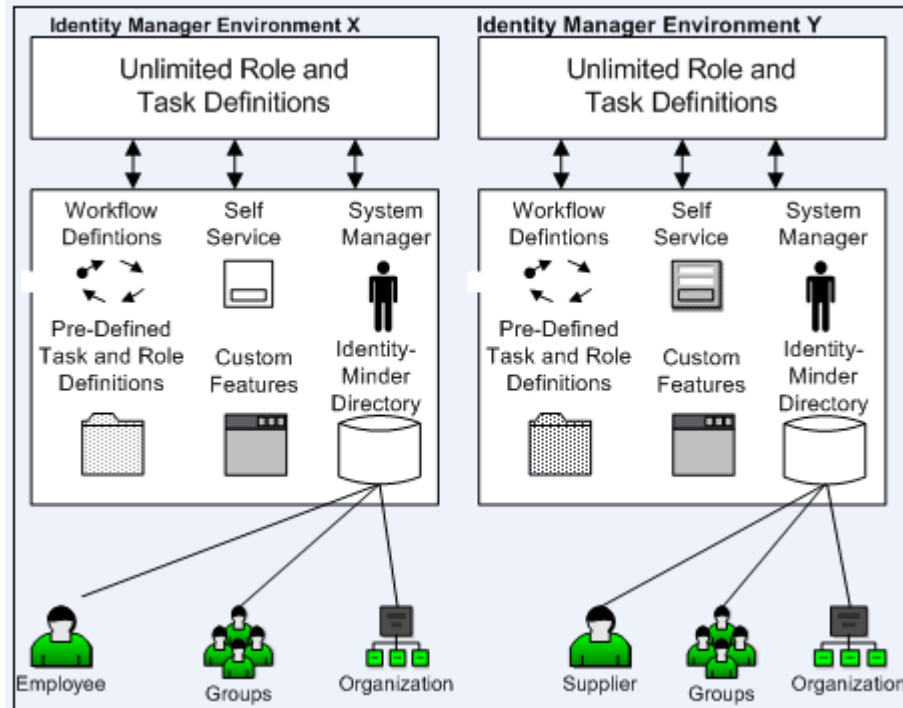
Each Identity Manager environment requires one or more system managers to customize the initial roles and tasks using the User Console. Once a system manager creates the initial roles and tasks, that manager can grant administrative privileges to users in that environment. These users become administrators who manage users, groups, and organizations. See the *Administration Guide*.

Multiple Identity Manager Environments

Create multiple Identity Manager environments when you want to do one of the following:

- **Manage additional user stores**—You can manage users in different types of user stores. For example, your company may store all of its user profiles in a Sun Java System LDAP directory. You enter into a joint venture with a partner that uses an Oracle database to store user information. You would want a different Identity Manager environment for each set of users.

- Manage objects with different LDAP object classes—When Identity Manager is managing an LDAP directory, you can manage objects of the same type in the same directory that have different object classes and attributes. For example, the following illustration shows a directory that contains two types of users:
 - Employees, who have an employee ID number
 - Suppliers, who are identified by supplier number



Identity Manager Management Console

As an Identity Manager system administrator, your responsibilities include the following:

- Creating an Identity Manager directory
- Configuring an Identity Manager environment
- Assigning a system manager
- Enabling custom features for initial use

To configure an Identity Manager environment, use the Management Console, a Web-based application.

The management console is divided into the following two sections:

- Directories—Use this section to create and manage Identity Manager directories, which describe user stores to Identity Manager.
- Environments—Use this section to create and manage Identity Manager environments, which control the management and graphical presentation of a directory.

How to Access the Identity Manager Management Console

To access the Management Console, enter the following URL in a browser:

`http://hostname:port/idmmanage`

hostname

Defines the fully-qualified domain name for the server where Identity Manager is installed

port

Defines the application server port.

The following are examples of paths to the Management Console:

- For WebLogic:
`http://myserver.mycompany.org:7001/idmmanage`
- For JBoss:
`http://myserver.mycompany.org:8080/idmmanage`
- For WebSphere:
`http://myserver.mycompany.org:9080/idmmanage`

Provisioning in an Identity Manager Environment

You can optionally configure provisioning for an Identity Manager environment to provide accounts in other systems, called endpoints, to users managed by Identity Manager. Accounts provide users with access to additional resources, such as an email account. You provide these additional accounts by assigning provisioning roles, which you create through Identity Manager. Provisioning roles are associated with account templates that define accounts that users can receive.

When you assign a provisioning role to a user, that user receives the accounts defined by the account templates in the role. The account templates also define how user attributes are mapped to accounts. The accounts exist in managed endpoints defined by the policies.

Note: For full details on using a provisioning in Identity Manager, see the *Provisioning Guide*.

Chapter 2: Protecting Identity Manager

This section contains the following topics:

[Management Console Security](#) (see page 19)

[Identity Manager User Console Security](#) (see page 20)

Management Console Security

After installing Identity Manager, you can limit access to the Management Console if Identity Manager integrates with SiteMinder.

Use SiteMinder to Secure the Management Console

To protect the Management Console initially, you can create a SiteMinder policy.

A SiteMinder policy identifies a resource that you want to protect, such as the Management Console, and grants a set of users access to that resource.

Note: The following procedure refers to the Policy Server User Interface, which is available with SiteMinder r6.0 SP 5 and SiteMinder r12. SiteMinder r12 also includes the Administrative UI, which is a browser-based console that you can use to manage the Policy Server. For information on using the Administrative UI, see the *CA SiteMinder Web Access Manager Policy Server Administration Guide*.

To create a SiteMinder Policy to protect the Management Console

1. Log in to the Policy Server User Interface as an administrator with Domain privileges.
2. In the Domains tab, expand the policy domain for the appropriate Identity Manager environment. The domain name has the following format:

*Identity Manager-environment*Domain

In this format, *Identity Manager-environment* specifies the name of the environment you are modifying. For example, when the name is *employees*, the domain name is *employeesDomain*.

Under the domain, expand Realms.

3. Create a new realm with the following resource filter:

`/idmmanage/`

4. Create a new rule. Specify an asterisk (*) as the filter to protect all pages in the Management Console.
5. Create new a policy and associate it with the rule you created in the previous step.

Be sure to associate users who can access the Management Console with the policy.
6. Restart the application server.

Identity Manager User Console Security

The User Console is the user interface that administrators use to manage objects such as users, groups, and organizations in an Identity Manager environment, with a set of associated roles and tasks. When an administrator logs into the User Console, that administrator can only see tasks that he can perform in that environment.

By default, Identity Manager protects access to the User Console with native authentication. Identity Manager administrators enter a valid username and password to log into an Identity Manager environment. Identity Manager authenticates the name and password against the user store that Identity Manager manages.

If Identity Manager integrates with SiteMinder, Identity Manager *automatically* uses SiteMinder basic authentication to protect the environment. No additional configuration is required to use basic authentication. You can configure advanced authentication methods using the SiteMinder Administrative User Interface.

Note: For more information, see the *CA SiteMinder Web Access Manager Policy Server Configuration Guide*.

Chapter 3: Sample Identity Manager Environment

This section contains the following topics:

- [Overview of Sample Identity Manager Environment](#) (see page 21)
- [How to Configure the NeteAuto Sample with Organization Support](#) (see page 21)
- [How to Configure the NeteAuto Sample without Organization Support](#) (see page 31)
- [How to Use the NeteAuto Identity Manager Environment](#) (see page 38)
- [How to Configure Additional Features](#) (see page 46)

Overview of Sample Identity Manager Environment

Identity Manager includes a sample environment that you can use to learn about and test Identity Manager.

The sample environment is based on a company called NeteAuto that buys and sells cars. NeteAuto administrators use Identity Manager to manage employees, suppliers, and regional dealerships.

There are sample NeteAuto environments for the following user store configurations:

- LDAP user stores that support organizations
- LDAP user stores that do not support organizations
- Relational Database user stores that support organizations
- Relational Database user stores that do not support organizations

How to Configure the NeteAuto Sample with Organization Support

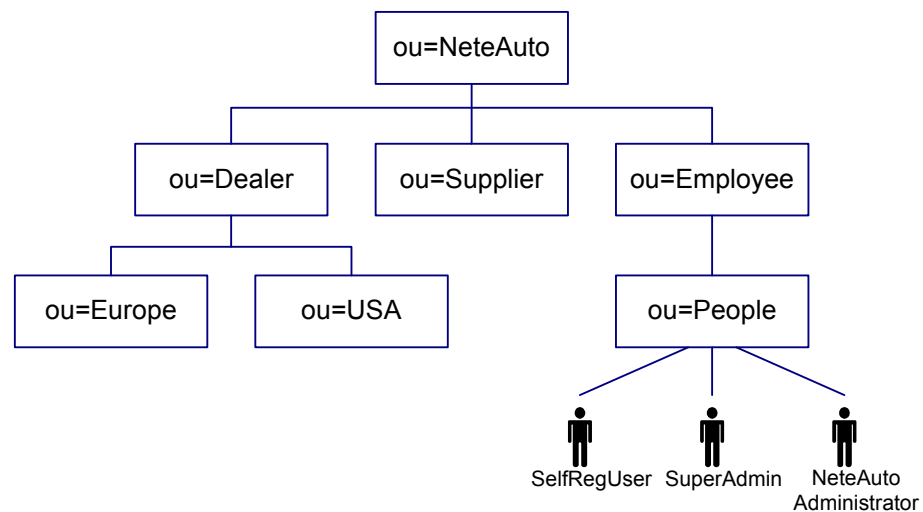
Configuring the NeteAuto sample with organization support involves the following steps:

- Installing the prerequisite software
- Installing the sample Identity Manager environment

- Configuring an LDAP user directory
- Configuring a relational database
- Creating the Identity Manager directory
- Creating the NeteAuto Identity Manager environment

LDAP Directory Structure for NeteAuto

The following illustration describes the NeteAuto sample for LDAP directories:



The sample Identity Manager environment includes the following users:

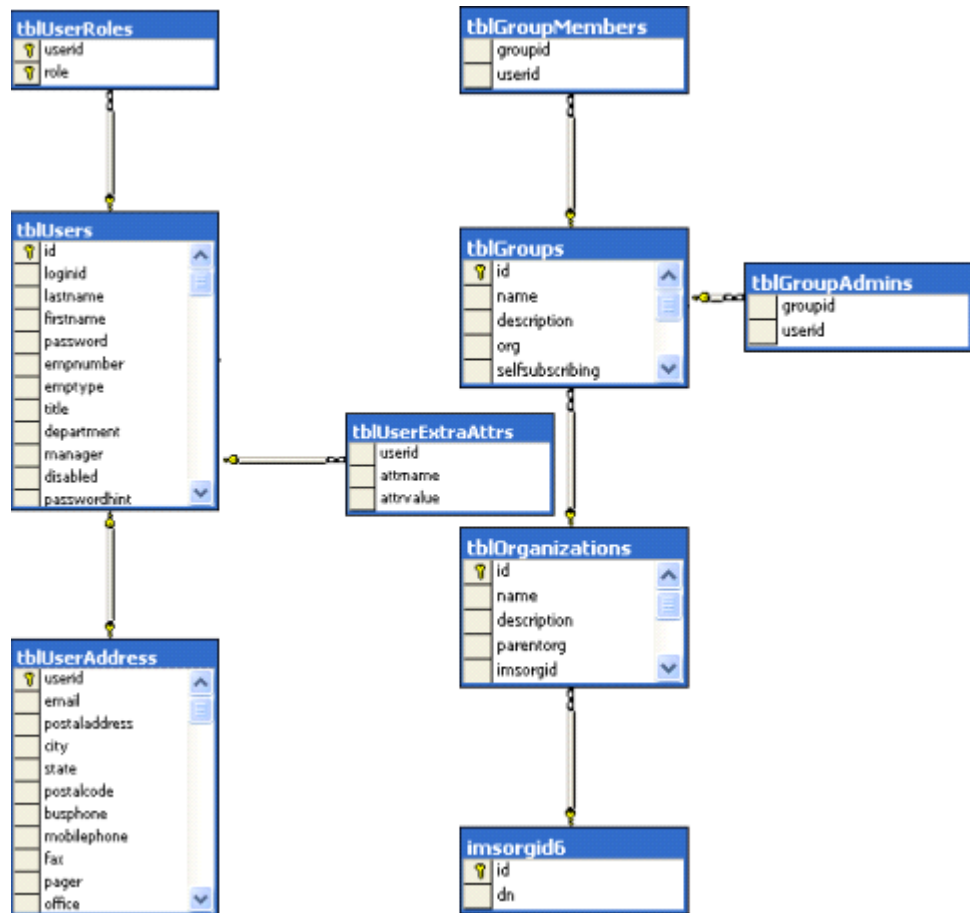
- SuperAdmin is the administrator account with the System Manager role for this Identity Manager environment. As SuperAdmin, you can perform all default admin tasks.

Note: For a description of the default admin tasks, see the *Administration Guide*.

- SelfRegUser is the administrator account that Identity Manager uses to enable self-registration for this Identity Manager environment.
- NeteAuto Administrator does not have any privileges when you install the NeteAuto environment. However, you can assign that user the Group Manager role, as described in Assign the Group Manager Role.

Relational Database for NeteAuto

The following illustration describes the relational database for the NeteAuto sample including an organization table:



Prerequisite Software for NeteAuto

The NeteAuto Identity Manager environment has the following prerequisites:

- Identity Manager must be installed as described in the *Installation Guide*. Be sure to install the Identity Manager Administrative Tools.
- You must have access to a Sun Java system (Sun ONE or iPlanet) Directory Server or a Microsoft SQL Server database.

Installation Files for the NeteAuto Environment

Identity Manager includes a set of files that you can use to set up a sample Identity Manager environment. An Identity Manager environment is a view of a management namespace that enables Identity Manager administrators to manage objects, such as users, groups, and organizations, with a set of associated roles and tasks. An Identity Manager environment controls the management and graphical presentation of a directory.

The sample Identity Manager environment includes the following:

- Sample objects, such as users and organizations
- Role, task, and screen definitions

Tasks appear in the User Console when you point to a vertical list of objects, such as users or groups. The tasks that appear are based on the roles that are assigned to the user who is logged in.

Note: For more information about roles and tasks, see the *Administration Guide*.

- A sample skin that customizes the User Console for NeteAuto users
- A directory configuration file that you use to create an Identity Manager directory

The files for creating the sample Identity Manager environment are installed in the following location:

`im_admin_tools_dir\samples\NeteAuto`

In this path:

`im_admin_tools_dir`

Defines the installed location of the Identity Manager administrative tools, as in the following example:

- **Windows:** C:\Program Files\CA\IAM Suite\Identity Manager\tools\samples\NeteAuto\
- **UNIX:** `home_directory/CA/IAM Suite/Identity Manager/tools/samples/NeteAuto`

Install the NeteAuto Environment

Perform the following process to install the NeteAuto environment.

To install the NeteAuto environment

1. Make sure that the [prerequisite software is installed](#) (see page 23).

2. Configure the user store and import the sample data.
 - For LDAP users: [Configure an LDAP User Directory](#) (see page 25)
 - For Microsoft SQL Server users: Configure a Relational Database
3. Create the NeteAuto Identity Manager directory.
4. Create the NeteAuto Identity Manager environment
5. [Configure the look and feel of the Identity Manager user interface for NeteAuto users](#) (see page 40).

Configure an LDAP User Directory

Depending upon your installation, you may already have an LDAP directory. You can use the following procedure to check whether the directory exists or to create the directory.

To configure an LDAP user directory

1. In the directory server console, create a new instance of LDAP with the following root:

`dc=security,dc=com`

Write down the port number, which you will need later.

2. Import the NeteAuto.ldif file to the directory server:

`im_admin_tools_dir\samples\NeteAuto\NeteAuto.ldif`

In this path:

im_admin_tools_dir

Defines the installed location of the Identity Manager Administrative Tools.

Note: If you experience problems importing the LDIF file or creating the Identity Manager directory, add the following to the beginning of the LDIF file:

```
dn: dc=security, dc=com
objectClass: top
objectClass: domain
dc: security
```

After saving the file, repeat steps 1 and 2.

Configure a Relational Database

Perform the following procedure to configure a relational database.

To configure a relational database

1. Create a database instance named NeteAuto.
2. Create a user named neteautoadmin with the password test. Grant neteautoadmin rights (such as public and db_owner rights) to NeteAuto by editing the properties of the user.

Note: The neteautoadmin must have at least select, insert, update and delete permissions for all the tables created by the .sql script for creating the NeteAuto database, and must be able to execute all of the stored procedures, if any, defined in these scripts.

3. While editing the user's properties, make NeteAuto the default database for neteautoadmin.
4. Run the following scripts in the order in which they are listed:
 - *db_type-rdbuserdirectory.sql*—Configures the tables for the NeteAuto sample, and creates the user entries
 - *ims_db_type_rdb.sql*—Configures support for organizations

db_type

Defines the mssql or oracle depending on the type of database that you are configuring.

These script files are located in the
im_admin_tools_dir\samples\NeteAutoRDB\Organization folder.

5. Define a JDBC data source named neteautoDS that points to the NeteAuto database.

The procedure for configuring a data source depends on the type of application server where Identity Manager is installed. The section [How to Create a JDBC Data Source](#) (see page 97) includes application server-specific instructions for creating a JDBC data source.

Create the Identity Manager Directory

Perform the following procedure to create the Identity Manager directory.

To create the Identity Manager directory

1. Open the Management Console by entering the following URL in a browser:

`http://domain_name:port/idmmanage`

domain_name

Defines the fully qualified domain name of the server where Identity Manager is installed

port

Defines the application server port number.

2. Click Directories.

The Identity Manager directories screen appears.

3. Click New to start the Identity Manager directory wizard.

4. Browse for the appropriate directory configuration .xml file, and click Next.

The directory configuration file is located in the following folders:

- For Sun Java System Directory Server user directories:

`im_admin_tools_dir\samples\NeteAuto\Organization\directory.xml`

- For relational databases:

`im_admin_tools_dir\samples\NeteAutoRDB\Organization\db_type
directory.xml`

`im_admin_tools_dir`

Defines the installed location of the Identity Manager administrative tools.

`db_type`

Specifies the type of database that you are configuring: mssql or oracle.

Status information is displayed in the Directory Configuration Output screen.

5. On the second page of the wizard, supply the following values:

■ Sun Java System Directory Server

Name

NeteAuto Directory

Description

Sample NeteAuto directory

Connection Object Name

NeteAuto Users

Host

Computer name or IP address of the system where the user store is installed

Port

Port number for the user store

Search root

dc=security,dc=com

Username

User name for an account that can access the user store.

Password and Confirm Password

Password for the user account

■ Microsoft SQL Server and Oracle Databases

Name

NeteAutoRDB Directory

Description

Sample NeteAuto directory

Connection Object Name

NeteAutoRDB

ODBC Data Source

neteautoDS

Username

neteautoadmin

Password

test

6. Click Next.
7. Click Finish to exit the wizard.

Create the NeteAuto Identity Manager Environment

Perform the following procedure to create the NeteAuto Identity Manager environment.

To create the NeteAuto Identity Manager environment

1. In the Management Console, click Environments.
2. In the Identity Manager environments screen, click New.
The Identity Manager environment wizard appears.
3. In the first page of the wizard, enter the following values:

Environment name

NeteAuto Environment

Description

This is a sample environment

Alias

neteauto

The alias is added to the URL for accessing the Identity Manager environment. For example, the URL for accessing the neteauto environment is the following:

`http://domain/idm/neteauto`

domain

Defines the fully qualified domain name of the server where Identity Manager is installed, for example:

`http://myserver.mycompany.org/idm/neteauto`

Note: The alias is case-sensitive.

Click Next.

4. Select the Identity Manager directory to associate with the environment that you are creating:
 - For Sun Java System Directory Server, use the NeteAuto Directory
 - For Microsoft SQL Server or Oracle database, use the NeteAutoRDB Directory

Click Next.

5. Configure support for public tasks, such as the self-registration and forgotten password tasks, as follows:
 - a. Type the following alias for public tasks:
`neteautopublic`
 - b. Enter SelfRegUser as the anonymous user account.
 - c. Click Validate to view the user's unique identifier.

Note: Users do not need to log in to use public tasks.
6. Select the tasks and roles to create for the NeteAuto environment:
 - a. Select Import roles from the file.
 - b. Browse to one of the following locations:
 - For a Sun Java System Directory Server user store:
`im_admin_tools_dir\samples\NeteAuto\RoleDefinitions.xml`
 - For a Microsoft SQL Server user store:
`im_admin_tools_dir\samples\NeteAutoRDB\Organization\mssqlRoleDefinitions.xml`
 - For an Oracle user store:
`im_admin_tools_dir\samples\NeteAutoRDB\Organization\oracleRoleDefinitions.xml`

In this path, `im_admin_tools_dir` defines the installed location of the Identity Manager administrative tools.
7. Specify a user to serve as the System Manager for this environment and click Next:
 - a. Type SuperAdmin in the System Manager field.
 - b. Click Add.

Identity Manager adds the unique identifier of the SuperAdmin user to the list of users.
 - c. Click Next.
8. Review the settings for the environment, and do the following.
 - (Optional) Click Previous to make changes.
 - Click Finish to create the Identity Manager environment with the current settings

The Environment Configuration Output screen shows the progress of the environment creation.
9. Click Continue to exit the Identity Manager environment wizard.
10. Restart the application server where Identity Manager is running.

Once you create the NeteAuto environment, you can do the following:

- [Create a skin for this Identity Manager environment](#) (see page 40)
- [Access the environment](#) (see page 38)

How to Configure the NeteAuto Sample without Organization Support

Configuring the NeteAuto sample without organization support involves the following steps:

- Installing the [prerequisite software](#) (see page 23)
- Installing the sample Identity Manager environment
- Configuring the database
- Creating a JDBC data source
- Creating the Identity Manager directory
- Creating the NeteAuto Identity Manager environment

Sample Identity Manager Environment Description

For Microsoft SQL Server and Oracle databases, Identity Manager includes a version of the NeteAuto environment that does not include organizations. This Identity Manager environment includes the following three users:

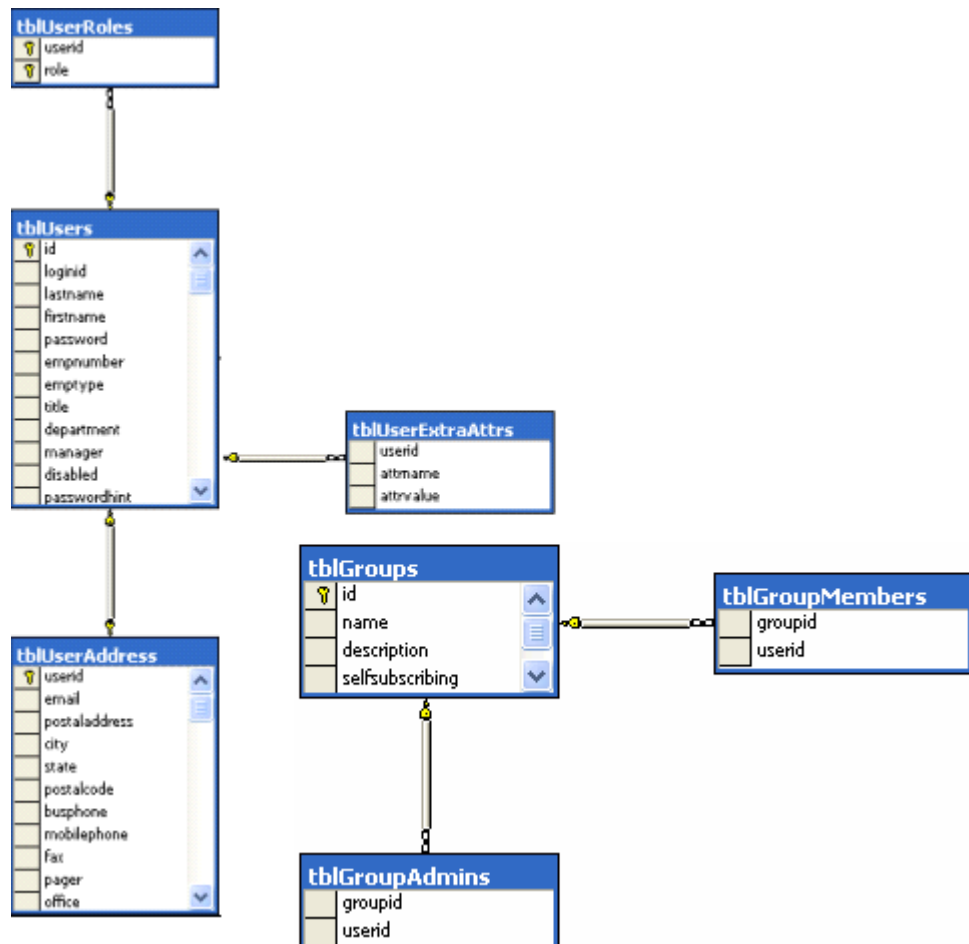
- SuperAdmin is the administrator account with the System Manager role for this Identity Manager environment. As SuperAdmin, you can perform all of the default admin tasks.

Note: For a description of the default admin tasks, see the *Administration Guide*.

- SelfRegUser is the administrator account that Identity Manager uses to enable self-registration for this Identity Manager environment.
- NeteAuto Administrator does not have any privileges when you install the NeteAuto environment.

However, you can assign the Group Manager role to the NeteAuto Administrator account.

The following illustration describes the NeteAuto sample for a relational database, without organizations:



Installation Files for the Neteauto Environment

Identity Manager includes a set of files that you can use to set up a sample Identity Manager environment. An Identity Manager environment is a view of a management namespace that enables Identity Manager administrators to manage objects, such as users and groups, with a set of associated roles and tasks. An Identity Manager environment controls the management and graphical presentation of a user store.

The sample Identity Manager environment includes the following:

- Sample users
- Role, task, and screen definitions

Tasks appear in the User Console when you click a category, such as users or groups. The tasks that appear are based on the roles that are assigned to the user who is logged in.

Note: For more information about roles and tasks, see the *Administration Guide*.

- A sample skin that customizes the User Console for NeteAuto users
- A directory configuration file that you use to create an Identity Manager directory

The files for creating the sample Identity Manager environment are installed in the following location:

`im_admin_tools_dir\samples\NeteAutoRDB\NoOrganization`

In this path, `im_admin_tools_dir` defines the installed location of the Identity Manager administrative tools, as shown in the following examples:

- **Windows:** C:\Program Files\CA\IAM Suite\tools\samples\NeteAutoRDB\NoOrganization
- **UNIX:** `home_directory/CA/IAM Suite/tools/samples/NeteAutoRDB/NoOrganization`

Install the NeteAuto Environment—No Organization Support

Perform the following process to install the NeteAuto environment.

To install the NeteAuto environment

1. Ensure that the [prerequisite software](#) (see page 34) is installed
2. Configure the Database
3. Create an ODBC Data Source
4. Create the Identity Manager Directory
5. Create the NeteAuto Identity Manager Environment
6. Configure the look and feel of the [Identity Manager user interface](#) (see page 40) for NeteAuto users

Prerequisite Software

The NeteAuto Identity Manager environment has the following prerequisites:

- Identity Manager must be installed as described in the *Installation Guide*. Be sure to install the Identity Manager Administrative Tools.
- You must have access to a Microsoft SQL Server or Oracle database.

Configure a Relational Database

Perform the following procedure to configure a relational database.

To configure a relational database

1. Create a database instance named NeteAuto.
2. Create a user named neteautoadmin with the password test. Grant neteautoadmin rights (such as public and db_owner rights) to NeteAuto by editing the properties of the user.

Note: The neteautoadmin must have at least select, insert, update and delete permissions for all the tables created by the .sql script for creating the NeteAuto database, and must be able to execute all of the stored procedures, if any, defined in these scripts.

3. While editing the user's properties, make NeteAuto the default database for neteautoadmin.
4. Run the following scripts in the order in which they are listed:
 - *db_type-rdbuserdirectory.sql*—Configures the tables for the NeteAuto sample, and creates the user entries
 - *ims_db_type_rdb.sql*—Configures support for organizations

db_type

Defines the mssql or oracle depending on the type of database that you are configuring.

These script files are located in the *im_admin_tools_dir*\samples\NeteAutoRDB\Organization folder.

5. Define a JDBC data source named neteautoDS that points to the NeteAuto database.

The procedure for configuring a data source depends on the type of application server where Identity Manager is installed. The section [How to Create a JDBC Data Source](#) (see page 97) includes application server-specific instructions for creating a JDBC data source.

Create the Identity Manager Directory

Perform the following procedure to create the Identity Manager directory.

To create the Identity Manager directory

1. Open the Management Console by entering the following URL in a browser:

`http://domain_name:port/idmmanage`

domain_name

Defines the fully qualified domain name of the server where Identity Manager is installed.

port

Defines the application server port number.

2. Click Directories.

The Identity Manager directories screen appears.

3. Click New to start the Identity Manager directory wizard.

4. Browse for one of the following directory configuration XML files, and click Next:

For Sun Java Systems:

The directory configuration file is the following:

`im_admin_tools_dir\samples\NeteAuto\NoOrganization\directory.xml`

In this path, *im_admin_tools_dir* defines the installed location of the Identity Manager administrative tools.

For SQL Server databases:

`im_admin_tools_dir\samples\NeteAuto\NoOrganization\mssql-directory.xml`

For Oracle databases:

`im_admin_tools_dir\samples\NeteAuto\NoOrganization\oracle-directory.xml`

Status information is displayed in the Directory Configuration Output screen.

5. In the second page of the wizard, supply the following values:

Name

NeteAutoRDB Directory

Description

Sample NeteAuto directory with no organization support

Connection Object Name

NeteAutoRDB

ODBC Data Source

neteautoDS

Username

neteautoadmin

Password

test

6. Click Next.
7. Click Finish to exit the wizard.

Create the NeteAuto Identity Manager Environment

Perform the following procedure to create the NeteAuto Identity Manager environment.

To create the NeteAuto Identity Manager environment

1. In the Management Console, click Environments.
2. In the Identity Manager environments screen, click New.
The Identity Manager environment wizard opens.
3. In the first page of the wizard, type the following values:
 - Environment name—NeteAuto environment
 - Description—This is a sample environment

- Alias—neteautoRDB

The alias is added to the URL for accessing the Identity Manager environment. For example, the URL for accessing the neteauto environment is:

`http://domain/idm/neteautoRDB`

In this path, *domain* defines the fully qualified domain name of the server where Identity Manager is installed, as in the following example:

`http://myserver.mycompany.org/idm/neteautoRDB`

Note: The alias is case-sensitive.

Click Next.

4. Select the NeteAutoRDB Directory Identity Manager directory to associate with the environment you are creating and click Next.
5. Configure support for public tasks, such as the self-registration and forgotten password tasks.

Note: Users do not need to log in to access public tasks.

- a. Type the following alias for public tasks:

`neteautoRDBpublic`

- b. Type SelfRegUser as the anonymous user account.
- c. Click Validate to view the user's unique identifier (2, in this case).

6. Select the tasks and roles to create for the NeteAuto environment:

- Select Import roles from the file.

- Browse to the following location:

`im_admin_tools_dir\samples\NeteAutoRDB\NoOrganizations\RoleDefinitions.xml`

In this path, *im_admin_tools_dir* defines the installed location of the Identity Manager administrative tools.

7. Specify a user to serve as the System Manager for this environment, and click Next:
 - a. Type SuperAdmin in the System Manager field.
 - b. Click Add.
 - c. Click Next.

8. Review the settings for the environment.

- Click Previous to make changes.
- Click Finish to create the Identity Manager environment with the current settings

The Environment Configuration Output screen shows the progress of the environment creation.

9. Click Finish to exit the Identity Manager environment wizard.

10. Start the Identity Manager environment.

Once you created the NeteAuto environment, you can do the following:

- Create a skin for this Identity Manager environment as described in [Set up the NeteAuto Skin](#) (see page 40)
- Access the environment as described in Using the NeteAuto Identity Manager environment

How to Use the NeteAuto Identity Manager Environment

You can use the NeteAuto Identity Manager Environment to manage self-service tasks and users.

Self-Service Task Management

The self-service tasks include the following:

- Registering as a new user
- Logging in as a self-registered user
- Using the forgotten password feature

Register as a New User

Perform the following procedure to register as a new user.

To register as a new user

1. Type the following URL in a browser:

`http://domain/idm/neteautopublic/imcss/index.jsp?task.tag=SelfRegistration`

In this example, *domain* defines the fully qualified domain name of the system where Identity Manager is running.

Note: If you did not [configure the Neteauto skin](#) (see page 40), you can omit imcss from the URL as follows:

`http://domain/idm/neteautopublic/index.jsp?task.tag=SelfRegistration`

This URL directs you to the [default ca console](#) (see page 183).

At the Self-Registration: End User License Agreement page, Identity Manager displays the CA website.

Note: You can configure the default Self-Registration task to display your company's End User License Agreement. For instructions, see the *Administration Guide*.

2. Click Accept to proceed.
3. In the Profile tab, do the following:
 - a. Type values for the required fields, indicated by an asterisk (*).
 - b. Type password hints and answers.

In case of a forgotten password, Identity Manager provides the password hint and requests the answer. If the answer is correct, Identity Manager prompts the user to specify and confirm a new password.

4. Leave the Groups tab unchanged.
5. Click Submit.

Log In as a Self-Registered User

Perform the following procedure to log in as a self-registered user.

To log in as a self-registered user

1. Type the following URL for the NeteAuto Identity Manager environment in a browser:

`http://domain/idm/neteauto/imcss/index.jsp`

In this example, *domain* defines the fully qualified domain name of the system where Identity Manager is running.

2. Log in using the username and password you specified when you registered.

Set Up the NeteAuto Skin

To set up the NeteAuto skin, you create a SiteMinder response in the SiteMinder Policy Server.

Note: The following procedure refers to the Policy Server User Interface, which is available with SiteMinder r6.0 SP 5 and SiteMinder r12. SiteMinder r12 also includes the Administrative UI, which is a browser-based console that you can use to manage the Policy Server. For information on using the Administrative UI, see the *CA SiteMinder Web Access Manager Policy Server Administration Guide*.

To set up the NeteAuto skin

1. In the Domains tab in the Policy Server user interface, expand neteautoDomain.
2. Under neteautoDomain, expand Realms.

The following realms appear:

neteauto_ims_realm

Protects the Identity Manager environment.

neteauto_pub_realm

Enables support for public tasks, such as self-registration and forgotten password tasks.

3. Create a rule in each of the realms. Specify the following:

- Resource: *
- Actions: GET, POST

To simplify the administration, include the NeteAuto skin in the rule name.

4. Create a response as follows:
 - a. Expand the Policy Domain from Step 1. Right-click Responses and click Create Response.
The Response Properties dialog appears.
 - b. Click Create.
The Response Attribute dialog appears.
 - c. Ensure WebAgent-HTTP-Header-Variable is selected.
 - d. Select Static in the Attribute Kind box.
 - e. In the Variable Name field, type skin.
 - f. In the Variable Value field, type neteauto.
5. Modify the policy that Identity Manager created in the neteautoDomain. Specify the following:
 - In the Users tab, click Add/Remove and proceed as follows:
 - For LDAP: Select ou=People, ou=Employees, ou=NeteAuto in Available Members and add it to Current Members. Click OK.
 - For relational databases: Search for users where the id attribute equals *. Select all the users in Available Members and add them to Current Members. Click OK.
 - In the Rules tab, click Add/Remove. Add the rules that you created in Step 3.
For each rule, click Set Response. Associate each rule with the response that you created in Step 4.

Note: The neteauto skin is based on the [imcss console](#) (see page 183). To view the skin, append /imcss/index.jsp to the URL for the NeteAuto Identity Manager environment as follows:

`http://domain/idm/neteauto/imcss/index.jsp`

[Access the NeteAuto Identity Manager Environment](#) (see page 42) provides complete instructions for accessing the Neteauto environment.

Use the Forgotten Password Feature

Perform the following procedure to use the forgotten password feature.

To use the forgotten password feature

1. Type the following URL in a browser:
`http://domain/idm/neteautopublic/index.jsp?task.tag=ForgottenPasswordReset`
In this URL, *domain* defines the fully qualified domain name of the system where Identity Manager is running.
2. Type the unique identifier for the self-registered user that you created in [Register as a New User](#) (see page 39), and click Next.
3. Each time that you are prompted, answer the verification question that you provided when you registered.
4. Click Submit.
Identity Manager prompts you to supply a new password.

User Management

User management includes the following operations:

- Accessing the NeteAuto Identity Manager environment
- Modifying a user
- Assigning the Group Manager role
- Creating a group
- Managing self-registered users

Access the NeteAuto Identity Manager Environment

Perform the following procedure to access the NeteAuto Identity Manager environment.

To access the sample Identity Manager environment

1. Type the following URL in a browser:
`http://domain/idm/neteauto/imcss/index.jsp`
In this path, *domain* defines the fully qualified domain name, as in the following example:
`http://myserver.mycompany.com/idm/neteauto/imcss/index.jsp.`
Note: If you did not configure the Neteauto skin, you can use the following URL to access the Neteauto environment:
`http://domain/idm/neteauto`

2. In the login screen, type the following credentials:

User Name

SuperAdmin

Password

test

Modify a User

Perform the following procedure to modify a user.

To modify a user

1. Log in the NeteAuto environment as SuperAdmin using the password test.
2. Select Users, Manage Users, Modify User.
The Select User screen appears.
3. Click Search.
Identity Manager displays a list of users in the NeteAuto environment.
4. Select the NeteAuto administrator, as follows:
 - For LDAP directories, NeteAuto Administrator
 - For relational databases, NeteAuto AdminClick Select. Identity Manager displays the profile for the NeteAuto administrator.
5. In the Title field, type Manager. Click Submit.
Identity Manager confirms the task submission.
6. Click OK to return to the main screen.

Assign the Group Manager Role

The group manager role must be assigned.

To assign the group manager role

1. As SuperAdmin, select the Roles and Tasks tab, then select Admin Roles, Modify Admin Roles.
2. Select the Group Manager role, and click Select.
The profile for the Group Manager role appears.
3. Click the Members tab, and click Add under Member Policies.
The Member Policy screen appears.

4. Under Member Rule, click the down arrow in the Users field.
From the drop-down list, select where <user-filter>.
The Users field changes to let you enter a filter for the rule.
5. Enter a membership rule as follows:
 - a. In the first field, select Title from the drop-down list.
 - b. In the second field, make sure the equal sign (=) is selected.
 - c. In the third field, type Manager.
6. In the Scope Rules section, define rules for the users, groups, and organizations (when supported) as follows:
 - a. In the Users field, click the down arrow to see a list of options. Select (all) from the list.
 - b. Repeat Step a in the Group and Organization fields (when supported).
 - c. Leave the Access Tasks field blank.
7. Click OK.
Identity Manager displays the member policy that you created.
8. Click Submit.
Identity Manager confirms the task submission.
9. Click OK to return to the main screen.
10. Close Identity Manager.

Create a Group

Perform the following procedure to create a group.

To create a group

1. Log in to Identity Manager as the NeteAuto administrator, as follows:
 - For LDAP directories, type the user name NeteAuto Administrator and the password test.
 - For relational databases, type the user name NeteAuto Admin and the password test.

The list of tasks that the NeteAuto administrator can perform appears. Because the NeteAuto administrator can perform only a small number of tasks, Identity Manager lists the tasks instead of categories.
2. Click Create Group.
3. Make sure Create a new group is selected, and click OK.

4. Implement one of the following steps that fits your case:
 - If the NeteAuto environment supports organizations, do the following:
 - a. In the Org Name field, click the ellipsis symbol (...) to select the organization where Identity Manager will create the group.
 - b. At the bottom of the Select Organization screen, expand NeteAuto.
 - c. Select the Dealer organization.
 - If the NeteAuto environment does not support organizations, go to the next step.
5. Type the following information for the group:
 - Group Name: Dealer Administrators
 - Group Description: Administrators for NeteAuto dealerships.
6. Click the Membership tab and click the Add a user.
The Select User screen appears.
7. Click Search.
8. Select the NeteAuto administrator, and click Select.
9. Click Submit to create the group.

Manage Self-Registered Users

Perform the following procedure when you want to manage self-registered users.

To manage self-registered users

1. Log in to Identity Manager as NeteAuto administrator, using the following credentials:
 - For LDAP directories:
 - Username**
NeteAuto Administrator
 - Password**
test
 - For relational databases:
 - Username**
NeteAuto Admin
 - Password**
test

The list of tasks that the NeteAuto administrator can perform appears on the left side of the User Console. Because the NeteAuto administrator can perform only a small number of tasks, Identity Manager lists the tasks instead of the categories.

2. Click Modify Group.

3. Click Search.

Identity Manager displays a list of groups.

4. Select Dealer Administrators, and click Select.

5. Click the Membership tab, and click Add a user.

The Select User screen appears.

6. Click Search.

7. In the User Search screen, select the user that you typed in [Register as a New User](#) (see page 39). Click Select.

8. Click Submit.

Identity Manager confirms the task submission.

9. Click OK to return to the main screen.

To confirm that the user is a member of the group that you created, use the View Group task.

How to Configure Additional Features

After you have installed the NeteAuto sample and practiced basic Identity Manager functionality, you may want to use the NeteAuto environment to practice and test additional Identity Manager features, including email notifications and workflow.

Note: For more information about these features, see the *Administration Guide*.

Chapter 4: Creating an Identity Manager Directory for an LDAP User Store

This section contains the following topics:

- [Identity Manager Directories](#) (see page 47)
- [How to Create an Identity Manager Directory](#) (see page 48)
- [Directory Structure](#) (see page 48)
- [Directory Configuration File](#) (see page 50)
- [How to Select a Directory Configuration Template](#) (see page 51)
- [How to Describe a User Directory to Identity Manager](#) (see page 53)
- [Connection to the User Directory](#) (see page 54)
- [Directory Search Parameters](#) (see page 58)
- [User, Group and Organization Managed Object Descriptions](#) (see page 59)
- [Well-Known Attributes for an LDAP User Store](#) (see page 75)
- [Describe the User Directory Structure](#) (see page 82)
- [How to Configure Groups](#) (see page 83)
- [Validation Rules](#) (see page 86)
- [Additional Identity Manager Directory Properties](#) (see page 86)
- [How to Improve Directory Search Performance](#) (see page 91)

Identity Manager Directories

An *Identity Manager directory* describes how objects such as users, groups, and organizations are stored in the user directory and represented in Identity Manager. An Identity Manager directory is associated with one or more Identity Manager environments.

How to Create an Identity Manager Directory

Creating an Identity Manager directory for an LDAP user store involves the following steps:

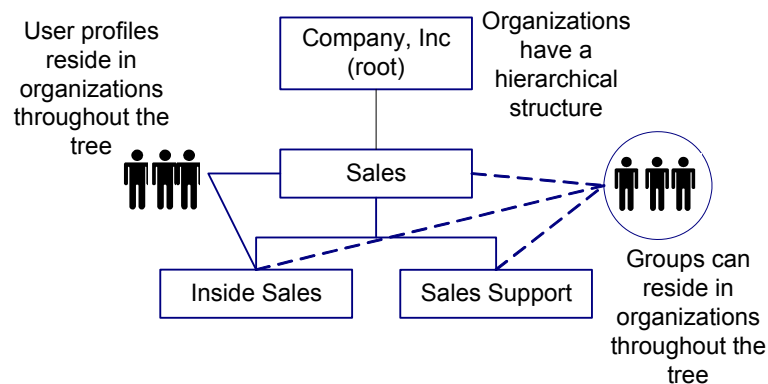
1. Determining the directory structure
2. Describing the objects in the user store by modifying a [directory configuration file \(directory.xml\)](#) (see page 53)
3. Importing the directory configuration file and [creating the directory](#) (see page 135)

Note: If you are using SiteMinder, ensure that you have applied the policy store schema prior to creating an Identity Manager Directory. For more information on specific policy store schemas and how to apply them, see the *Installation Guide*.

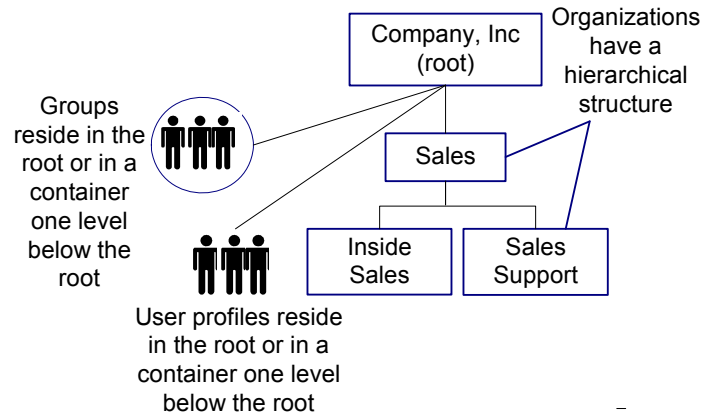
Directory Structure

Identity Manager supports the following directory structures:

- Hierarchical—Contains a parent organization (root) and suborganizations. The suborganizations may also have suborganizations, which creates a multi-level structure, as shown in the following illustration:

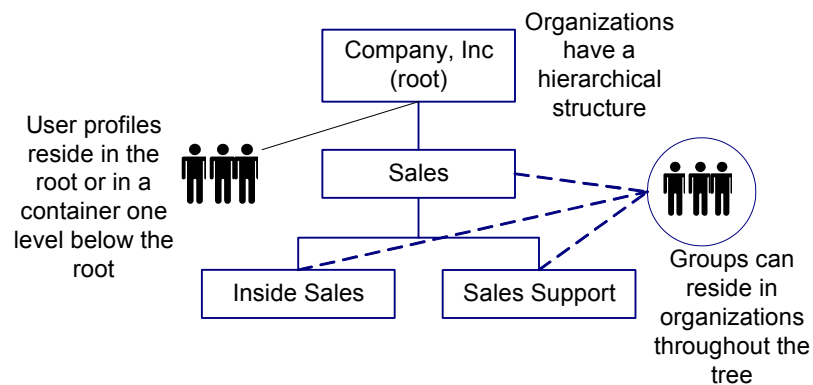


- Flat—User and groups are stored at the search root or in a container one level below the search root. Organizations have a hierarchical structure, as shown in the following illustration of a flat directory structure:



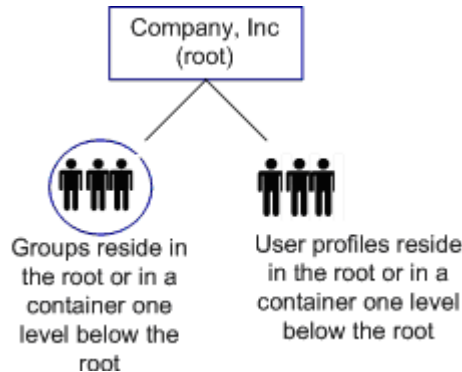
To facilitate user management and delegation in flat directory structures, users and groups belong to logical organizations. The logical organization is stored as an attribute in user and group profiles.

- Flat User—Organizations and groups are stored hierarchically, but users are stored at the search root or in a container one level below the search root, as shown in the following illustration of a flat user directory structure:



In flat user directory structures, users belong to logical organizations. A user's logical organization is stored as an attribute in a user's profile.

- No organizations—The directory does not include organizations. Users and groups are stored at the search root or in a container one level below the search root. A no-organizations directory structure is shown in the following illustration:



Note: A directory may contain more than one type of structure. For example, user profiles may be stored in a flat structure in one part of the directory and hierarchically in another. To support a hybrid directory structure, create multiple Identity Manager environments.

Directory Configuration File

To describe a user directory's structure to Identity Manager, you must create a directory configuration file.

The directory configuration file contains one or more of the following sections:

Identity Manager Directory Information

Contains information about the Identity Manager directory that is used by Identity Manager.

Note: Do not modify information in this section. Identity Manager prompts you to provide this information when you create an Identity Manager directory in the Management Console.

Attribute Validation

Defines the validation rules that apply to the Identity Manager directory.

Provider Information

Describes the user store that Identity Manager will manage.

Directory Search Information

Enables you to specify how Identity Manager searches the user store.

User Object

Describes how users are stored in the user store and represented in Identity Manager.

Group Object

Describes how groups are stored in the user store and represented in Identity Manager.

Organization Object

If the user store includes organizations, describes how organizations are stored and represented in Identity Manager.

Self-Subscribing Object

Configures support for groups that self-service users can join.

Directory Groups Behavior

Specifies whether the Identity Manager directory supports dynamic and nested groups.

To create a directory configuration file, you modify a configuration template.

How to Select a Directory Configuration Template

Identity Manager supplies directory configuration templates that support different directory types and structures. To create an Identity Manager directory, modify the template that most closely matches your directory structure.

The templates described in the following table are located in the Administrative Tools directory:

im_admin_tools_dir\directoryTemplates\directory\

im_admin_tools_dir

Defines the installed location of Identity Manager administrative tools

directory

Defines the type of directory, as in the following examples:

- **Windows:** C:\Program Files\CA\CA IAM Suite\directoryTemplates\ActiveDirectory
- **UNIX:** HOME/CA/CA IAM Suite/directoryTemplates/iPlanetFlat

The types of directories and the corresponding configuration templates are shown in the following table:

Directory Type	Template
Active Directory (ADSI) LDAP directory with a hierarchical structure	ActiveDirectory\directory.xml
Microsoft ADAM Directory with a hierarchical structure	ADAM\directory.xml
IBM Directory Server directory with a hierarchical structure	IBMDirectoryServer\directory.xml
Novell eDirectory user directory with a hierarchical structure	eDirectory\directory.xml
Oracle Internet Directory with a hierarchical structure	OracleInternetDirectory\directory.xml
Sun Java System (SunOne or iPlanet) LDAP directory with a hierarchical structure	IPlanetHierarchical\directory.xml
Sun Java System (SunOne or iPlanet) LDAP directory with a flat structure	IPlanetFlat\directory.xml
Sun Java System (SunOne or iPlanet) LDAP directory that does not include organizations	IPlanetNoOrganizations\directory.xml
eTrust Directory user store with a hierarchical structure	eTrustDirectory\directory.xml
Provisioning Directory This template configures the Provisioning Directory for an Identity Manager environment. It is only for use in Identity Manager implementations that include provisioning. See the <i>Provisioning Guide</i> for more information. Note: You can use this configuration template as installed. You do not need to modify this template.	ProvisioningServer\directory.xml
Custom directory	Use the template that most closely resembles your directory

Copy the configuration template to a new directory or save it with a different name to prevent overwriting it.

How to Describe a User Directory to Identity Manager

To manage a directory, Identity Manager must understand the directory's structure and content. To describe the directory to Identity Manager, modify the directory configuration file (directory.xml) in the appropriate template directory.

The directory configuration file has the following important conventions:

- **##**—Indicates required values.
To provide all the required information, locate all double pound signs (##) and replace them with appropriate values. For example, ##DISABLED_STATE indicates that you must supply an attribute to store the status of a user's account.
- **@**—Indicates values that Identity Manager populates. Do not modify these values in the directory configuration file. Identity Manager prompts you to supply the values when you import the directory configuration file.

Before you modify the directory configuration file, you need the following information:

- LDAP object classes for the user, group, and organization objects
- List of attributes in user, group, and organization profiles

How to Modify the Directory Configuration File

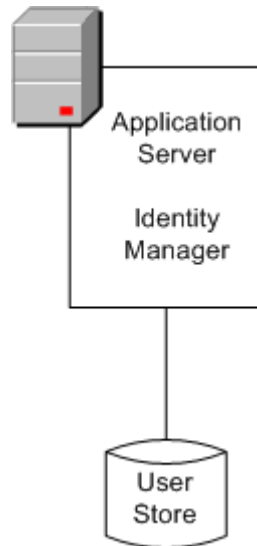
Perform the following steps to modify the directory configuration file.

Note: Steps that are required are noted accordingly.

1. Limit the size of [search results](#) (see page 58).
2. Modify the default user, organization, or group managed objects.
3. Change the default attribute descriptions.
4. Modify [well-known attributes](#) (see page 75). (required)
Well-known attributes identify special attributes, such as the password attribute, in Identity Manager.
5. [Configure Identity Manager for your directory structure](#) (see page 82) (required).
6. Enable users to [subscribe to groups](#) (see page 83).

Connection to the User Directory

Identity Manager connects to a user directory to store user, group, and organizational information, as shown in the following illustration:



A new directory or database is not required. However, the existing directory or database must be located on a system that has a fully qualified domain name (FQDN).

For a list of supported directory and database types, see the [CA support site](#). Search for the Identity Manager support matrix.

You configure a connection to the user store when you create an Identity Manager directory in the Management Console.

If you export the directory configuration after creating an Identity Manager directory, the user directory connection information is displayed in the Provider element of the directory configuration file.

Provider Element

Configuration information is stored in the Provider element and its subelements in the directory.xml file.

Note: If you are creating a new Identity Manager directory, you do not need to provide directory connection information in the directory.xml file. You provide connection information in the Identity Manager Directory wizard in the Management Console. Modify the Provider element for updates only.

The Provider element includes the following subelements:

LDAP

Describes the user directory to which you are connecting.

Credentials

Provides the user name and password for accessing the LDAP user store.

Connection

Supplies the host name and port for the computer where the user store is located.

Provisioning Domain

Defines the Provisioning Domain that Identity Manager manages (for provisioning users only)

A completed Provider element may resemble the following:

```
<Provider type="LDAP" userdirectory="@SMDirName">
  <LDAP searchroot="@SMDirSearchRoot" secure="@SMDirSecure" />
  <Credentials user="@SMDirUser"
    cleartext="true">@SMDirPassword</Credentials>
  <Connection host="@SMDirHost" port="@SMDirPort" />
  <eTrustAdmin domain="@SMDirETrustAdminDomain" />
</Provider>
```

The Provider element includes the following parameters:

type

Specifies the type of the database. For all LDAP user stores, specify LDAP (default).

userdirectory

Specifies the name of the user directory connection.

Note: Do not specify a name for the user directory connection in the directory.xml file. Identity Manager prompts you to supply the name when you create the Identity Manager directory in the Management Console.

Note: The parameters are optional.

LDAP Subelement

The LDAP subelement includes the following parameters:

searchroot

Specifies the location in an LDAP directory that serves as the starting point for the directory—typically, an organization (o) or organizational unit (ou).

secure

Forces a Secure Sockets Layer (SSL) connection to the LDAP user directory, as follows:

- True—Identity Manager uses a secure connection.
- False—Identity Manager connects to the user directory without SSL (default).

Note: The parameters are optional.

Credentials Subelement

To connect to an LDAP directory, Identity Manager must provide valid credentials. The credentials are defined in the Credentials subelement, which resembles the following:

```
<Credentials user="@SMDirUser" cleartext="true">  
  "MyPassword"  
</Credentials>
```

If you do not specify a password in the Credentials subelement, you are prompted for the password when you create the Identity Manager directory in the Management Console.

Note: We recommend specifying the password in the Management Console.

If you specify the password in the Management Console, Identity Manager encrypts the password for you. Otherwise, if you do not want the password to appear in clear text, you must encrypt it using the password tool installed with Identity Manager.

Note: You can specify only one set of credentials. If you define multiple directories, as described in Connection Subelement, the credentials that you specify must apply to all the directories.

The Credentials subelement includes the following parameters:

user

Specifies the login ID for an account that can access the directory.

For provisioning users, the user account that you specify must have the Domain Administrator profile or an equivalent set of privileges in the Provisioning Server.

Note: Do not specify a value for the user parameter in the directory.xml file. Identity Manager prompts you to supply the login ID when you create the Identity Manager directory in the Management Console.

cleartext

Determines whether the password is displayed in clear text in the directory.xml file, as follows:

- True—The password is displayed in clear text.
- False—The password is encrypted (default).

Note: The parameters are optional.

Connection Subelement

The Connection subelement describes the location of the user store that Identity Manager will manage.

You can specify multiple Connection subelements to define load balancing for user directories. When load balancing is configured, LDAP queries are distributed evenly to the defined user directories, starting with the directory defined in the first Connection subelement. For example when Directory 1 and Directory 2 are configured for load balancing, the first LDAP query is made against Directory 1 and the second query is made against Directory 2.

Note: If you configure load balancing, you must be able to access all the configured directories using one set of credentials.

The Connection subelement includes the following parameters:

host

Specifies the host name or IP address of the system where the user directory is located.

Note: If the system that you are connecting to has an IPv6 address, enclose the IP address in brackets ([]), as follows:

```
<Connection host="[2::9255:214:22ff:fe72:525a]" port="20389"
failover="[2::9255:214:22ff:fe72:525a]:20389"/>
```

port

Specifies the port number for the user directory.

failover

Specifies the host name and IP address of the system where redundant user stores exist, in case the primary system is unavailable. If multiple servers are listed, Identity Manager attempts to connect to the systems in the order in which they are listed.

Specify the host name and IP address in the failover attribute in a *space-separated* list, as follows:

```
failover="IPAddress:port IPAddress:port"
```

For example:

```
<Connection host="123.456.789.001" port="20389"
```

```
failover="123.456.789.002:20389 123.456.789.003:20389"/>
```

Note: Port 20389 is the default port for the Provisioning Server.

Note: The parameters are optional.

Provisioning Subelement

If Identity Manager environment includes provisioning, define the Provisioning Domain as follows:

```
<Provisioning domain="@SMDirProvisioningDomain" />
</Provider>
```

The Provisioning subelement includes the following parameter:

domain

Contains the name of the Provisioning Domain that Identity Manager will manage.

If you do not specify a value for the domain parameter in the directory configuration file (directory.xml), you are prompted for the domain name when you create the Identity Manager directory in the Management Console.

Directory Search Parameters

You can set the following search parameters in the DirectorySearch element:

maxrows

Defines the maximum number of results that Identity Manager can return when searching a user directory. When the number of results exceeds the limit, an error is displayed.

By setting a value for the `maxrows` parameter in the `directory.xml` file, you can override the settings in the Policy server and LDAP directory that limit search results. When conflicting settings apply, the LDAP server uses the lowest setting.

Note: The `maxrows` parameter does not limit the number of results that are displayed on an Identity Manager task screen. To configure display settings, modify the list screen definition in the Identity Manager User Console. For instructions, see the *Administration Guide*.

timeout

Determines the maximum number of seconds that Identity Manager searches a directory before terminating the search.

Note: The `DirectorySearch` element is optional.

User, Group and Organization Managed Object Descriptions

In Identity Manager, you manage the following types of objects that correspond to entries in a user directory:

Users

Represent users in an enterprise. A user belongs to a single organization.

Groups

Represent associations of users who have something in common.

Organizations

Represent business units. Organizations may contain users, groups, and other organizations.

An object description contains the following information:

- Information about the [object](#) (see page 108), such as the LDAP object class and the container in which objects are stored.
- The [attributes that store information about an entry](#) (see page 113). For example, the `pager` attribute stores a pager number.

Note: An Identity Manager environment supports only one type of user, group and organization object. For example, all user objects have the same object class.

Managed Object Descriptions

A managed object is described by specifying object information in the User Object, Group Object and Organization Object sections of the directory configuration file.

Note: When you use the configuration template (directory.xml file) for user directories that do not support organizations, there is no Organization Object section.

Each of these sections contains `ImsManagedObject` elements, such as the following:

```
<ImsManagedObject name="User" description="My Users"
objectclass="top,person,organizationalperson,inetorgperson" objecttype="USER">
```

Optionally, the `ImsManagedObject` element may include a `Container` element, such as the following:

```
<Container objectclass="top,organizationalUnit" attribute="ou" value="people" />
```

Specify Object Information

Object information is specified by supplying values for various parameters.

To specify object information

1. Locate the `ImsManagedObject` element in the User Object, Organization Object or Group Object section.
2. Supply values for the following parameters:

Name

Specifies a unique name for the managed object.

Note: This parameter is required.

Description

Contains a description of the managed object.

ObjectClass

Specifies the name of the LDAP object class for the object type (user, group, or organization). The object class determines the list of available attributes for an object.

If attributes from multiple object classes apply to an object type, list the object classes in a comma-delimited list. For example, if an object contains attributes from the `person`, `organizationalperson` and `inetorgperson` object classes, add these object classes as follows:

```
objectclass="top,person,organizationalperson,inetorgperson"
```

Each LDAP directory includes a set of predefined object classes. Refer to the directory server documentation for information about predefined object classes

Note: This parameter is required.

ObjectType

Specifies the type of the managed object. The valid values are as follows:

- User
- Organization
- Group

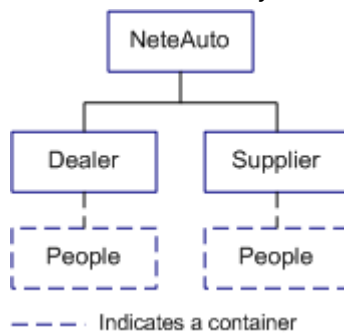
Note: This parameter is required.

3. Optionally, supply container information.

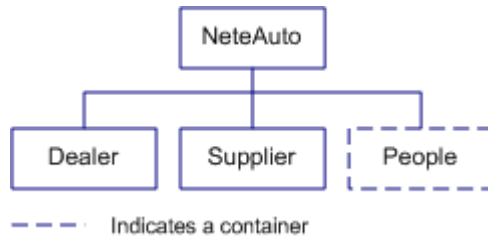
Containers

To simplify administration, you can group objects of a specific type in a container. When you specify a container in the directory configuration file, Identity Manager manages only entries in the container. For example, if you specify a user container named `People`, Identity Manager manages users in the `People` container, as shown in the following illustrations:

- Hierarchical Directory



- Flat Directory



In these examples, all users exist in the People containers.

When you specify a container, note the following:

- If no container exists in an organization, Identity Manager creates the container when the first entry is added. For a hierarchical directory, Identity Manager creates the container in the organization where the entry is added. For flat directories and directories that do not support organizations, Identity Manager creates the container under the search root, which you specify when you create the Identity Manager directory.
- Identity Manager ignores entries that are not in the specified container. For example, when you specify the People container, you cannot manage users that exist outside of the People container.

Note: To manage users who are not in the specified container, you can create another Identity Manager environment.

Specify a User or Group Container

Perform the following procedure to specify a user or group container.

To specify a user or group container

1. Locate the Container element in the User Object or Group Object section.
2. Supply values for the following parameters:

objectclass

Determines the LDAP object class of the container where objects of a specific type are created. For example, the default value for the user container is "top,organizationalUnit," which indicates that users are created in LDAP organizational units (ou).

When you are managing dynamic or nested groups, be sure to specify an objectclass that [supports these group types](#) (see page 84).

Note: This parameter is required.

attribute

Specifies the attribute that stores the container name, for example, ou.

The attribute is paired with the value to form the relative DN of the container, as in the following example:

ou=People

Note: This parameter is required.

value

Specifies the name of the container.

Note: This parameter is required.

Note: You cannot specify containers for organizations.

Attribute Descriptions

An attribute stores information about an entry, such as a telephone number or address. An entry's attributes determine its profile.

In the directory configuration file, attributes are described in `ImsManagedObjectAttr` elements. In the User Object, Group Object and Organization Object sections of the directory configuration file, you can do the following:

- Modify default attribute descriptions to describe the attributes in your user store.
- Create new attribute descriptions by copying an existing description and modifying values as needed.

There is one `ImsManagedObjectAttr` element for each attribute in user, group, and organization profiles. For example, an `ImsManagedObjectAttr` element may describe a user ID.

An `ImsManagedObjectAttr` element resembles the following:

```
<ImsManagedObjectAttr physicalname="uid" displayname="User ID" description="User ID" valuetype="String" required="true" multivalued="false" wellknown="%USER_ID%" maxlength="0" />
```

The `ImsManagedObjectAttr` has the following parameters:

physicalname

This parameter must contain one of the following items:

- The name of the LDAP attribute where the profile value is stored. For example, a user's ID is stored in the `uid` attribute in the user directory.

Note: To improve performance, index LDAP attributes that are used in search queries in the User Console.

- A [well-known attribute](#) (see page 75). When you supply a well-known attribute, Identity Manager computes the value automatically. For example, when you specify the well-known attribute `%ORG_MEMBERSHIP%`, Identity Manager determines the organization to which the entry belongs based on the entry's DN.

description

Contains the description of the attribute

displayname

Specifies a unique name for the attribute.

In the User Console, the display name appears in the list of attributes that are available to add to a task screen. This parameter is required.

Note: Do not modify an attribute's `displayname` in the directory configuration file (`directory.xml`). To change the name of the attribute on a task screen, you can specify a label for the attribute in the task screen definition. For more information, see the *Administration Guide*.

valuetype

Specifies the attribute's data type. The valid values are as follows:

- String (Default value)
- Number
- Integer
- Date

required

Indicates whether the attribute is required, as follows:

- True—The attribute is required
- False—The attribute is optional (default)

Note: If an attribute is required by the LDAP directory server, set the `required` parameter to `true`.

multivalued

Indicates whether the attribute can have multiple values. For example, the group membership attribute is multi-valued to store the user DN of each group member. The valid values are as follows:

- True—The attribute can have multiple values
- False—The attribute can have only a single value (default)

Important! The Group Membership and Admin Roles attributes in the User object definition must be multivalued.

wellknown

Defines the name of the well-known attribute.

[Well-known attributes have a specific meaning in Identity Manager](#) (see page 75). They are identified by the following syntax:

%ATTRIBUTENAME%

maxlength

Defines the maximum length that an attribute's value can have. Set the maxlength parameter to 0 to specify an unlimited length.

Note: This parameter is required.

permission

Indicates whether an attribute's value can be modified in a task screen. The valid values are as follows:

- READONLY—The value is displayed but cannot be modified
- WRITEONCE—The value cannot be modified once the object is created. For example, a user ID cannot be changed after the user is created
- READWRITE—The value can be modified (default)

hidden

Indicates whether an attribute appears in Identity Manager task forms. The valid values are as follows:

- True—The attribute is not displayed to users.
- False—The attribute is displayed to users (default)

Logical attributes use hidden attributes.

Note: For more information, see the *Programming Guide for Java*.

system

Specifies attributes that are used by Identity Manager only, and should not be modified by users in the User Console. The valid values are as follows:

- True—Users should not modify the attribute. The attribute will not appear in the Identity Manager user interface.
- False—Users can modify this attribute. It is available to add to task screens in the Identity Manager user interface. (default)

validationruleset

Associates a validation rule set with the attribute.

The validation rule set that you specify must be defined in a [ValidationRuleSet element in the directory configuration file](#) (see page 223).

objectclass

Indicates the LDAP auxiliary class for a user, group, or organization attribute when the attribute is not part of the primary objectclass specified in the `ImsManagedObject` element.

For example, suppose the primary object class for users is `top`, `person`, `organizationalperson`, which defines the following user attributes:

- common name (cn)
- surname (sn)
- user id (uid)
- password (userPassword)

To include the attribute `employeeID`, which is defined in the `Employee` auxiliary class, you would add the following attribute description:

```
<ImsManagedObjectAttr physicalname="employeeID" displayname="Employee ID"
description="Employee ID" valuetype="String" required="true" multivalued="false"
maxlength="0" objectclass="Employee"/>
```

Specify Attribute Descriptions

Describing attributes involves the following steps:

1. Read the relevant sections among the following topics:
 - [eTrust Directory Considerations](#) (see page 73)
 - [Microsoft Active Directory Considerations](#) (see page 74)
 - [IBM Directory Server Considerations](#) (see page 74)
 - [Oracle Internet Directory Considerations](#) (see page 75)

2. In the User Object, Group Object and Organization Object sections of the directory configuration file, do the following:
 - Modify default attribute descriptions to describe your directory attributes.
 - Create new attribute descriptions by copying an existing description and modifying values as needed.

Note: If you specify a physical attribute when you create a new attribute description, the physical attribute must exist in the object class (or classes) that you specified for the object type.
3. (Optional) Change the display settings for the attribute to prevent displaying sensitive information, such as passwords or salaries, in the User Console.
4. (Optional) Configure a default sort order.
5. If you are managing a directory with a Flat or Flat User structure or a directory that does not include organizations, go to [Describe the User Directory Structure](#) (see page 82).

Managing Sensitive Attributes

Identity Manager provides the following methods for managing sensitive attributes:

- Data classifications for attributes

Data classifications allow you to specify display and encryption properties for attributes in the directory configuration file (directory.xml).

You can define data classifications that manage sensitive attributes as follows:

- Display the value of an attribute as a series of asterisks in Identity Manager task screens.

For example, you can display passwords as asterisks instead of displaying them in clear text.

- Hide the attribute value in View Submitted Task screens

This enables you to hide attributes, such as salary, from administrators who need to view task status in Identity Manager but do not need to view salary details.

- Encrypt an attribute

- Field styles in task profile screens

If you do not want to modify an attribute in the directory.xml file, you can set the display property for the attribute in screen definitions where the sensitive attribute appears.

The field style enables you to display attributes, such as passwords, as a series of asterisks instead of clear text.

Note: For more information about the field style for sensitive attributes, see the *Administration Guide* and the User Console online help.

Data Classification Attributes

The Data Classification element provides a way to associate additional properties with an attribute description. The values in this element determine how Identity Manager handles the attribute. This element supports the following parameters:

- sensitive

Causes Identity Manager to display the attribute as a series of asterisks (*). This prevents the attribute from appearing in clear text.

For example, you may configure the password attribute as sensitive.

If you create a copy of an existing user in the User Console, this parameter also prevents the attribute from being copied to the new user.

- vst_hide

Hides the attribute in the Event Details screen for the View Submitted Tasks tab. Unlike sensitive attributes, which are displayed as asterisks, vst_hidden attributes are not displayed.

You can use this parameter to prevent changes to an attribute, such as salary, from displaying in View Submitted Tasks.

- attributelevelencrypt

Encrypts attribute values when they are stored in the user store. Identity Manager uses RC2 encryption or FIPS 140-2 encryption, if Identity Manager is FIPS 140-2 enabled.

The attributes appear in clear text during runtime.

Note: For more information about FIPS 140-2 support in Identity Manager, see the *Administration Guide*.

- previouslyencrypted

Causes Identity Manager to detect and decrypt any encrypted values in the attribute when it accesses the object in the user store.

You use this data classification to decrypt any previously encrypted values.

The clear text value will be saved to the store when you save the object.

Configure Data Classification Attributes

To configure data classification attributes

1. Locate the attribute in directory configuration file.
2. After the attribute description, add the following:

```
<DataClassification name="parameter">
```

parameter

Represents one of the following parameters:

sensitive

vst_hide

ignore_on_copy

attributelevelencrypt

previouslyencrypted

For example, an attribute description that includes the vst_hide data classification attribute resembles the following:

```
<ImManagedObjectAttr physicalname="salary" displayname="Salary"
description="salary" valuetype="String" required="false" multivalued="false"
maxlength="0">
  <DataClassification name="vst_hide"/>
```

Attribute-Level Encryption

You can encrypt an attribute in the user store by specifying an AttributeLevelEncrypt data classification for that attribute in the directory configuration file (directory.xml). When attribute-level encryption is enabled, Identity Manager encrypts that attribute's value before storing it in the user store. The attribute is displayed as clear text in the User Console.

Note: [Managing Sensitive Attributes](#) (see page 68) describes methods for displaying sensitive data in the User Console.

The attribute is encrypted using RC2 encryption or FIPS 140-2 encryption, if FIPS 140-2 support is enabled.

Before implementing attribute-level encryption, note the following:

- If an attribute is encrypted, Identity Manager cannot find that attribute in a search.

Consider setting the attribute to `searchable="false"` in the `directory.xml` file—For example:

```
<ImManagedObjectAttr physicalname="title" description="Title"
displayname="Title" valuetype="String" maxlength="0"
searchable="false">
```

```
<DataClassification name="AttributeLevelEncrypt"/>
```

```
</ImManagedObjectAttr>
```

- In implementations where the Identity Manager uses a shared user store and Provisioning Directory, do not encrypt attributes that are used by the Provisioning Server.
- If you enable attribute-level encryption for a user store that is used by applications other than Identity Manager, the other applications will not be able to use the encrypted attribute.

How To Add Attribute-Level Encryption

When you add attribute-level encryption to an Identity Manager directory, Identity Manager automatically encrypts existing clear text attribute values when you save the object associated with the attribute. For example, if you encrypt the password attribute, Identity Manager encrypts the password when it saves a user's profile.

Note: To encrypt the attribute value, the task that you use to save the object must include the attribute. To encrypt the password attribute in the previous example, the password field must be added to the task you use to save the object, such as the Modify User task.

All new objects are created with encrypted values in the user store.

To add attribute-level encryption to an existing user store, you complete the following steps:

1. Complete one of the following:
 - Create a new Identity Manager directory
 - Update an existing directory by exporting the directory settings.
2. Add the data classification, `AttributeLevelEncrypt`, to the attribute that you want to encrypt in the `directory.xml` file.

For example:

```
<ImManagedObjectAttr physicalname="salary" displayname="Salary"
description="salary" valuetype="String" required="false" multivalued="false"
maxlength="0" searchable="false">
<DataClassification name="AttributeLevelEncrypt"/>
```

3. If you created a new Identity Manager Directory, associate the directory with an environment.
4. To force Identity Manager to encrypt all values immediately, modify all objects using the Bulk Loader.

Note: For more information about the Bulk Loader, see the *Administration Guide*.

How to Remove Attribute-Level Encryption

If you have an encrypted attribute in the Identity Manager Directory and then choose to store that attribute's values as clear text, you can remove the `AttributeLevelEncrypt` data classification.

Once the data classification has been removed, Identity Manager stops encrypting the new attribute values. Existing values are decrypted when you save the object associated with the attribute.

Note: To decrypt the attribute value, the task that you use to save the object must include the attribute. For example, to decrypt a password for an existing user, you save the user object with a task that includes the password field, such as the Modify User task.

To force Identity Manager to detect and decrypt any encrypted values that remain in the user store for the attribute, you can specify another data classification, `PreviouslyEncrypted`. The clear text value is saved to the user store when you save the object.

Note: Adding the PreviouslyEncrypted data classification adds extra processing on every object load. To prevent performance issues, consider adding the PreviouslyEncrypted data classification, loading and saving each object associated with that attribute, and then removing the data classification. This method automatically converts all stored encrypted values to stored clear text.

To remove attribute-level encryption from an existing user store, you complete the following steps:

1. Export the directory settings for the appropriate Identity Manager Directory.
2. In the directory.xml file, remove the data classification, AttributeLevelEncrypt, from attributes that you want to decrypt.
3. If you want to force Identity Manager to remove previously encrypted values, add the PreviouslyEncrypted data classification attribute.

For example:

```
<ImManagedObjectAttr physicalname="salary" displayname="Salary"
description="salary" valuetype="String" required="false" multivalued="false"
maxlength="0" searchable="false">
<DataClassification name="PreviouslyEncrypted"/>
```

4. To force Identity Manager to decrypt all values immediately, modify all objects using the Bulk Loader.

Note: For more information about the Bulk Loader, see the *Administration Guide*.

eTrust Directory Considerations

When you describe attributes for an eTrust Directory user store, note the following:

- Attribute names are case-sensitive.
- Using the seeAlso attribute as the attribute that indicates a self-subscribing group may cause errors when administrators create groups.
- Using the photo attribute as the attribute that indicates a user account's status (enabled or disabled) may cause errors when an administrator creates a user.

Note: For additional information about eTrust Directory requirements, see the eTrust Directory documentation.

Microsoft Active Directory Considerations

When you describe attributes for Active Directory, note the following:

- The case of the attributes specified in attribute descriptions must match the case of the attributes in Active Directory. For example, when you select the unicodePwd attribute as the attribute that stores user passwords, you must specify unicodePwd (with a capital P) in the directory configuration file.
- For user and group objects, you must include the sAMAccountName attribute.

IBM Directory Server Considerations

When you describe attributes for an IBM Directory Server user directory, see the following sections:

- [Groups in Directory Server Directories](#) (see page 74)
- [The Objectclass "Top" in the Organization Object Description](#) (see page 75)

Groups in Directory Server Directories

IBM Directory Server requires groups to contain at least one member. To address this requirement, Identity Manager adds a *dummy user* as a member of a new group when the group is created.

Configure a Dummy User

To configure a dummy user

1. In the Group Object section of the directory configuration file, locate the following elements:

```
<PropertyDict name="DUMMY_USER">
  <Property name="DUMMY_USER_DN">##DUMMY_USER_DN</Property>
</PropertyDict>
```

Note: If these elements do not exist in the directory configuration file, add them exactly as they appear here.

2. Replace ##DUMMY_USER_DN with a user DN. Identity Manager will add this DN as a member of all new groups.

Note: If you specify the DN of an existing user, that user will appear as a member of all groups created by Identity Manager. To prevent the *dummy user* from appearing as a group member, specify a DN that does not exist in the directory.

3. Save the directory configuration file.

The Objectclass Top in the Organization Object Description

Important! In the description of the organization object in directory configuration file, do not include the objectclass top.

For example, when the objectclass of the organization object is top, organizationalUnit, specify the objectclass as follows:

```
<ImManagedObject name="Organization" description="My Organizations"
objectclass="organizationalUnit" objecttype="ORG">
```

Including top may cause unpredictable search results.

Oracle Internet Directory Considerations

When you describe attributes for an Oracle Internet Directory (OID) user store, specify LDAP attributes using lowercase letters only.

Well-Known Attributes for an LDAP User Store

Well-known attributes have special meaning in Identity Manager. They are identified by the following syntax:

`%ATTRIBUTENAME%`

In this syntax, *ATTRIBUTENAME* must be uppercase.

A well-known attribute is mapped to one physical attribute, using an [attribute description](#) (see page 113).

In the following attribute description, the attribute userpassword is mapped to the well-known attribute %PASSWORD% so that Identity Manager will treat the value in userpassword as a password as follows:

```
<ImManagedObjectAttr
  physicalname="userpassword"
  displayname="Password"
  description="Password"
  valuetype="String"
  required="false"
  multivalued="false"
  wellknown="%PASSWORD%"
  maxlength="0" />
```

Some well-known attributes are required; others are optional.

User Well-Known Attributes

A list of user well-known attributes and the items to which they map follows:

%ADMIN_OF%

Maps to the list of groups for which the user is an administrator.

This well-known attribute may improve search performance at sites with many groups. When the %ADMIN_OF% well-known attribute is specified, Identity Manager looks for the groups that a user can manage in the %ADMIN_OF% attribute instead of checking every group in the user store.

%ADMIN_ROLE_CONSTRAINT%

Maps to the list of an administrator's admin roles.

The physical attribute mapped to %ADMIN_ROLE_CONSTRAINT% must be multivalued to accommodate multiple roles.

We recommend indexing the LDAP attribute that is mapped to %ADMIN_ROLE_CONSTRAINT%.

%CERTIFICATION_STATUS%

Maps to a user's certification status.

This attribute is required to use the user certification feature.

Note: For more information about user certification, see the *Administration Guide*.

%DELEGATORS%

Maps to a list of users who have delegated work items to the current user.

This attribute is required to use delegation. The physical attribute mapped to %DELEGATORS% must be multi-valued and capable of holding strings.

Important! Editing this field directly using Identity Manager tasks or an external tool can cause significant security implications.

%EMAIL%

Maps to a user's email address.

Required to use the email notification feature

%ENABLED_STATE%

(Required)

Maps to a user's status.

Note: This attribute must match the Disabled Flag user directory attribute in the SiteMinder user directory connection.

%FIRST_NAME%

Maps to a user's first name.

%FULL_NAME%

Maps to a user's first and last names.

%IDENTITY_POLICY%

Specifies the list of identity policies that have been applied to a user account.

Identity Manager uses this attribute to determine whether an identity policy should be applied to a user. When the policy has the Apply Once setting enabled, and the policy is listed in the %IDENTITY_POLICY% attribute, Identity Manager does not apply the changes in the policy to the user.

Note: For more information about identity policies, see the *Administration Guide*.

%LAST_CERTIFIED_DATE%

Maps to the date when a user's roles were certified.

Required to use the user certification feature.

Note: For more information about user certification, see the *Administration Guide*.

%LAST_NAME%

Maps to a user's last name.

%MEMBER_OF%

Maps to the list of groups of which the user is a member.

The physical attribute mapped to %MEMBER_OF% must be multivalued to accommodate multiple groups.

Using this attribute will improve response time when searching for a user's groups.

You can use this attribute with Active Directory or any directory schema that maintains a user's group membership on the user object.

%ORG_MEMBERSHIP%

(Required)

Maps to the DN of the organization to which the user belongs.

Identity Manager uses this well-known attribute to determine a [directory's structure](#) (see page 82).

This attribute is not required when the user directory does not include organizations.

% ORG_MEMBERSHIP_NAME %

(Required)

Maps to the user-friendly name of the organization in which the user's profile exists.

This attribute is not required when the user directory does not include organizations.

% PASSWORD %

Maps to a user's password.

Note: This attribute must match the Password Attribute in the SiteMinder user directory connection.

% PASSWORD_DATA %

(Required for password policy support)

Specifies the attribute that tracks password policy information.

% PASSWORD_HINT %

(Required)

Maps to a user-specified question and answer pair. The question and answer pair is used if users forget their passwords.

To support multiple question and answer pairs, the %PASSWORD_HINT% attribute must be multi-valued.

Note: If you are using SiteMinder's Password Services feature to manage passwords, the Password Hint attribute must match the Challenge/Response attribute in the SiteMinder user directory.

% USER_ID %

(Required)

Maps to a user's ID.

Group Well-Known Attributes

The following is a list of group well-known attributes:

%GROUP_ADMIN_GROUP%

Indicates which attribute stores a list of groups that are administrators of the group. For example, when group 1 is an administrator of group A, group 1 is stored in the %GROUP_ADMIN_GROUP% attribute.

Note: If you do not specify a %GROUP_ADMIN_GROUP% attribute, Identity Manager stores administrator groups in the %GROUP_ADMIN% attribute.

Note: To add a group as an administrator of another group, see the *Administration Guide*.

%GROUP_ADMIN%

Indicates which attribute contains the DNs of the group's administrators.

The physical attribute mapped to %GROUP_ADMIN% must be multivalued.

%GROUP_DESC%

Indicates which attribute contains a group's description.

%GROUP_MEMBERSHIP%

(Required)

Indicates which attribute contains a list of the group's members.

The physical attribute mapped to %GROUP_MEMBERSHIP% must be multivalued.

The %GROUP_MEMBERSHIP% well-known attribute is not required for Provisioning user directories.

%GROUP_NAME%

(Required)

Indicates which attribute stores a group name.

%ORG_MEMBERSHIP%

(Required)

Indicates which attribute contains the DN of the organization to which the group belongs.

Identity Manager uses this well-known attribute to determine a [directory's structure](#) (see page 82).

This attribute is not required when the user directory does not include organizations.

%ORG_MEMBERSHIP_NAME%

Indicates which attribute contains the user-friendly name of the organization in which the group exists.

This attribute is not valid for user directories that do not include organizations.

%SELF_SUBSCRIBING%

Indicates which attribute determines whether users can subscribe to a [group](#) (see page 81).

%NESTED_GROUP_MEMBERSHIP%

Indicates which attribute stores a list of groups that are members of the group. For example, when group 1 is a member of group A, group 1 is stored in the %NESTED_GROUP_MEMBERSHIP% attribute.

If you do not specify a %NESTED_GROUP_MEMBERSHIP% attribute, Identity Manager stores nested groups in the %GROUP_MEMBERSHIP% attribute.

To include groups as members of other groups, configure support for nested groups as described in Configuring Dynamic and Nested Groups for instructions.

%DYNAMIC_GROUP_MEMBERSHIP%

Indicates which attribute stores the LDAP query that generates a [dynamic group](#) (see page 130).

Note: To extend the available attributes for the Group object to include %NESTED_GROUP_MEMBERSHIP% and %DYNAMIC_GROUP_MEMBERSHIP% attributes, you can use auxiliary object classes.

Organization Well-Known Attributes

The following well-known attributes apply only to environments that support organizations:

%ORG_DESCR%

Indicates which attribute contains an organization's description.

%ORG_MEMBERSHIP%

(Required)

Indicates which attribute contains the DN of an organization's parent organization.

%ORG_MEMBERSHIP_NAME%

Indicates which attribute contains the user-friendly name of the parent organization of an organization.

%ORG_NAME%

(Required)

Indicates which attribute contains the name of the organization.

%Admin_Role_Constraint% Attribute

When you create an admin role, you specify one or more rules for role membership. Users who satisfy the membership rules are granted the role. For example, when the membership rule for the User Manager role is title=User Manager, users who have the title User Manager possess the User Manager role.

Note: For more information about rules, see the *Administration Guide*.

%ADMIN_ROLE_CONSTRAINT% enables you to designate a profile attribute to store an administrator's admin roles.

How to Use the %ADMIN_ROLE_CONSTRAINT% Attribute

To use %ADMIN_ROLE_CONSTRAINT% as the constraint for all admin roles, do the following:

- Pair the %ADMIN_ROLE_CONSTRAINT% well-known attribute with a multivalued profile attribute to accommodate multiple roles.
- When you configure an admin role in the User Console, make sure the constraint is as follows:

Admin Roles equals *role name*

role name

Defines the name of the role for which you are supplying the constraint, as in the following example:

Admin Roles equals User Manager

Note: Admin Roles is the default displayname for the %ADMIN_ROLE_CONSTRAINT% attribute.

Configure Well-Known Attributes

Perform the following procedure to configure well-known attributes.

To configure well-known attributes

1. In the directory configuration file, search for the following:

#

2. Replace the value that begins with ## with the appropriate LDAP attribute.
3. Repeat Steps 1 and 2 until you have replaced all required values.
4. Map optional well-known attributes to physical attributes, as necessary.
5. Save the directory configuration file.

Describe the User Directory Structure

Identity Manager uses the %ORG_MEMBERSHIP% well-known attribute to determine the structure of a user directory.

The procedure for describing the user directory structure depends on the type of directory structure.

How to Describe a Hierarchical Directory Structure

The directory configuration file is already configured for a hierarchical directory structure. As a result, you do not have to modify the %ORG_MEMBERSHIP% attribute description.

How to Describe a Flat User Directory Structure

To describe a user directory with a flat user structure, do the following:

1. Locate the %ORG_MEMBERSHIP% attribute description in the User Object section of the directory.xml file.
2. In the physicalname parameter, replace %ORG_MEMBERSHIP% with the name of the attribute that will store the organization to which the user belongs

How to Describe a Flat Directory Structure

To describe a flat directory structure, do the following:

1. Locate the %ORG_MEMBERSHIP% attribute description in the User Object section of the directory.xml file.
2. In the physicalname parameter, replace %ORG_MEMBERSHIP% with the name of the attribute that will store the organization to which the user belongs.

3. Repeat Step 1 in the Group Object section.
4. In the `physicalname` parameter, replace `%ORG_MEMBERSHIP%` with the name of the attribute that will store the organization to which the group belongs

How to Describe a User Directory that Does Not Support Organizations

Verify that no object descriptions or well-known attributes are defined for organizations in `directory.xml`.

How to Configure Groups

For configuration, groups can be divided as follows:

- Self-subscribing groups
- Dynamic and Nested groups

Configure Self-Subscribing Groups

You can enable self-service users to join groups by configuring support for self-subscribing groups in the directory configuration file.

When a user self-registers, Identity Manager looks for groups in specified organizations, and displays the self-subscribing groups to the user.

To configure a self-subscribing group

1. In the Self-subscribing Groups section, add a `SelfSubscribingGroups` element as follows:

```
<SelfSubscribingGroups type=search_type org=org_dn>
```

2. Add values for the following parameters:

type

Indicates where Identity Manager searches for self-subscribing groups as follows:

- **NONE**—Identity Manager does not search for groups. Specify **NONE** to prevent users from self-subscribing to groups.
- **ALL**—Identity Manager begins searching for groups at the root. Specify **ALL** when users can subscribe to groups throughout a hierarchical directory.

- **INDICATEDORG**—Identity Manager searches for self-subscribing groups in the user's organization and its suborganizations. For example, when a user's profile is in the Marketing organization, Identity Manager searches for self-subscribing groups in the Marketing organization, and in all suborganizations.
- **SPECIFICORG**—Identity Manager searches in a specific organization. Supply the distinguished name (DN) of the specific organization in the org parameter.

org

Specifies the unique identifier of the organization where Identity Manager searches for self-subscribing groups.

Note: You must specify the org parameter when type=SPECIFICORG.

Once support for self-subscribing groups is configured in the Identity Manager directory, Identity Manager administrators can specify which groups are self-subscribing in the User Console.

Note: For more information about managing groups, see the *Administration Guide*.

Configure Dynamic and Nested Groups

If you are managing an LDAP user store, you can configure support for the following types of groups in the directory configuration file:

Dynamic Groups

Enables you to dynamically define group membership by specifying an LDAP filter query in the User Console. With dynamic groups, administrators do not have to search for and add group members individually.

Nested Groups

Enables you to add groups as members of other groups.

You can enable dynamic and nested groups using the directory configuration file.

To configure a dynamic or nested group

1. Map the following [well-known attributes](#) (see page 79) to a physical attribute for the Group managed object as needed:
 - %DYNAMIC_GROUP_MEMBERSHIP%
 - %NESTED_GROUP_MEMBERSHIP%

2. In the Directory Groups Behavior section, add the following GroupTypes element:

```
<GroupTypes type=group>
```

3. Type a value for the following parameter:

group

Enables support for dynamic and nested groups. The valid values are as follows:

- NONE—Identity Manager does not support dynamic and nested groups.
- ALL—Identity Manager supports dynamic and nested groups.
- DYNAMIC—Identity Manager supports dynamic groups only.
- NESTED—Identity Manager supports nested groups only.

Once support for dynamic and nested groups is configured in the Identity Manager directory, Identity Manager administrators can specify which groups are dynamic and nested in the User Console.

Note: When you set the group type to NESTED or ALL *without* setting the %NESTED_GROUP_MEMBERSHIP% well-known parameter, Identity Manager stores both the nested groups and users in the %GROUP_MEMBERSHIP% well-known parameter. Processing group membership may be slightly slower.

Add Support for Groups as Administrators of Groups

If you are managing an LDAP user store, you can enable groups to serve as administrators of other groups. When you assign a group as an administrator, only administrators of that group will be administrators of the specified group. Members of the administrator group you specify will not have privileges to manage the group.

To configure support for groups as administrators of other groups

1. Map the %ADMIN_GROUP_ADMIN% well-known attribute to a physical attribute that will store the list of groups that serve as administrators.

[Group Well-Known Attributes](#) (see page 79) provides more information about the %ADMIN_GROUP_ADMIN% attribute.

Note: If you set the admin group type to ALL without setting the %ADMIN_GROUP_ADMIN% well known, Identity Manager stores administrator groups in the %GROUP_ADMIN% attribute.

2. In the Directory AdminGroups Behavior section, configure the AdminGroupTypes element as follows:

```
<AdminGroupTypes type="ALL">
```

Note: The default AdminGroupTypes is NONE.

Once support for groups as administrators is configured in the Identity Manager directory, Identity Manager administrators can specify groups as administrators of other groups in the User Console.

Validation Rules

A validation rule enforces requirements on data that a user types in a task screen field. The requirements can enforce a data type or format, or make sure that the data is valid in the context of other data on the task screen.

Validation rules are associated with profile attributes. Before processing a task, Identity Manager makes sure that the data entered for a profile attribute satisfies any associated validation rules.

You can define validation rules and associate them with profile attributes in the directory configuration file.

Additional Identity Manager Directory Properties

You can configure the following additional properties:

- Sort order for search results.
- Search across object classes to verify that a new user does not exist already.
- Wait time to avoid Identity Manager timing out before completion of the replication of data from the master LDAP directory to the slave LDAP directory.

Configure Sort Order

You can specify a sort attribute for each managed object, such as users, groups, or organizations. Identity Manager uses this attribute to sort search results in custom business logic, which you create with the Identity Manager APIs.

Note: The sort attribute does not affect the way search results are displayed in the User Console.

For example, when you specify the `cn` attribute for the user object, Identity Manager sorts the results of a search for users alphabetically by the `cn` attribute.

To configure default sort order

1. After the last `IMSManagedObjectAttr` element in the section for the managed object to which the sort order applies, add the following statements:

```
<PropertyDict name="SORT_ORDER">
  <Property name="ATTR">your_sort_attribute
</Property>
</PropertyDict>
```

2. Replace *your_sort_attribute* with the attribute on which Identity Manager will sort the search results.

Note: Specify only one physical attribute. Do not specify a well-known attribute.

For example, to sort user search results based on the value of the `cn` attribute, add the following elements after the last `IMSManagedObjectAttr` element in the User Object section of the directory configuration file:

```
<!-- ***** User Object ***** -->
<IMSManagedObject name="User" description="My Users"
  objectclass="top,person,organizationalperson,user"
  objecttype="USER">
.
.
.
  <IMSManagedObjectAttr physicalname="departmentnumber"
    displayname="Department" description="Department"
    valuetype="String" required="true"
    multivalued="false" maxlength="0" />
  <PropertyDict name="SORT_ORDER">
    <Property name="ATTR">cn</Property>
  </PropertyDict>
</IMSManagedObject>
```

Search across Objectclasses

When you create a user, Identity Manager searches the user store to verify that the user does not already exist. This search is limited to users who have the objectclasses specified in the user object definition in the directory configuration file (directory.xml). If no existing user is found in those objectclasses, Identity Manager tries to create the user.

When a user exists that has the same unique identifier (user ID) but a different objectclass, the LDAP server fails to create the user. The error is reported in the LDAP server, but Identity Manager does not recognize the error. Identity Manager appears to create the user successfully.

To prevent this issue, you can configure a SEARCH_ACROSS_CLASSES property that causes Identity Manager to search users across all objectclass definitions when checking for existing users.

Note: This property affects only searches for duplicate users when performing tasks such as creating a user. For all other searches, objectclass constraints apply.

To search across objectclasses

1. In the directory configuration file (directory.xml), locate the `ImsManagedObject` element that describes the user object.
2. Add the following `PropertyDict` element:

```
<PropertyDict name="SEARCH_ACROSS_CLASSES" description="allowing checking an
attribute across classes ">
  <Property name="ENABLE">true</Property>
</PropertyDict>
```

Note: The `PropertyDict` element must be the last element in the `ImsManagedObject` element, as in the following example:

```
<ImsManagedObject name="User" description="My Users"
objectclass="top,person,organizationalperson,inetorgperson,customClass"
objecttype="USER">
  <ImsManagedObjectAttr physicalname="departmentnumber"
displayname="Department" description="Department" valuetype="String"
required="true" multivalued="false" maxlength="0" />
  .
  .
  .
  <PropertyDict name="SEARCH_ACROSS_CLASSES" description="allow checking an
attribute across classes ">
    <Property name="ENABLE">true</Property>
  </PropertyDict>
```


Specify Replication Wait Time

In a deployment that includes replication between master and slave LDAP directories, you can configure the Policy Server to communicate with a slave directory. In this configuration, the Policy Server automatically detects referrals that point to the master directory during operations that write data to the LDAP directory. The data is stored in the master LDAP directory and replicated to the slave LDAP directory according to the replication scheme of your network resources.

In this configuration, when you create an object in Identity Manager, the object is created in the master directory and replicated to the slave directory. A delay may occur during the replication process that causes the create action to fail in Identity Manager.

To prevent this issue from occurring, you can specify the amount of time (in milliseconds) that Identity Manager waits before "timing out" in the `REPLICATION_WAIT_TIME` property.

Note: On Solaris systems, the replication wait time is processed in seconds not milliseconds, although it is specified as milliseconds. For example, when you specify the `REPLICATION_WAIT_TIME` as 800 milliseconds, which is less than a second, then IdentityMinder waits 0 seconds. When the `REPLICATION_WAIT_TIME` is 1600 milliseconds, Identity Manager waits 1 second.

To specify replication wait time

1. In the directory configuration file (`directory.xml`), locate the `ImsManagedObject` element that describes the user object.
2. Add the following `PropertyDict` element:

```
<PropertyDict name="REPLICATION_WAIT_TIME" description="time delay in
milliseconds for ldap prodvider to allow replication to propagate from master
to slave">
  <Property name=REPLICATION_WAIT_TIME"><time in milliseconds></Property>
</PropertyDict>
```

Note: The `PropertyDict` element must be the last element in the `ImsManagedObject` element, as in the following example:

```
<ImsManagedObject name="User" description="My Users"
objectclass="top,person,organizationalperson,inetorgperson,customClass"
objecttype="USER">
  <ImsManagedObjectAttr physicalname="departmentnumber"
displayname="Department" description="Department" valuetype="String"
required="true" multivalued="false" maxlength="0" />
  .
  .
  .
```

```
<PropertyDict name="REPLICATION_WAIT_TIME" description="time delay in  
milliseconds for ldap provider to allow replication to propagate from master  
to slave">  
<Property name="REPLICATION_WAIT_TIME">800</Property>  
</PropertyDict>
```

When the replication wait time is not defined, the default value 0 is used.

Specify LDAP Connection Settings

To improve performance, you can specify the following parameters in the directory configuration file (directory.xml):

Connection Timeout

Specifies the maximum number of seconds that Identity Manager searches a directory before terminating the search.

This property is specified in the directory configuration file as follows:

```
com.sun.jndi.ldap.connect.timeout
```

Connection Pool Max Size

Specifies the maximum number of connections that Identity Manager can make to the LDAP directory.

This property is specified in the directory configuration file as follows:

```
com.sun.jndi.ldap.connect.pool.maxsize
```

Connection Pool Default Size

Specifies the default number of connections between Identity Manager and the LDAP directory.

This property is specified in the directory configuration file as follows:

```
com.sun.jndi.ldap.connect.pool.prefsiz
```

To specify LDAP connection settings

1. In the directory configuration file (directory.xml), locate the `ImsManagedObject` element that describes the user object.
2. Add the following `PropertyDict` element:

```
<PropertyDict name="LDAP_CONNECTION_SETTINGS" description="LDAP Connection
Settings">
  <Property name="com.sun.jndi.ldap.connect.timeout">5000</Property>
  <Property name="com.sun.jndi.ldap.connect.pool.maxsize">200</Property>
  <Property name="com.sun.jndi.ldap.connect.pool.prefsize">10</Property>
</PropertyDict>
```

3. Save the directory.xml file.

Identity Manager configures these settings when you create the Identity Manager directory with this file.

How to Improve Directory Search Performance

To improve the performance of directory searches for users, organizations, and groups, do the following:

- Index the attributes that administrators can specify in search queries.
Note: For Oracle Internet Directory, a search may fail when an attribute in a search query is not indexed.
- Override the default directory timeout setting by specifying values for the following search parameters in a directory configuration file (directory.xml):
 - maxpagesize—Maximum page size for search results.
 - maxrows—Maximum number of rows returned after a search.
 - timeout—Maximum number of seconds that Identity Manager searches a directory before terminating the search.
- Create multiple SiteMinder User Directory Connections and configure them for failover. Be sure that each user directory connection points to the same user directory and port.

Configuring multiple SiteMinder User Directory connections for failover provides additional connections to the user directory. When one connection is busy, there are other connections available for requests.

Note: For more information on configuring user directories for Identity Manager, see the *Installation Guide*.

- Tune the user directory. See the documentation for the user directory that you are using.

Chapter 5: Creating an Identity Manager Directory for a Relational Database

This section contains the following topics:

[Identity Manager Directories](#) (see page 93)

[Important Notes When Configuring Identity Manager for Relational Databases](#) (see page 95)

[Create an Oracle Datasource for WebSphere](#) (see page 96)

[How to Create an Identity Manager Directory](#) (see page 97)

[How To Create a JDBC Datasource](#) (see page 97)

[How to Create an ODBC Data Source For Use with SiteMinder](#) (see page 105)

[How to Describe a Database in a Directory Configuration File](#) (see page 105)

[Connection to the User Directory](#) (see page 119)

[Well-Known Attributes for a Relational Database](#) (see page 125)

[How to Configure Self-Subscribing Groups](#) (see page 130)

[Validation Rules](#) (see page 131)

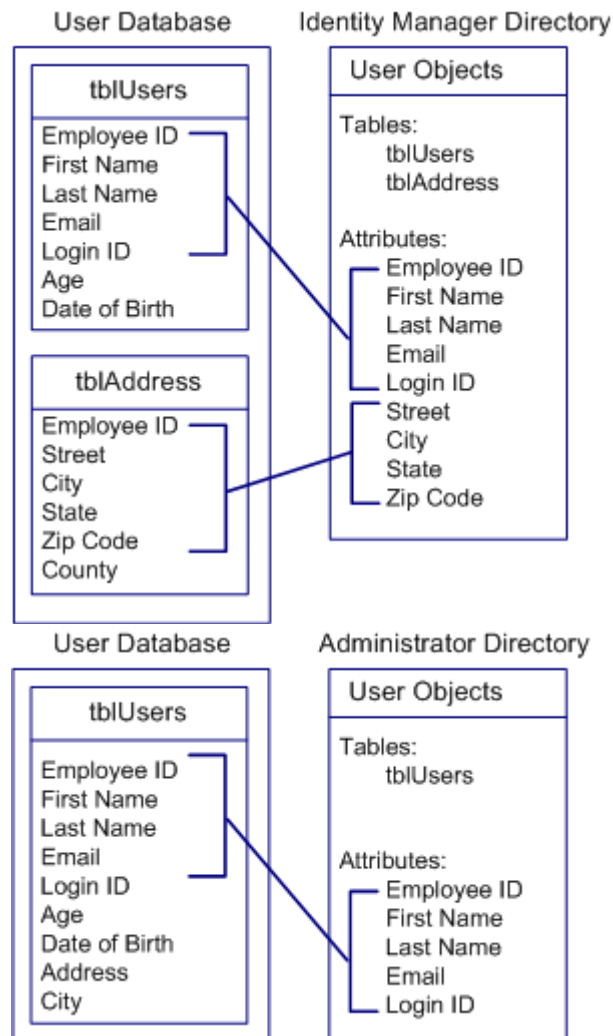
[Organization Management](#) (see page 131)

[How to Improve Directory Search Performance](#) (see page 134)

Identity Manager Directories

An *Identity Manager directory* describes how objects, such as users, groups, and (optionally) organizations are stored in the user store and represented in Identity Manager. An Identity Manager directory is associated with one or more Identity Manager environments.

The following illustration shows how an Identity Manager directory relates to a user store:



Note: Some user attributes in the database are not part of the Identity Manager directory, and therefore not managed by Identity Manager.

Important Notes When Configuring Identity Manager for Relational Databases

Before you configure Identity Manager to manage a relational database, ensure that the database meets the following requirements:

- The database must be accessible through a JDBC driver or an Open Database Connectivity (ODBC) driver (when Identity Manager integrates with SiteMinder). The driver should support outer joins. If more than two tables are used to represent a managed object, the driver should also support nested outer joins.

Note: If the driver does not support outer joins, Identity Manager uses inner joins when querying the database. This may cause unexpected query results.

- You must be able to uniquely identify each object that Identity Manager manages, such as a user, group, or organization (when supported). For example, the unique identifier for users may be a login ID.

Note: The unique identifier must be stored in a single column.

- Identity Manager requires some multivalued attributes, which can be stored as a delimited list in a single cell or in multiple rows in a separate table. For example, the following tblGroupMembers table stores the members of a group:

ID	Members
Research	dmason
Research	rsavory
Marketing	dmason
Marketing	awelch

The ID column contains the unique identifier for a group and the Members column contains the unique identifier for a member of the group. For example, dmason and rsavory are members of the Research group. When a new member is added to that group, another row is added to tblGroupMembers.

- When your environment includes organizations, do the following:
 - Edit and run a SQL script, included with Identity Manager, against the database to [configure organization support](#) (see page 132).
 - Identity Manager requires a top level organization, named the root. All other organizations relate to the root organization.
- For more information about organization requirements, see [Organizations Management](#) (see page 131).

Create an Oracle Datasource for WebSphere

1. In the WebSphere Administrative Console, navigate to the JDBC provider that you created when you configured the JDBC driver.
2. Create a new data source with the following properties and click Apply:
 - Name:** User Store Data Source
 - JNDI Name:** userstore
 - URL:** jdbc:oracle:thin:@db_systemname:1521:oracle_sid
3. Configure a new J2C Authentication Data Entry for the User Store Data Source:
 - a. Enter the following properties:
 - Alias:** User Store
 - User ID:** *username*
 - password:** *password*

where *username* and *password* are the username and password for the account you specified when you created the database.
 - b. Click OK, then use the navigation links at the top of the screen to return to the data source you are creating.
4. Select the User Store J2C Authentication Data Entry you created from the list box in the following fields:
 - Component-managed Authentication Alias
 - Container-managed Authentication Alias
5. Click OK, then save the configuration.

Note: To verify that the data source is configured correctly, click Test Connection in the configuration screen for the data source. If the test connection fails, restart WebSphere and test the connection again.

How to Create an Identity Manager Directory

The steps to configure Identity Manager to manage your directory are as follows:

1. If you are using SiteMinder, ensure that you have applied the policy store schema prior to creating an Identity Manager Directory.

Note: For more information on specific policy store schemas and how to apply them, see the *Installation Guide*.

2. Create a data source for the user database that Identity Manager will manage.
3. Describe the database to Identity Manager by modifying a directory configuration file (directory.xml). For more information, see How to Describe a Database in a Directory Configuration File.
4. In the Management Console, import the directory configuration file and create the directory.

How To Create a JDBC Datasource

Identity Manager requires a JDBC datasource in the application server where Identity Manager is installed to connect to the user store. The instructions for creating a datasource are different for each application server.

To create a JDBC datasource

1. Create a copy of the following file:

`jboss_home\server\default\deploy\objectstore-ds.xml`

jboss home

The installed location of the Jboss application server where Identity Manager is installed

The new file should exist in the same location.

2. Rename the file to userstore-ds.xml.
3. Edit userstore-ds.xml as follows:
 - a. Locate the <jndi-name> element.
 - b. Change the value of the <jndi-name> element from jdbc/objectstore to userstore as follows:

`<jndi-name>userstore</jndi-name>`

- c. In the <connection-url> element, change the DatabaseName parameter to the name of the database that serves as the user store as follows:

<connection-url>

jdbc:sqlserver://*ipaddress*:*port*;selectMethod=cursor;DatabaseName=*userstore_name*

</connection-url>

ipaddress

Specifies the IP address of the machine where the user store is installed

port

Specifies the port number for the database

userstore_name

Specifies the name of the database that serves as the user store

- d. Change the value of the <user-name> element to the username for an account that has read and write access to the user store.
- e. Change the value of the <password> element to the password for the account specified in the <user-name> element.

4. Save the file.

Create a JDBC Datasource for JBoss Application Servers

To create a JDBC datasource for JBoss

1. Create a copy of the following file:

jboss_home\server\default\deploy\objectstore-ds.xml

jboss_home

The installed location of the Jboss application server where Identity Manager is installed

The new file should exist in the same location.

2. Rename the file to userstore-ds.xml.
3. Edit userstore-ds.xml as follows:
 - a. Locate the <jndi-name> element.
 - b. Change the value of the <jndi-name> element from jdbc/objectstore to userstore as follows:

<jndi-name>userstore</jndi-name>

- c. In the <connection-url> element, change the DatabaseName parameter to the name of the database that serves as the user store as follows:

<connection-url>

jdbc:sqlserver://*ipaddress:port*;selectMethod=cursor;DatabaseName=*userstore_name*

</connection-url>

ipaddress

Specifies the IP address of the machine where the user store is installed

port

Specifies the port number for the database

userstore_name

Specifies the name of the database that serves as the user store

- d. Change the value of the <user-name> element to the username for an account that has read and write access to the user store.
- e. Change the value of the <password> element to the password for the account specified in the <user-name> element.

4. Save the file.

WebLogic Data Sources

The following sections describe how to create a JDBC datasource for WebLogic 8.1 and 9.2.

Create a JDBC Datasource for WebLogic

For WebLogic 8.1 application servers, you create connection pools and a datasource when you configure a connection to the user store.

Note: For complete information about WebLogic connection pools, refer to your WebLogic documentation at:
<http://edocs.beasys.com/wls/docs81/ConsoleHelp/jdbc.html>

To create the connection pool

1. Create the connection pools in the WebLogic Console:

- For MS SQL Server databases, use the following values:

Name: User Store Pool

Database Name: *userstore_name*

Host Name: *db_systemName*

Port: 1433

Database User Name: *username*

URL: jdbc:microsoft:sqlserver://*db_systemName*:1433

Driver Class Name: com.microsoft.jdbc.sqlserver.SQLServerDriver

Properties: *user=username*

databaseName=userstore_name

selectMethod=cursor

Password: *password*

- For SQL Server 2005 databases, use the following values:

Name: User Store Pool

Database Name: *userstore_name*

Host Name: *db_systemName*

Port: 1433

Database User Name: *username*

URL: jdbc:sqlserver://*db_systemName*:1433

Driver Class Name: com.microsoft.sqlserver.jdbc.SQLServerDriver

Properties: *user=username*

databaseName=userstore_name

selectMethod=cursor

Password: *password*

- For Oracle databases, use the following values:

Name: User Store Pool

Database Name: *userstore_name*

Host Name: *db_systemname*

Port: 1521

Database User Name: *username*

URL: jdbc:oracle:thin:@*db_systemname*:1521:*oracle_SID*

Driver Class Name: oracle.jdbc.driver.OracleDriver

Properties: user=*username*

Password: *password*

Note: After configuration, deploy the pool. For WebLogic 8.1, select the Targets and Deploy tab and ensure that the server instance *wl_server_name* is selected. After you deploy the pools, ensure there are no errors displayed in the console.

2. Configure a new JDBC Data Source with the following parameters:

Name: User Store Data Source

JNDI Name: userstore

Pool Name: User Store Pool

3. After configuration, deploy the pool by selecting the Targets and Deploy tab and select the server instance *wl_server_name*.

After you deploy the pool, check the console to see if any errors occurred.

Note: You may see an error that says the datasource cannot be created with a non-existent pool. To resolve this error, restart WebLogic.

Create a JDBC Datasource for WebLogic 9.2

You create a datasource in the WebLogic Administration Console.

Note: See the BEA documentation web site for complete information about [WebLogic connection pools](#).

To create the data source

1. Create a JDBC Data Source with the following parameters in the WebLogic Administration Console:

Name: User Store Data Source

JNDI Name: userstore

2. Create the connection pool for the data source with the following information:

- For MS SQL Server databases, use the following values:

URL: jdbc:microsoft:sqlserver://db_systemName:1433

Driver Class Name: com.microsoft.jdbc.sqlserver.SQLServerDriver

Properties: user=*username*

databaseName=*user store name*

selectMethod=cursor

Password: *password*

- For SQL Server 2005 databases, use the following values:

URL: jdbc:sqlserver://db_systemName:1433

Driver Class Name: com.microsoft.sqlserver.jdbc.SQLServerDriver

Properties: user=*username*

databaseName=*user store name*

selectMethod=cursor

Password: *password*

- For Oracle databases, use the following values:

URL: jdbc:oracle:thin:@tp_db_systemname:1521:oracle_SID

Driver Class Name: oracle.jdbc.driver.OracleDriver

Properties: user=*username*

Password: *password*

3. After configuration, set the target for the pool to the server instance *wl_server_name*.

After you deploy the pool, check the console to see if any errors occurred.

Note: You may see an error that says the datasource cannot be created with a non-existent pool. To resolve this error, restart WebLogic.

WebSphere Datasources

The following sections describe how to create a SQL or Oracle datasource for Websphere application servers.

Create a SQL Server Datasource for WebSphere 6.0

1. In the WebSphere Administrative Console, navigate to the JDBC provider that you created when you configured the JDBC driver.
2. Select Data Sources in the Additional Properties section.
3. Create a new data source with the following properties and click Apply:
Name: User Store Data Source
JNDI Name: userstore
databaseName: *userstore_name*
serverName: *db_systemname*
4. Configure the selectMethod property as follows:
 - a. Select Custom Properties in the Additional Properties section.
 - b. Click the selectMethod custom property.
 - c. Enter the following in the Value field:
cursor
 - d. Click OK, then use the navigation links at the top of the screen to return to the data source you are creating.

5. Configure a new J2C Authentication Data Entry for the User Store Data Source:
 - a. Select J2EE Connector Architecture (J2C) authentication data entries from the Related Items section.
 - b. Click New.
 - c. Enter the following properties:
Alias: User Store
User ID: *username*
password: *password*
where *username* and *password* are the username and password for the account you specified when you created the database.
 - d. Click OK, then use the navigation links at the top of the screen to return to the data source you are creating.
6. Select the User Store J2C Authentication Data Entry you created from the list box in the Component-managed Authentication Alias field.
7. Click OK, then save the configuration.
Note: To verify that the data source is configured correctly, click Test Connection in the configuration screen for the data source. If the test connection fails, restart WebSphere and test the connection again.

Create an Oracle Datasource for WebSphere

1. In the WebSphere Administrative Console, navigate to the JDBC provider that you created when you configured the JDBC driver.
2. Create a new data source with the following properties and click Apply:
Name: User Store Data Source
JNDI Name: userstore
URL: jdbc:oracle:thin:@*db_systemname*:1521:*oracle_sid*
3. Configure a new J2C Authentication Data Entry for the User Store Data Source:
 - a. Enter the following properties:
Alias: User Store
User ID: *username*
password: *password*
where *username* and *password* are the username and password for the account you specified when you created the database.
 - b. Click OK, then use the navigation links at the top of the screen to return to the data source you are creating.

4. Select the User Store J2C Authentication Data Entry you created from the list box in the following fields:
 - Component-managed Authentication Alias
 - Container-managed Authentication Alias
5. Click OK, then save the configuration.

Note: To verify that the data source is configured correctly, click Test Connection in the configuration screen for the data source. If the test connection fails, restart WebSphere and test the connection again.

How to Create an ODBC Data Source For Use with SiteMinder

If Identity Manager integrates with SiteMinder, define an ODBC data source on the SiteMinder machine that points to the database. Note the name of the data source for later use. Proceed as follows:

- **Windows:** Configure the ODBC data source as a System DN. See your Windows operating system documentation for instructions.
- **UNIX:** Add an entry specifying the parameters for the ODBC data source in the `system_odbc.ini` file, located in `policy_server_installation/db`.

How to Describe a Database in a Directory Configuration File

To manage a database, Identity Manager must understand the database structure and content. To describe the database to Identity Manager, create a directory configuration file (`directory.xml`).

The directory configuration file contains one or more of the following sections:

Identity Manager Directory Information

Contains information about the Identity Manager directory that is used by Identity Manager.

Attribute Validation

Defines the validation rules that apply to the Identity Manager directory.

Provider Information

Describes the user store that Identity Manager will manage.

Directory Search Information

Enables you to specify how Identity Manager searches the user store.

User Object (see page 108)

Describes how users are stored in the user store and represented in Identity Manager.

Group Object (see page 108)

Describes how groups are stored in the user store and represented in Identity Manager.

Organization Object (see page 108)

When the user store includes organizations, describes how organizations are stored and represented in Identity Manager.

Self-Subscribing Groups

Configures support for groups that self-service users can join.

The directory where you installed the administrative tools for Identity Manager includes the following directory configuration file template for relational databases:

`tools_dir\directoryTemplates\RelationalDatabase\directory.xml`

tools_dir

Defines the installed location of Identity Manager administrative tools, as in the following examples:

- **Windows:** C:\Program Files\CA\IAM Suite\Identity Manager\tools\directoryTemplates\RelationalDatabase\directory.xml
- **UNIX:** *HOME*/CA/IAM Suite/Identity Manager/tools/directoryTemplates/RelationalDatabase/directory.xml

Note: The directory configuration file template in `directoryTemplates\RelationalDatabase` is configured for environments that support organizations. To see a directory configuration file for an environment that does not include organizations, you can look at the `directory.xml` file for the NeteAuto sample, which is located in `tools_dir\samples\NeteAutoRDB\NoOrganization`

Copy the configuration template to a new directory or save it with a different name to prevent overwriting it. You can then modify the template to reflect your database structure.

The directory configuration file has two important conventions:

- ##—Indicates required values.

To provide all of the required information, locate all double pound signs (##) and replace them with appropriate values. For example, ##PASSWORD_HINT indicates that you must supply an attribute to store a question that a user answers to receive a temporary password in the case of a forgotten password.

- @—Indicates values that Identity Manager populates. Do not modify these values in the directory configuration file. Identity Manager prompts you to supply the values when you import the directory configuration file.

Before you modify the directory configuration file, you need the following information:

- Table names for the user, group, and organization objects (when your structure includes organizations)
- A list of attributes in user, group, and organization profiles (when your structure includes organizations)

Modify the Directory Configuration File

Perform the following procedure to modify the directory configuration file.

To modify the directory configuration file

1. Configure a connection to the database.
2. Specify the amount of time that Identity Manager searches a directory before terminating the search.
3. Define the user and group managed [objects that Identity Manager will manage](#) (see page 108).
4. Modify well-known attributes.
Well-known attributes identify special attributes, such as the password attribute, in Identity Manager.
5. Configure support for self-subscribing groups.
6. If your environment includes organizations, configure organization support.

More information:

[Managed Object Descriptions](#) (see page 108)

[Well-Known Attributes for an LDAP User Store](#) (see page 75)

[How to Configure Self-Subscribing Groups](#) (see page 130)

[Organization Management](#) (see page 131)

Managed Object Descriptions

In Identity Manager, you manage the following types of objects, corresponding to entries in a user store:

- Users—Represent users in an enterprise.
- Groups—Represent associations of users who have something in common.
- (Optional) Organizations—Represent business units. Organizations may contain users, groups, and other organizations.

Note: [Organizations Management](#) (see page 131) provides information about configuring organizations.

An object description contains the following:

- [Information about the object](#) (see page 108), such as the tables in which the object is stored.
- [The attributes that store information about an entry](#) (see page 113). For example, the pager attribute stores a pager number.

Important! An Identity Manager environment supports only one type of user, group, and organization object.

How to Describe a Managed Object

A managed object is described by specifying object information in the User Object, Group Object, and Organization Object sections (when the database includes organizations) of the directory configuration file.

Each of these sections contains an `ImsManagedObject` element, such as the following:

```
<ImsManagedObject name="User" description="My Users">
```

The `ImsManagedObject` element may include the following elements:

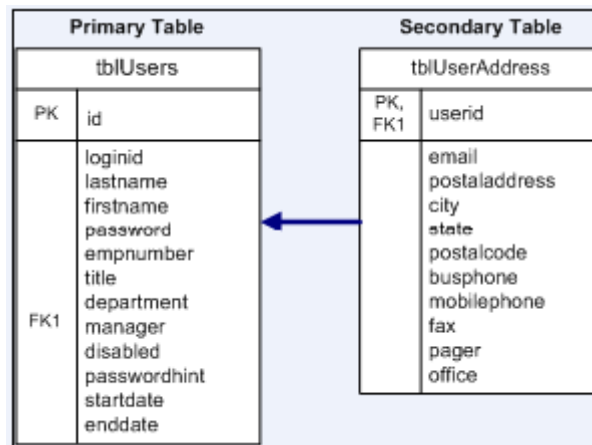
- Table (required)
- UniqueIdentifier (required)
- `ImsManagedObjectAttr` (required)
- `RootOrg` (for organization objects only)

Database Tables

Use the Table element in the directory configuration file to define the tables that store information about a managed object.

Each managed object must have one primary table that contains the unique identifier for the object. Additional information may be stored in secondary tables.

The following illustration shows a database that stores user information in a primary and secondary table:



If an object's information is stored in multiple tables, create a Table element for each table. Use the Reference element in the Table element for a secondary table to define its relationship to the primary table.

For example, if basic information about a user is stored in tblUsers and address information is stored in tblUserAddress, the table definitions for the User managed object would resemble the following entries:

```
<Table name="tblUsers" primary="true" />
<Table name="tblUserAddress">
  <Reference childcol="userid" primarycol="id" />
</Table name>
```

Table Elements

The parameters for a table element are as follows:

name

(Required)

Specifies the name of the table that stores some or all of the attributes in a managed object's profile.

primary

Indicates whether the table is the primary table for the managed object. The primary table contains the unique identifier for the object, as follows:

- True—The table is the primary table.
- False—The table is a secondary table (default).

If you do not specify the primary parameter, Identity Manager assumes that the table is a secondary table.

Note: Only one table can be the primary table.

filter

Identifies a subset of the table entries that apply to the managed object if the table stores information for more than one object type.

The filter parameter may resemble the following:

```
filter="type='USER'"
```

Note: The filter applies only to queries that Identity Manager generates. If you overwrite a generated query with a custom query, you must specify the filter in the custom query.

fullouterjoin

Indicates whether the outer join is a full outer join.

- True— The outer join is a full outer join. In this case, the condition required to return a valid row must be found in both tables in the join for a row to be returned.
- False—The outer join is a left outer join relative to the primary table. In this case, only the rows in one table in the query need to satisfy the condition (default).

Note: The parameters are optional unless otherwise specified.

The Table parameter can contain one or more Reference elements to link a primary table to secondary tables.

Reference Element

The parameters in the Reference element are as follows:

childcol

Indicates the column in the secondary table (specified in the corresponding Table element) that maps to the column in the primary table.

primarycol

Indicates the column in the primary table that maps to the column in the secondary table.

Note: The parameters are optional unless otherwise specified.

Specify Object Information

Object information is specified by supplying values for various parameters.

To specify object information

1. Locate the `ImsManagedObject` element in the User Object, Group Object, or Organization Object section.
2. Supply values for the following parameters:

name

(Required)

Provides a unique name for the managed object.

description

Provides the description of the managed object.

objecttype

(Required)

Specifies the type of managed object. The valid values are as follows:

- USER
- GROUP
- ORGANIZATION

The `ImsManagedObject` element should resemble the following:

```
<ImsManagedObject name="User" description="My Users" objecttype="USER">
```

3. Supply Table information, as described in [Database Tables](#) (see page 109).
4. Specify the column that contains the [unique identifier for the object](#) (see page 118).
5. Describe the [attributes that constitute the object's profile](#) (see page 113).
6. If you are configuring an organization object, go to [Organizations Management](#) (see page 131).

Custom Operations

You can define custom operations for certain managed objects to do the following:

- Use stored procedures
- Optimize queries for their database structure
- Retrieve a database-generated unique identifier

Custom operations apply only to attributes.

When specifying custom operations, remember the following:

- Users who specify custom operations must be familiar with SQL.
- Identity Manager does not validate custom operations. Syntax errors and invalid queries are not reported until runtime.
- If you specify a custom operation for an attribute, you cannot use that attribute in search filters in Identity Manager tasks.
- Custom operations must conform to XML standards. Represent special characters using XML syntax. For example, specify a single quotation mark (') as '

To specify a custom operation, use the Operation element.

Operation Element

The Operation element defines a SQL statement that executes a custom query or calls a stored procedure to create, retrieve, modify, or delete an attribute. It is a subelement of the IMSManagedObjectAttr element, as shown in the following example:

```
<ImManagedObjectAttr physicalname="tblUsers.id" displayname="User Internal ID"
description="User Internal ID" valuetype="Number" required="false"
multivalued="false" maxlength="0" hidden="false" permission="READONLY">
  <Operation name="GetDb" value="select @@identity" />
```

Operation element parameters are as follows:

name

Specifies a pre-defined name for an operation. The valid operations are as follows:

- Create
- Get
- Set

- Delete
- GetDB

The GetDB operation retrieves a unique identifier from the database during a Create task, when the unique identifier is generated by the database or from a stored procedure.

value

Defines the SQL statement or stored procedure to execute. The valid values are as follows:

- INSERT
- SELECT
- UPDATE
- DELETE
- CALL (for stored procedures)

Note: The parameters are optional unless otherwise specified.

The Operation element can contain one or more Parameter elements.

How to Modify Attribute Descriptions

An attribute stores information about a user, group, or organization entity, such as a telephone number or address. An entity's attributes determine its profile.

In the directory configuration file, attributes are described in `ImsManagedObjectAttr` elements. In the User Object, Group Object and Organization Object sections of the directory configuration file, you can do the following:

- Modify default attribute descriptions to describe your database attributes.
- Create new attribute descriptions by copying an existing description and modifying values as needed.

There is one `ImsManagedObjectAttr` element for each attribute in user, group, and organization profiles. For example, an `ImsManagedObjectAttr` element may describe a user ID.

An `ImsManagedObjectAttr` element resembles the following:

```
<ImsManagedObjectAttr
  physicalname="tblUsers.id"
  displayname="User Internal ID"
  description="User Internal ID"
  valuetype="Number"
  required="false"
  multivalued="false"
  maxlength="0"
  hidden="false"
  permission="READONLY">
```

Note: When you are using an Oracle database, note the following when you configure managed object attributes:

- Oracle databases are case-sensitive by default. The case of the attributes and table names in the directory configuration file must match the case of the attributes in Oracle.

Be sure to specify a maximum length for String datatypes to prevent truncation. To limit the length of strings, you can create a [validation rule](#) (see page 205) that displays an error when a user types a string that exceeds the maximum length.

The `ImsManagedObjectAttr` parameters are as follows.

Note: The parameters are optional unless otherwise specified.

physicalname

(Required)

Specifies the physical name of the attribute, and it must contain one of the following:

- The name and location where the value is stored.

Format: *tablename.columnname*

For example, when an attribute is stored in the `id` column in the `tblUsers` table, the physical name for that attribute is as follows:

`tblUsers.id`

You must define each table that contains an attribute in a [Table element](#) (see page 109).

- A well-known attribute.

A well-known attribute can represent a computed value. For example, you can use a well-known attribute to refer to an attribute that is computed by a [custom operation](#) (see page 112).

displayname

(Required)

Specifies a unique name for the attribute.

In the User Console, the display name appears in the list of attributes that are available to add to a task screen.

Note: Do not modify an attribute's displayname in the directory configuration file (directory.xml). To change the name of the attribute on a task screen, you can specify a label for the attribute in the task screen definition. For more information, see the *Administration Guide*.

description

Provides the description of the attribute.

valuetype

Specifies the attribute's data type. It can be one of the following types:

- String
- Integer
- Number
- Date

When an attribute's valuetype is incorrect, Identity Manager queries may fail.

To make sure that an attribute is stored correctly in the database, you can associate it with a [validation rule](#) (see page 205).

required

Indicates whether a value must be specified for the attribute, as follows:

- True—Required
- False—Optional (default)

multi-valued

Indicates whether the attribute can have multiple values, as follows:

- True— An attribute can have multiple values.
- False— An attribute can have only a single value (default).

For example, the group membership attribute in a user profile is multi-valued to store the groups to which a user belongs.

To store multi-valued attributes in a delimited list instead of in a multi-row table, you must define the delimiter character in the delimiter parameter.

Make sure that the number of possible values and the length of each value that the column enables are sufficient.

Important! The Group Membership attribute in the User object definition must be multi-valued.

wellknown

Provides the name of the well-known attribute.

Well-known attributes have a specific meaning in Identity Manager.

Format: %*ATTRIBUTENAME*%

Note: When a custom operation is associated with an attribute, you must specify a [well-known attribute](#) (see page 75).

maxlength

Determines the maximum size of the column.

permission

Indicates whether an attribute's value can be modified in a task screen, as follows:

- READONLY—The value is displayed but cannot be modified
- WRITEONCE—The value cannot be modified once the object is created. For example, a user ID cannot be changed after the user is created
- READWRITE—The value can be modified (default)

hidden

Indicates whether an attribute appears in the Identity Manager task screens, as follows:

- True—The attribute is not displayed to users.
- False—The attribute is displayed to users (default).

Logical attributes use hidden attributes.

Note: For more information about logical attributes, see the *Programming Guide for Java*.

system

Indicates attributes that are used by Identity Manager only, and should not be modified by users in the User Console, as follows:

- True—Users cannot modify the attribute. The attribute will not appear in the User Console.
- False—Users can modify this attribute. It is available to add to task screens in the User Console (default).

validationruleset

[Associates a validation rule set with the attribute](#) (see page 205).

The validation rule set that you specify must be defined in a ValidationRuleSet element in the directory configuration file.

delimiter

Defines the character that separates values when multiple values are stored in a single column.

Important! The multivalued parameter must be set to true for the delimiter parameter to apply.

Note: To prevent displaying sensitive information, such as passwords or salaries, in the User Console, you can specify DataClassification parameters.

Parameter Element

A Parameter element specifies values that are passed to the query. When multiple Parameter elements are defined, the values are passed to the query in the order in which they are listed.

A Parameter element requires the name parameter. The value of the name parameter can be a physical attribute or a [well-known attribute](#) (see page 75).

Note: Identity Manager must understand the values that are passed to a query in the Parameter element. For example, the value can be a physical name or a well-known attribute that is defined in the ImsManagedObjectAttr attributes.

When you specify a physical attribute, note the following:

- Use the following syntax to specify a physical attribute:

tablename.columnname

- *tablename*

Provides the name of the table where the attribute is located. The table you specify should be the primary table.

- *columnname*

Provides the name of the column that stores the attribute.

- The attribute that you specify must exist in the database and be defined in the directory configuration file, as described in [How to Modify Attribute Descriptions](#) (see page 113).

Example: Custom Operations for the Business Number Attribute

In the following example, the Business Number attribute is generated by calling a stored procedure; it is not a physical attribute in the database.

```
<ImManagedObjectAttr wellknown="%BUSINESS_NUMBER%" displayname="Business
Number" description="Business Number" valuetype="String" required="false"
multivalued="false" maxlength="0">
  <Operation name="Get" value="call sp_getbusinessnumber(?)">
    <Parameter name="%USER_ID%"/>
  </Operation>
  <Operation name="Set" value="call sp_setbusinessnumber(?,?)">
    <Parameter name="%USER_ID%"/>
    <Parameter name="%BUSINESS_NUMBER%"/>
  </Operation>
  <Operation name="Delete" value="call sp_deletebusinessnumber(?)">
    <Parameter name="%USER_ID%"/>
  </Operation>
```

Note the following:

- `sp_getbusinessnumber`, `sp_setbusinessnumber`, and `sp_deletebusinessnumber` are user-defined stored procedures.
- The value returned from the Get operation is mapped to the `%BUSINESS_NUMBER%` attribute.
- The question mark (?) indicates substitutions that are made at runtime before the query is executed. For example, in the Get operation the `%USER_ID%` well-known attribute is passed to the `sp_getbusinessnumber` stored procedure.

How to Specify the Unique Identifier for a Managed Object

Each object that Identity Manager manages must have a unique identifier. The unique identifier must be stored in a single column in the managed object's primary table. Primary tables are described in [Database Tables](#) (see page 109).

Use the `UniqueIdentifier` and `UniqueIdentifierAttr` elements to define the unique identifier as follows:

```
<UniqueIdentifier>
  <UniqueIdentifierAttr name="tablename.columnname" />
</UniqueIdentifier>
```

The `UniqueIdentifierAttr` element requires the `name` parameter. The value of the `name` parameter is the attribute in which the unique identifier is stored. The value can be a physical attribute or a [well-known attribute](#) (see page 75).

When you specify a physical attribute, note the following:

- The attribute that you specify must exist in the database and be defined in the directory configuration file, as described in [How to Modify Attribute Descriptions](#) (see page 113). In the attribute description, be sure to specify read-only or write-once permission to prevent the unique identifier from changing during a session.
- Use the following syntax to specify a physical attribute:

tablename.columnname

tablename

Defines the name of the table where the attribute is located. The table you specify should be the primary table.

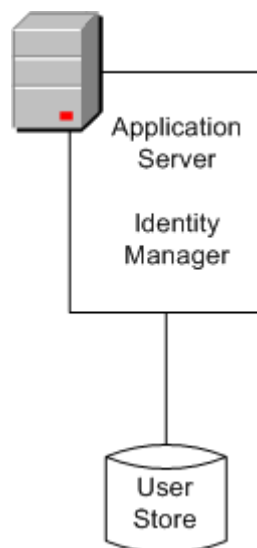
columnname

Defines the name of the column that stores the attribute.

- If the unique identifier is generated by the database, you must specify a [custom operation for the attribute](#) (see page 112). For example, you may have to specify an operation that fetches the last generated identifier from the database.

Connection to the User Directory

Identity Manager connects to a user directory to store user, group, and organizational information, as shown in the following illustration:



A new directory or database is not required. However, the existing directory or database must be located on a system that has a fully qualified domain name (FQDN).

For a list of supported directory and database types, see the [CA support site](#). Search for the Identity Manager support matrix.

You configure a connection to the user store when you create an Identity Manager directory in the Management Console.

If you export the directory configuration after creating an Identity Manager directory, the user directory connection information is displayed in the Provider element of the directory configuration file.

Description of a Database Connection

To describe a database connection, use the Provider element and its subelements in the directory.xml file.

Note: If you are creating a new Identity Manager directory, you do not need to provide directory connection information in the directory.xml file. You provide connection information in the Identity Manager Directory wizard in the Management Console.

Modify the Provider element for updates only.

Provider Element

The Provider element includes the following subelements:

Credentials (required)

Supplies the username and password for accessing the database.

DSN (required)

Identifies the ODBC datasource to use when connecting to the user store.

Note: This subelement only applies when Identity Manager integrates with SiteMinder. In Identity Manager environments that do not include SiteMinder, this subelement is ignored.

SiteMinderQuery

Specifies custom query schemes for locating user information in a relational database.

Note: This subelement only applies when Identity Manager integrates with SiteMinder. In Identity Manager environments that do not include SiteMinder, this subelement is ignored.

A completed database connection resembles the following:

```
<Provider type="RDB" userdirectory="@SMDirName">
  <Credentials user="@SMDirUser"
    cleartext="true">@SMDirPassword</Credentials>
  <DSN name="@SMDirDSN" />
</Provider>
```

The parameters for the Provider element are as follows:

type

Specifies the type of database. For Microsoft SQL Server and Oracle databases, specify RDB (default).

userdirectory

Specifies the name of the user directory connection. This parameter corresponds to the Connection Object name that you supply during directory creation.

If Identity Manager integrates with SiteMinder for authentication, it creates a user directory connection in SiteMinder with the name you specify for the Connection Object during installation. If you want to connect to an existing SiteMinder user directory, enter the name of that user directory when prompted for the Connection Object. Identity Manager will populate the userdirectory parameter with the name you specify.

If Identity Manager does not integrate with SiteMinder, the value of the userdirectory parameter is any name that you give the JDBC connection to the user store.

Note: Do not specify a name for the user directory connection in the directory.xml file. Identity Manager prompts you to supply the name during directory creation.

Database Credentials

To connect to the database, Identity Manager must provide valid credentials to the data source. The credentials are defined in the Credentials element, which resembles the following example:

```
<Credentials user="@SMDirUser" cleartext="true">
  "MyPassword"
</Credentials>
```

If you do not specify a password in the Credentials element, you are prompted for the password when you create the Identity Manager directory in the Management Console.

Note: We recommend specifying the password in the Management Console.

If you specify the password in the Management Console, Identity Manager encrypts the password for you. Otherwise, if you do not want the password to appear in clear text, you must encrypt the password using the password tool installed with Identity Manager. SiteMinder Passwords has instructions on using the password tool.

Note: You can specify only one set of credentials. When you define multiple data sources, the credentials that you specify must apply to all data sources.

The credential parameters are as follows:

user

Defines the login ID for an account that can access the data source.

Do not specify a value for the user parameter in the directory.xml file. Identity Manager prompts you to supply the login ID when you create the Identity Manager directory in the Management Console.

cleartext

Determines whether the password is displayed in clear text in the directory.xml file:

- True—The password is displayed in clear text.
- False—The password is encrypted (default).

Note: These parameters are optional.

Data Source Name (DSN)

The DSN element in the directory.xml file has one parameter—the name of the ODBC data source that Identity Manager uses to connect to the database. The value of the name parameter must match the name of an existing data source.

Note: This element applies only when Identity Manager integrates with SiteMinder. If Identity Manager does not integrate with SiteMinder, this element is ignored.

If the value of the name parameter is @SmDirDSN, you do not have to specify a DSN name in the directory.xml file. Identity Manager will prompt you to supply the DSN name when you import the directory.xml file.

To configure failover, define multiple DSN elements. If the primary data source fails to respond to a request, the next data source defined responds to the request.

For example, suppose you configure the following:

```
<DSN name="DSN1">  
<DSN name="DSN2">
```

Identity Manager uses the data source DSN1 to connect to the database. If there is a problem with DSN1, Identity Manager will try to connect to the database using DSN2.

Note: The credentials you specify in the [Credentials element](#) (see page 121) must apply to all the DSNs that you define.

SQL Query Schemes

Identity Manager uses query schemes to find user and group information in a relational database.

Note: This element applies only when Identity Manager integrates with SiteMinder. In environments that do not include SiteMinder, this parameter is ignored.

When you create an Identity Manager directory in the Management Console, Identity Manager generates a set of query schemes that are based on the required query schemes in SiteMinder. (For complete information about SiteMinder query schemes, see the *CA SiteMinder Web Access Manager Policy Server Configuration Guide*.) The table and column names in the SiteMinder query schemes are replaced with data that you specify in the directory configuration file.

How to Define Custom Query Schemes

Query schemes are defined in SiteMinderQuery elements in the directory configuration file. A SiteMinderQuery element resembles the following:

```
<SiteMinderQuery name="SetUserProperty" query="update tblUsers set %s =  
&apos;%s&apos; where loginid = &apos;%s&apos;" />
```

Note: In the sample query, ' is the XML syntax for the single quote (').

The SiteMinderQuery element only applies when Identity Manager integrates with SiteMinder.

The query scheme parameters are as follows:

name

Specifies the redefined name of a SiteMinder query scheme.

Do not modify this value.

query

Specifies the SQL statement or stored procedure to execute. The valid values are as follows:

- INSERT
- SELECT
- UPDATE
- DELETE
- CALL (for stored procedures)

Note: These parameters are required for the SiteMinderQuery element.

Before you customize query schemes, you should do the following:

- Become familiar with the default query schemes.

Note: For more information on SQL query schemes see the *CA SiteMinder Web Access Manager Policy Server Configuration Guide*.

- Acquire extensive experience developing SQL queries.

Modify the Default Query Schemes

Perform the following procedure to modify the default query schemes.

To modify the default query schemes

1. Export the directory configuration file.

Identity Manager generates a directory configuration file that contains all the current settings for the Identity Manager directory, including the generated query schemes.

2. Save the directory configuration file.

Note: If you want to create a backup of the original directory configuration file, save the file with a different name or in a different location before saving the exported file.

3. Locate the Identity Manager-generated query scheme that you want to modify.

4. Enter the query scheme or stored procedure to execute in the query parameter.

Note: Do not modify the query name.

5. After making all your changes, save the directory configuration file.

Import the file to [update the Identity Manager Directory](#) (see page 146).

Well-Known Attributes for a Relational Database

Well-known attributes have special meaning in Identity Manager. They are identified by the following syntax:

`%ATTRIBUTENAME%`

In this syntax, *ATTRIBUTENAME* must be uppercase.

A well-known attribute is mapped to one physical attribute using an [attribute description](#) (see page 113).

In the following attribute description, the attribute `tblUsers.password` is mapped to the well-known attribute `%PASSWORD%` so that Identity Manager will treat the value in `tblUsers.password` as a password:

```
<ImManagedObjectAttr
  physicalname="tblUsers.password"
  displayname="Password"
  description="Password"
  valuetype="String"
  required="false"
  multivalued="false"
  wellknown="%PASSWORD%"
  maxlength="0" />
```

Some well-known attributes are required; others are optional.

User Well-Known Attributes

A list of user well-known attributes follows:

%ADMIN_OF%

Contains the list of groups for which the user is an administrator.

This well-known attribute may improve search performance at sites with many groups. If the `%ADMIN_OF%` well-known attribute is specified, Identity Manager looks for the groups that a user can manage in the `%ADMIN_OF%` attribute, instead of checking every group in the user store.

%ADMIN_ROLE_CONSTRAINT%

Contains the list of [administrator's admin roles](#) (see page 128).

The physical attribute mapped to `%ADMIN_ROLE_CONSTRAINT%` must be multivalued to accommodate multiple roles.

We recommend indexing the attribute that is mapped to `%ADMIN_ROLE_CONSTRAINT%`.

%CERTIFICATION_STATUS%

(Required for using the user certification feature)

Contains the user's certification status.

Note: For more information about user certification, see the *Administration Guide*.

%DELEGATORS%

Maps to a list of users who have delegated work items to the current user.

This attribute is required to use delegation. The physical attribute mapped to %DELEGATORS% must be multi-valued and capable of holding strings.

Important! Editing this field directly using Identity Manager tasks or an external tool can cause significant security implications.

%EMAIL%

(Required for enabling the email notification feature)

Stores a user's email address

%ENABLED_STATE%

(Required)

Tracks a user's status.

Note: The data type of the physical attribute mapped to %ENABLED_STATE% must be String.

%FIRST_NAME%

Contains a user's first name.

%FULL_NAME%

(Required)

Contains a user's first and last name.

%IDENTITY_POLICY%

Contains the list of identity policies that have been applied to a user account.

Identity Manager uses this attribute to determine whether an identity policy should be applied to a user. If the policy has the Apply Once setting enabled, and the policy is listed in the %IDENTITY_POLICY% attribute, Identity Manager does not apply the changes in the policy to the user.

Note: For more information about identity policies, see the *Administration Guide*.

%LAST_CERTIFIED_DATE%

(Required for using the user certification feature)

Contains the date when a user's roles were certified.

Note: For more information about user certification, see the *Administration Guide*.

%LAST_NAME%

Contains the user's last name.

%ORG_MEMBERSHIP%

(Required when organizations are supported)

Contains the unique identifier for the organization to which the user belongs.

%ORG_MEMBERSHIP_NAME%

(Required when organizations are supported)

Contains the user-friendly name of the organization to which the user belongs.

%PASSWORD%

Contains a user's password.

%PASSWORD_DATA%

(Required for password policy support)

Specifies the attribute that tracks password policy information.

%PASSWORD_HINT%

(Required)

Contains user-specified question and answer pairs. The question and answer pairs are used in case of forgotten passwords.

%USER_ID%

(Required)

Stores a user's login ID.

Group Well-Known Attributes

A list of group well-known attributes follows:

%GROUP_ADMIN%

Contains a group's administrators.

Note: The %GROUP_ADMIN% attribute must be multivalued.

% GROUP_DESC %

Contains a group's description.

% GROUP_ID %

Contains the unique identifier for a group.

% GROUP_MEMBERSHIP %

(Required)

Contains a list of the group's members.

Note: The %GROUP_MEMBERSHIP% attribute must be multivalued.

% GROUP_NAME %

(Required)

Stores a group's name.

% ORG_MEMBERSHIP %

(Required when organizations are supported)

Contains the unique identifier for the organization to which the group belongs.

% ORG_MEMBERSHIP_NAME %

(Required when organizations are supported)

Contains the user-friendly name of the organization to which the group belongs.

% SELF_SUBSCRIBING %

Determines whether users can subscribe to a group.

%Admin_Role_Constraint% Attribute

When you create an admin role, you specify one or more rules for role membership. Users who satisfy the membership rules have the role. For example, if the membership rule for the User Manager role is title=User Manager, users who have the title User Manager have the User Manager role.

Note: For more information about rules, see the *Administration Guide*.

%ADMIN_ROLE_CONSTRAINT% lets you designate one profile attribute to store all of an administrator's admin roles.

How to Use the %ADMIN_ROLE_CONSTRAINT% Attribute

To use the %ADMIN_ROLE_CONSTRAINT% as the constraint for all admin roles, do the following:

- Pair the %ADMIN_ROLE_CONSTRAINT% well-known attribute with a multivalued profile attribute to accommodate multiple roles
- When you configure an admin role in the Identity Manager user interface, make sure the constraint is the following:

Admin Roles equals *role name*

role name

Defines the name of the role for which you are supplying the constraint.

For example, Admin Roles equals User Manager

Note: Admin Roles is the default displayname for the %ADMIN_ROLE_CONSTRAINT% attribute.

Configure Well-Known Attributes

Perform the following procedure to configure well-known attributes.

To configure well-known attributes

1. In the directory configuration file, search for the following:

`##`

Required values are identified by two pound signs (##).

2. Replace the value that begins with ## with the physical name of the attribute you want as it exists in the database. Supply the attribute name using the following format:

tablename.columnname

For example, if the password attribute is stored in the password column in the tblUsers table, specify the following:

`tblUsers.password`

3. Repeat Steps 1 and 2 until you have replaced all required values and included optional values that you want.
4. Map optional well-known attributes to physical attributes, as necessary.
5. Save the directory configuration file.

How to Configure Self-Subscribing Groups

You can enable self-service users to join groups by configuring support for self-subscribing groups in the directory configuration file.

1. In the Self-subscribing Groups section, add a SelfSubscribingGroups element as follows:

```
<SelfSubscribingGroups type=search_type org=org_dn>
```

2. Type values for the following parameters:

type

Indicates where Identity Manager searches for self-subscribing groups. The valid values are as follows:

- **NONE**—Identity Manager does not search for groups. Specify NONE to prevent users from subscribing to groups.
- **ALL**—Identity Manager searches all groups in the user store. Specify ALL when users can subscribe to all groups.
- **INDICATEDORG** (*for environments that support organizations only*)—Identity Manager searches for self-subscribing groups in the user's organization and its suborganizations. For example, when a user's profile is in the Marketing organization, Identity Manager searches for self-subscribing groups in the Marketing organization, and in all suborganizations.
- **SPECIFICORG** (*for environments that support organizations only*)—Identity Manager searches in a specific organization. Supply the unique identifier of the specific organization in the org parameter.

org

Defines the unique identifier of the organization where Identity Manager searches for self-subscribing groups.

Note: You must specify the org parameter if type=SPECIFICORG.

3. Restart the SiteMinder Policy Server if you changed any of the following:
 - The type parameter to or from SPECIFICORG
 - The value of the org parameter

Once support for self-subscribing groups is configured in the Identity Manager directory, Identity Manager administrators can specify which groups are self-subscribing in the User Console.

When a user self-registers, Identity Manager looks for groups in the specified organizations, and then displays the self-subscribing groups to the user.

Validation Rules

A validation rule enforces requirements on data that a user types in a task screen field. The requirements can enforce a data type or format, or make sure that the data is valid in the context of other data on the task screen.

Validation rules are associated with profile attributes. Before processing a task, Identity Manager ensures that the data entered for a profile attribute satisfies any associated validation rules.

You can define validation rules and associate them with profile attributes in the directory configuration file.

Organization Management

For relational databases, Identity Manager has the option to manage organizations. When your database supports organizations, the following is true:

- Organizations have a hierarchical structure.
- All managed objects, such as users, groups, and other organizations belong to an organization.
- When you delete an organization, the objects that belong to that organization are also deleted.

You configure the organization object in the same way that you configure the user and group objects with a few additional steps.

How to Set Up Organization Support

Implement the following steps to set up organization support:

1. [Configure Organization Support in the Database](#) (see page 132),
2. Describe the organization object in the [ImsManagedObject](#) (see page 108).
Be sure to configure the Table and UniqueIdentifier subelements.
3. Configure the [top-level organization](#) (see page 132).
4. [Describe the attributes](#) (see page 113) that constitute an organization.
5. Define the well-known attributes for the [organization object](#) (see page 133).

Configure Organization Support in the Database

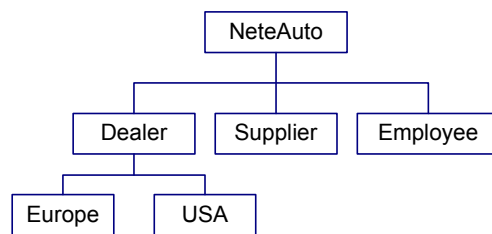
To configure organization support in the database

1. Open one of the following SQL scripts in an editor:
 - Microsoft SQL Server databases:
ims_mssql_rdb.sql
 - Oracle databases:
ims_oracle_rdb.sqlThese files are located in
im_admin_tools_dir\directoryTemplates\RelationalDatabase
2. In the SQL script, search for and replace <@primary organization table@> with the name of the primary table for the organization object. Save the SQL script.
3. Run the SQL script against the database.

Root Organization Specification

The root organization serves as the top-level or parent organization in the directory. All organizations relate to the root organization.

In the following illustration, NeteAuto is the root organization. The other organizations are suborganizations of NeteAuto:



A complete root organization definition resembles the following sample:

```
<ImManagedObject name="Organization" description="My Organizations"
objecttype="ORG">
  <RootOrg value="select orgid from tblOrganizations where parentorg is null">
    <Result name="%ORG_ID%" />
  </RootOrg>
</ImManagedObject>
```

After you define the basic information for the organization object, including the tables that constitute the organization profile and the organization object's unique identifier, specify the root organization in the directory.xml file:

- In the value parameter of the RootOrg element, define the query that Identity Manager uses to retrieve the root organization, as in the following example:

```
<RootOrg value="select orgid from tblOrganizations where parentorg is null">
```

- In the name parameter of the Result element, type the unique identifier of the organization, as in the following example:

```
<Result name="%ORG_ID%" />
```

Note: The value of the name parameter should be the unique identifier for the organization object.

Well-Known Attributes for Organizations

Define well-known attributes for the attributes in an organization's profile as described in [Well-Known Attributes](#) (see page 75).

The required and optional organization well-known attributes are as follows:

% ORG_DESCR %

Contains an organization's description.

% ORG_MEMBERSHIP %

(Required)

Contains the organization's parent organization.

Note: See Defining the Organizational Hierarchy for more information about the %ORG_MEMBERSHIP% attribute.

% ORG_MEMBERSHIP_NAME %

(Required)

Contains the user-friendly name of an organization's [parent organization](#) (see page 134).

% ORG_NAME %

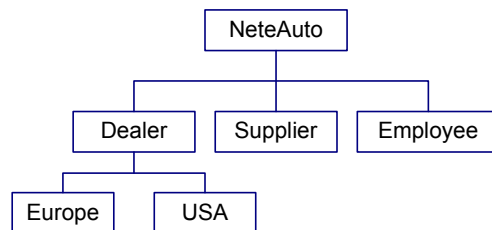
(Required)

Contains the name of the organization.

How to Define the Organizational Hierarchy

In Identity Manager, organizations have a hierarchical structure that includes a root organization and suborganizations. The suborganizations may also have suborganizations.

Each organization, except the root organization, has a parent organization. For example, in the following illustration, Dealer is the parent organization for the USA and Europe organizations:



The unique identifier of the parent organization is stored in an attribute in the organization's profile. Using the information in this attribute, Identity Manager can construct the organization hierarchy.

To specify the attribute that stores the parent organization, use the %ORG_MEMBERSHIP% and %ORG_MEMBERSHIP_NAME% well-known attributes with the physical attribute that stores the name of the parent organization in an attribute description as follows:

```
<ImManagedObjectAttr physicalname="tblOrganizations.parentorg"
displayname="Organization" description="Parent Organization" valuetype="Number"
required="false" multivalued="false" wellknown="%ORG_MEMBERSHIP%" maxlength="0"
/>
```

How to Improve Directory Search Performance

To improve the performance of directory searches for users, organization, and groups, do the following:

- Index the attributes that administrators can specify in search queries.
- Override the default directory timeout setting by specifying values for the timeout search parameters in a directory configuration file (directory.xml)
- Tune the user directory. See the documentation for the database that you are using.

Configure database-specific options in the ODBC data source. For more information, see the documentation for the data source.

Chapter 6: Managing Identity Manager Directories

An Identity Manager directory provides information about a user directory that Identity Manager manages. This information describes how objects such as users, groups, and organizations are stored in the user store and displayed in Identity Manager.

You create, view, export, update, and delete Identity Manager directories in the Identity Manager directory section of the Management Console.

Note: If Identity Manager uses a cluster of SiteMinder Policy Servers, stop all but one Policy Server before creating or updating Identity Manager directories.

This section contains the following topics:

[Prerequisites to Creating an Identity Manager Directory](#) (see page 135)

[Create an Identity Manager Directory in the Management Console](#) (see page 136)

[View an Identity Manager Directory](#) (see page 138)

[Identity Manager Directory Properties](#) (see page 138)

[How to Update Settings for an Identity Manager Directory](#) (see page 145)

Prerequisites to Creating an Identity Manager Directory

Before you create an Identity Manager Directory, you must do the following:

- Configure a directory configuration file (directory.xml). See the relevant section for information:
 - [Creating an Identity Manager Directory for an LDAP User Store](#) (see page 47).
 - [Creating an Identity Manager Directory for a Relational Database](#) (see page 93).
- Stop all but one Identity Manager node before you create or modify an Identity Manager Directory.

Note: When you have a cluster of Identity Manager nodes, only one Identity Manager node can be enabled when you make changes in the Management Console.

- Stop all but one Policy Server before creating or updating Identity Manager Directories.

Note: When you have a cluster of SiteMinder Policy Servers, only one SiteMinder policy server can be enabled when you make changes in the Management Console.

Create an Identity Manager Directory in the Management Console

The Identity Manager directory wizard guides you through the steps to create an Identity Manager Directory.

To create an Identity Manager directory

1. Open the Management Console by typing the following URL in a browser:

`http://domain_name:port/idmmanage`

domain_name

Defines the fully qualified domain name of the server where Identity Manager is installed

port

Defines the application server port number.

2. Click Directories.

The Identity Manager Directories window appears.

3. Click New to start the Identity Manager directory wizard.

4. Type the path and filename of the directory configuration XML file for creating the Identity Manager Directory, or browse for the file. Click Next.

5. Supply values for the fields on this window as follows:

Name

Determines the name of the Identity Manager directory that you are creating.

Description

(Optional) Describes the Identity Manager Directory.

Connection Object Name

Specifies the name of the user directory that the Identity Manager Directory describes. Enter *one* of the following:

- If Identity Manager does not integrate with SiteMinder, specify any meaningful name for the object that Identity Manager uses to connect to the user store.
- If Identity Manager integrates with SiteMinder and you want to create a new user directory connection object in SiteMinder, specify any meaningful name. Identity Manager creates the user directory connection object in SiteMinder with the name you specify.
- If Identity Manager integrates with SiteMinder and you want to connect to an existing SiteMinder user directory, specify the name of the SiteMinder user directory connection object exactly as it appears in the Policy Server user interface.

Host (for LDAP directories only)

Specifies the host name or IP address of the system where the user directory is installed.

For eTrust Directory user stores, use the full domain name of the host system. Do not use localhost.

For Active Directory user stores, specify the domain name, not the IP address.

Port (for LDAP directories only)

Specifies the port number of the user directory.

Provisioning Domain

Provisioning Domain that Identity Manager manages.

Note: The Provisioning Domain name is case-sensitive.

Username / User DN

Specifies the user name for an account that can access the user store.

For Provisioning user stores, the user account you specify must have the Domain Administrator profile, or an equivalent set of privileges for the Provisioning Domain.

Password

Specifies the password for the user account that you specified in the Username (for Relational Databases) or User DN field (for LDAP directories).

Confirm Password

Enter the password that you typed in the Password field again for confirmation.

Secure Connection (for LDAP directories only)

Indicates whether Identity Manager uses a secure connection.

Be sure to select this option for Active Directory user stores.

Note: The fields that appear on this window depend on the user store type and the information you provided in the directory configuration file in Step 4. If you provided values for any of these fields in the directory configuration file, Identity Manager does not prompt you to supply these values again.

Click Next.

6. Review the settings for the Identity Manager Directory. Click Finish to create the Identity Manager Directory with the current settings or click Previous to make changes.

Status information is displayed in the Directory Configuration Output window.

7. Click Continue to exit the wizard.

View an Identity Manager Directory

Perform the following procedure to view an Identity Manager Directory.

To view an Identity Manager directory

1. In the Identity Manager Management Console, click Directories.
2. Click the name of the Identity Manager directory to view. The Directory Properties window appears, showing the Identity Manager directory properties.

Identity Manager Directory Properties

The Identity Manager Directory properties are as follows:

Name

Defines the unique name of the Identity Manager Directory.

Description

Provides a description for the Identity Manager Directory.

Type

Defines the type of directory provider.

Connection Object Name

Displays the name of the user directory that the Identity Manager Directory describes.

If Identity Manager integrates with SiteMinder, the connection object name matches the name of the SiteMinder user directory connection.

URL

Provides the URL or IP address of the user store.

Username

Specifies the user name for an account that can access the user store.

Search Maximum Rows (for LDAP directories only)

Indicates the maximum number of rows returned as the result of a search.

Search Timeout

Specifies the maximum number of seconds that Identity Manager searches a user store before terminating the search.

Provisioning Domain

Provisioning Domain that Identity Manager manages.

Identity Manager Directory Properties Window

General information about an Identity Manager directory is presented in the properties window for the directory that you select. The Directory Properties window is divided into the following sections:

- Directory Properties—Basic properties for the Identity Manager directory.
- [Managed Objects](#) (see page 140)—Descriptions of the type of user store objects that Identity Manager manages.
- [Validation Rule Sets](#) (see page 144)—List of validation rule sets that apply to the Identity Manager directory.

- **Environment(s)**—The Identity Manager environments associated with the Identity Manager directory. An Identity Manager directory may be associated with multiple Identity Manager environments.

To view more information about an Identity Manager environment, click the name of the environment.

To modify properties in an Identity Manager directory, import a directory configuration file as described in [Update an Identity Manager Directory](#) (see page 146).

How to View Managed Objects Properties and Attributes

A managed object describes a type of entry in the user store, such as a user, group, or organization. The properties and attributes that apply to a managed object apply to all entries of that type. For example, a user profile will have all the properties and attributes of the User managed object.

To view the details of a managed object, click the object's name to open the Managed Object Properties window.

Managed Object Properties

The Managed Object Properties window describes the properties and attributes for a type of managed object.

The information about the Managed Object Properties window depends on the type of user store you are managing. A managed object's properties are as follows:

Description

Provides a description of the managed object.

Type

Indicates the type of entry that the managed object represents. An object's type can be one of the following types:

- User
- Group
- Organization

Object Class (for LDAP directories only)

Specifies the object classes for the managed object. A managed object may have multiple object classes.

Sort Order (for LDAP directories only)

Specifies the attribute that Identity Manager uses to sort search results in custom business logic. It does not affect the order of search results in the User Console.

For example, when you specify the cn attribute for the user object, Identity Manager sorts the results of a search for users alphabetically by the cn attribute.

Primary Table (for relational databases only)

Specifies the table that contains the unique identifier for the managed object.

Container Properties (for LDAP Directories Only)

In an LDAP directory, a *container* groups objects of a specific type. When a container is specified, Identity Manager handles only entries in the container. For example, when you specify the container ou=People, Identity Manager handles only users who exist in the People container.

Note: Users and groups that exist in the LDAP directory but not in the defined container may appear in the User Console. You may experience problems when managing those users and groups.

Containers group users and groups only. You cannot specify a container for organizations.

The properties of a container are as follows:

objectclass

Specifies the LDAP object class of the container where objects of a specific type are created. For example, the default value for the user container is "top,organizationalUnit," which indicates that users are created in LDAP organizational units (ou).

ID

Specifies the attribute that stores the container name, for example, ou. The attribute is paired with the Name value to form the relative DN of the container, as in the following example:

ou=People

Name

Specifies the container name.

Secondary Table Properties (for Relational Databases Only)

Secondary tables contain additional attributes for a managed object. For example, a secondary table named `tblUserAddress` may contain the street, city, state, and ZIP code attributes for the User managed object.

The following properties are displayed for secondary tables:

Table

Specifies the name of the table.

Reference

Describes the mapping between the primary table and the secondary table.

The reference is displayed using the following format:

primarytable.attribute=secondarytable.attribute

For example, `tblUsers.id = tblUserAddress.userid` indicates that the `id` attribute in the primary table, `tblUsers`, maps to the `userid` attribute in the `tblUserAddress` table.

Attributes Properties in the Managed Object Properties Window

The following properties are displayed for attributes in the Managed Object Properties window:

Display Name

The user-friendly name of the attribute. This name appears in the list of available attributes when you design a task window for a particular task in the User Console.

Physical Name

The name of the attribute in the user store.

Well-Known Name

Well-known names indicate attributes that have special meaning in Identity Manager, such as the attribute used to store user passwords.

Attribute Properties in the Attribute Properties Windows

You can see additional details about an attribute by clicking its name to open the Attribute Properties window.

The following attribute properties are displayed in the Attribute Properties window:

Description

Provides a description for the attribute.

Physical Name

Specifies the name of the attribute in the user store.

Object Class (for user, group, and organization attributes in LDAP directories only)

The LDAP auxiliary class for a user attribute, when the attribute is not part of the primary object class specified for the User object.

You can specify an auxiliary object class for the User and Group objects only.

Well-Known Name

Indicates attributes that have special meaning in Identity Manager, such as the attribute used to store user passwords.

Required

Indicates whether a value is required for the attribute, as follows:

- True indicates that the attribute must have a value.
- False indicates that a value is optional.

Read Only

Indicates the permission level for an attribute, as follows:

- True indicates that the attribute cannot be modified.
- False indicates that the attribute can be modified.

Hidden

Indicates whether an attribute can be displayed in a task window for a particular task.

Hidden attributes are often used in logical attribute schemes.

Note: For more information, see the *Programming Guide for Java*.

Supports Multiple Values

Indicates whether the attribute can have multiple values, as follows (for example, the attribute used to store the members of a group is multivalued):

- True indicates that the attribute can support multiple values.
- False indicates that the attribute can have only a single value.

Multiple Value Delimiter (for relational databases only)

The character that separates values when multiple values are stored in a single column.

System Attribute

Indicates whether the attribute is used only by Identity Manager, as follows:

- True indicates that the attribute is a system attribute. The attribute is not available to add to task windows.
- False indicates that the attribute can be used by users. The attribute may appear on task windows.

Data Type

Specifies the attribute's data type. The default value is String.

Maximum Length

Specifies the maximum length that an attribute value can have. If set to 0, there is no limit on the value's length.

Validation Rule Set

Specifies the name of a validation rule set, when the attribute is associated with one.

Validation Rule Sets

A validation rule enforces requirements on data that a user types in a task window field. The requirements can enforce a data type or format, or ensure that the data is valid in the context of other data in the task window.

One or more validation rules are grouped in a validation rule set. A validation rule set is then associated with a profile attribute. For example, you can create a validation rule set that contains a Format Date validation rule, which enforces a date format of mm-dd-yyyy. You can then associate the validation rule set with the attribute that stores an employee's start date.

Note: You create validation rules and rule sets in the directory configuration file or in the User Console.

The Managed Object Properties window displays a list of validation rule sets that apply to the Identity Manager directory. To view the details of a validation rule set, click the rule set's name to open the Validation Rule Set Properties window.

Validation Rule Properties

The following information is displayed in the Validation Rule Properties window:

Name

Provides the name of the validation rule.

Description

Provides a description of the rule.

Class

Provides the name of the Java class that implements the validation rule.

This field does not appear unless the validation rule is defined in a Java class.

Filename

Provides the name of the file that contains the JavaScript implementation of the validation rule.

This field does not appear unless the validation rule is defined in a file.

Regular Expression

Provides the regular expression that implements the validation rule.

This field does not appear unless the validation rule is defined as a regular expression.

Validation Rule Set Properties

The following information is displayed in the Validation Rule Set Properties window:

Name

Specifies the name of the validation rule set.

Description

Provides a description for the validation rule set.

The Validation Rule Set Properties page also includes a list of validation rules in the set. You can click the name of the validation rule to open the Validation Rule Properties window.

How to Update Settings for an Identity Manager Directory

To view the current settings of an Identity Manager directory, you must export it and save it as an XML file.

After you export the directory settings, you can modify and re-import the XML file to update the directory, or you can import the XML file to another directory to configure the same settings for that directory.

Export an Identity Manager Directory

Perform the following procedure to export an Identity Manager directory.

To export an Identity Manager directory

1. Click Directories.
The list of Identity Manager directories appears.
2. Click the name of the directory to export.
The Properties for the Identity Manager directory window appears.
3. At the bottom of the properties window, click Export.
4. When prompted, save the XML file.

Update an Identity Manager Directory

The purpose of updating an Identity Manager directory is the following:

- Adding or changing managed object definitions, including an object's attributes
- Setting search parameters
- Changing directory properties

Note: Identity Manager does not delete object or attribute definitions.

The directory configuration file can contain only the changes that you want to make. You do not have to include properties or attributes that are already defined.

Note: When you have a cluster of Identity Manager nodes, only one Identity Manager node can be enabled when you make changes in the Management Console. Stop all but one Identity Manager node before you create or modify an Identity Manager directory.

To update an Identity Manager directory in the Management Console

1. Export the current Identity Manager directory settings to an XML file.
2. Modify the XML file to reflect your changes.
3. Click Directories.
The list of Identity Manager directories appears.
4. Click the name of the directory to update.
Properties for the Identity Manager directory appear.
5. At the bottom of the properties window, click Update.

6. Type the path and file name of the XML file for updating the Identity Manager directory, or browse for the file. Click Finish.

Status information is displayed in the Directory Configuration Output field.

7. Click Continue.

Delete an Identity Manager Directory

Before you delete an Identity Manager Directory, you must delete any Identity Manager Environments associated with it.

To delete an Identity Manager Directory

1. In the Management Console, click Directories.
The list of Identity Manager Directories appears.
2. Select the check box to the left of the directory (or directories) to delete.
3. Click Delete.
A confirmation message appears.
4. Click OK to confirm the deletion.

Chapter 7: Managing Identity Manager Environments

This section contains the following topics:

[Identity Manager Environments](#) (see page 149)
[Prerequisites to Creating an Identity Manager Environment](#) (see page 150)
[Create an Identity Manager Environment](#) (see page 151)
[How to Access an Identity Manager Environment](#) (see page 155)
[Modify Identity Manager Environment Properties](#) (see page 156)
[Export an Identity Manager Environment](#) (see page 158)
[Import an Identity Manager Environment](#) (see page 159)
[Restart an Identity Manager Environment](#) (see page 159)
[Delete an Identity Manager Environment](#) (see page 160)
[Optimize Policy Rule Evaluation](#) (see page 161)
[Role and Task Settings](#) (see page 162)
[Modify the System Manager Account](#) (see page 163)
[Access the Status of an Identity Manager Environment](#) (see page 165)

Identity Manager Environments

An Identity Manager environment is a view of a user store. In an Identity Manager environment, you can manage users, groups, organizations, tasks, and roles.

Using the Management Console, you can do the following:

- Create, modify, or delete an Identity Manager environment
- Export and import an Identity Manager environment
- Configure advanced settings
- Import custom roles and tasks
- Reset the System Manager account

Prerequisites to Creating an Identity Manager Environment

Before you begin, use the worksheet in the following table to collect the information that you will need:

Identity Manager Environment Configuration Worksheet	
Required Information	Value
A meaningful Identity Manager environment name that you choose For example: MyEnvironment	
A base URL, which Identity Manager uses to form the Redirect URL for the default password policy for the environment For example: http://server.yourcompany.org	
An alias that is added to the URL for accessing protected tasks in the environment For example: http://server.yourcompany.org/idm/alias	
An alias that is added to the URL for accessing public tasks, such as self-registration and forgotten password tasks For example: http://server.yourcompany.org/idm/public_alias/index.jsp?task.tag=SelfRegistration	
Note: When your environment does not include public tasks, you do not need to specify a public alias.	
If you supplied a public alias, the name of an existing user who will serve as the public user. Identity Manager uses the public user's credentials in place of user-supplied credentials when accessing public tasks.	

Identity Manager Environment Configuration Worksheet

Required Information**Value**

The name of an [Identity Manager directory](#) (see page 93)

The name of the provisioning directory, when the Identity Manager environment supports provisioning

The unique identifier for an existing user who will administer the Identity Manager environment

For example: myadmin

The name of the SiteMinder agent or agent group that will protect the Identity Manager environment

Create an Identity Manager Environment

Identity Manager environments let you manage objects in a directory with a set of roles and tasks. Use the Identity Manager environment wizard to guide you through the steps to create an Identity Manager Environment.

Note the following before creating an Identity Manager environment:

- If you are using an LDAP user store, and you configured a user container such as ou=People, in the directory configuration file (directory.xml) for your Identity Manager directory, ensure that the users you select when you create the Identity Manager environment exist in that container. Selecting a user account that does not exist in the user container may cause failures.
- When you configure an Identity Manager environment to manage an LDAP user directory with a flat or flat user structure, the profile for the user that you select must include the organization to which the user belongs. To ensure that the user's profile is configured correctly, add the name of the user's organization to the physical attribute that corresponds to the %ORG_MEMBERSHIP% well-known attribute in the [directory.xml file](#) (see page 82). For example, when the physical attribute description is mapped to the %ORG_MEMBERSHIP% well-known attribute in the directory.xml file, and the user belongs to the Employees organization, the user's profile should contain the attribute/value pair description=Employees.

To create an Identity Manager environment

1. If Identity Manager uses a cluster of Policy Servers, stop all but one Policy Server.
2. If you have a cluster of Identity Manager nodes, stop all but one Identity Manager node.
3. In the Management Console, click Environments.
4. Click New.

The Identity Manager environment wizard opens.

5. Supply the following information:

- **Environment name**

Specifies a unique name for the environment

- **Description**

Describes the environment

- **Protected alias**

Specifies a unique name that is added to the URL for accessing protected tasks in the Identity Manager environment. For example, when the alias is employees, the URL for accessing the employee environment is `http://myserver.mycompany.com/idm/employees`

Note: The alias is case-sensitive and cannot contain spaces. We recommend using lowercase letters without punctuation or spaces when you specify the alias.

- **Base URL**

Specifies the URL for Identity Manager. Do not include the alias, for example, `http://myserver.mycompany.com/idm`.

If you are using a Web Agent, the Base URL should be changed to reflect the URL of the Web Agent.

Note: If you are using a Web Agent to protect Identity Manager resources, do not specify a port number in the Base URL field. If you are using a Web Agent and the Base URL contains a port number, the links to Identity Manager tasks will not work properly.

For more information about protecting Identity Manager resources, see the *Installation Guide* for your application server.

Click Next.

6. Select an Identity Manager directory to associate with the environment you are creating, and click Next.

7. When the Identity Manager environment supports provisioning, select the appropriate provisioning server to use.

Note: You are not prompted to select a provisioning server if you selected a Provisioning directory as the Identity Manager directory.

8. Configure support for public tasks. Typically, these tasks are self-service tasks, such as self-registration or forgotten password tasks. Users do not need to log in to access public tasks.

Note: To enable users to use self-service tasks, you must configure public task support.

- a. Specify a unique name that is added to the URL for accessing public tasks.

Example: You would use the following URL to access the default self-registration task:

`http://myserver.mycompany.com/idm/alias/index.jsp?task.tag=SelfRegistration`

In this URL, *alias* is the unique name that you supply.

- b. Specify one of the following existing user accounts that will serve as the public user account. Identity Manager uses this account to allow unknown users to access public tasks without having to supply credentials.
 - LDAP users enter the unique identifier or relative DN of the public user account. This value must be mapped to the [%USER_ID% well-known](#) (see page 75). For example, if the user's DN is uid=Admin1, ou=People, ou=Employees, ou=NeteAuto, type Admin1.
 - Relational database users type the value that is mapped to the %USER_ID% well-known attribute in the directory configuration file, or the unique identifier for the user.

Click Validate to view the user's full identifier.

9. Select the tasks and roles to create for this environment. You can do the following:

- **Create default roles**

Creates a set of default tasks and roles that are initially available in the environment. Administrators can use these tasks and roles as templates for creating new tasks and roles in the User Console.

- **Create only the system manager role**

Creates only the System Manager role and the tasks associated with it.

The System Manager role is required to access the environment.

A System Manager can create new tasks and roles in the User Console.

■ **Import roles from the file**

Imports a role definition file that you exported from another Identity Manager environment.

Note: To use the Identity Manager environment, the role definitions file must include at least the System Manager role or a role that includes similar tasks.

Select the Import roles from the file radio button, and type the path and filename of the role definitions file or browse for the file to import.

10. Define a user to serve as the System Manager for this environment as follows:

- a. In the System Manager field, type the value that is mapped to the %USER_ID% well-known attribute in the directory configuration file, or specify one of the following user accounts:
 - LDAP users enter the unique identifier or relative DN of the user. For example, if the user's DN is uid=Admin1, ou=People, ou=Employees, type Admin1.
 - Relational database users type the unique identifier for the user.
- b. Click Add.

Identity Manager adds the complete identifier of the user to the list of users.

- c. Click Next.

Note: The System Manager should *not* be the same user as the administrator of the user store.

11. In the Inbound Administrator field, specify an Identity Manager administrator account that can execute admin tasks that are mapped to inbound mappings.

The user must be able to execute all those tasks on any user. The Provisioning Synchronization Manager role contains the provisioning tasks that are included in the default inbound mappings.

A page summarizing the settings for the environment appears.

Note: Be sure to configure the logout functionality for the agent that you select.

12. Review the settings for the environment. Click Previous to make changes or click Finish to create the Identity Manager environment with the current settings.

The Environment Configuration Output screen displays the progress of the environment creation.

13. Click Continue to exit the Identity Manager environment wizard.
14. Start the Environment.
15. If you stopped any Policy Servers in Step 1, restart them now.

How to Access an Identity Manager Environment

After you have created an Identity Manager environment, you can access it by typing a URL in a browser.

The format of the URL depends on how you configured the environment and the type of task that you want to access.

- To access protected tasks from the User Console, use the following URL:

`http://domain_name/idm/alias`

domain_name

Defines the fully qualified domain name of the server where Identity Manager is installed—for example, myserver.mycompany.com

alias

Defines the environment's alias, for example, employees.

Log into the Identity Manager environment with a privileged administrator account, such as the System Manager account that you created for the Identity Manager environment.

Note: All Identity Manager tasks are protected unless you configure public tasks.

- To access public tasks, which do not require users to provide credentials, use a URL with the following format:

`http://domain_name/idm/alias/index.jsp?task.tag=tasktag`

domain_name

Defines the fully qualified domain name of the server where Identity Manager is installed, for example, myserver.mycompany.com.

alias

Defines the alias for public tasks, for example, self-service.

task_tag

Defines the tag for the task to invoke.

You specify the task tag when you configure a task in the User Console.

The task tags for the default self-registration and forgotten password reset tasks are SelfRegistration and ForgottenPasswordReset.

Note: For more information, see the *Administration Guide*.

Modify Identity Manager Environment Properties

The Identity Manager environment Properties screen in the Management Console lets you do the following:

- View the current settings for the environment
- Modify the description, base URL, and protected and public aliases
- Import an existing Identity Manager environment after an upgrade

Note: For more information on importing existing Identity Manager environments, see the upgrade section of the *Installation Guide*.

- Start and stop the environment
- Access pages for configuring the following:
 - **Advanced Settings**
Configures advanced features, including features built using the Identity Manager APIs.
 - **Role and Task Settings**
Import a role definition file that you exported from another Identity Manager environment.
 - **System Manager**
Assigns system manager roles.

To modify environment properties

1. If Identity Manager uses a cluster of SiteMinder Policy Servers, stop all but one Policy Server.
2. If you have a cluster of Identity Manager nodes, stop all but one Identity Manager node.

3. Click Environments.

The Identity Manager environments screen appears with a list of Identity Manager environments.

4. Click the name of the Identity Manager environment to modify.

The Identity Environment Properties screen appears and displays the following properties:

OID

Defines a unique identifier for the environment. Identity Manager generates this identifier when you create an Identity Manager environment.

You use the OID when you configure task removal from a task persistence database. See the *Installation Guide*.

Name

Specifies the unique name of the Identity Manager environment.

Description

Provides a description of the Identity Manager environment.

Identity Manager Directory

Specifies the Identity Manager directory with which the environment is associated.

Provisioning Server

Specifies the Provisioning directory used as the provisioning user store.

Click the right arrow button to configure the provisioning directory in the Provisioning Properties page.

Version

Defines the version number of Identity Manager.

Base URL

Specifies the portion of the Identity Manager URL that does not include the protected or public alias for the environment.

Identity Manager uses the base URL to form the Redirect URL to point to the Password Services task in the default password policy for the environment.

Protected Alias

Defines the name appended to the base URL for accessing protected tasks in the User Console for an Identity Manager environment.

Public Alias

Defines the name appended to the base URL for accessing public tasks, such as the self-registration and forgotten password tasks.

Public User

Defines the user account that Identity Manager uses in place of user-supplied credentials to access public tasks.

Job Timeout

Determines the amount of time that Identity Manager waits after a task is submitted before displaying a status message.

This value is set in the User Console page in Advanced Settings.

Status

Stops or restarts the Identity Manager environment.

Merge Task Persistence Data from Identity Manager 8.1

Migrates data from an Identity Manager 8.1x task persistence database to an Identity Manager r12 task persistence database.

For more information, see the *Installation Guide*.

Note: The Merge Task Persistence Data from Identity Manager 8.1 button is only visible in environments that were created in previous versions of Identity Manager and migrated to Identity Manager r12.

5. Modify the description, base URL, or protected or public alias, as needed.
6. If you modified any environment properties, restart the Identity Manager environment.
7. If you stopped any Policy Servers in Step 1, restart them now.

Export an Identity Manager Environment

To deploy an Identity Manager environment on a production system, you export the environment from a development or staging system and import that environment to the production system.

To export an Identity Manager environment

1. Click Environments in the Management Console.
The Identity Manager environments screen appears with a list of Identity Manager environments.
2. Select the environment that you want to export.
3. Click the Export button.
A File Download screen appears.

4. Save the ZIP file to a location that is accessible to the production system.
5. Click Finish.

The environment information is exported to a ZIP file that you can import into another environment.

Import an Identity Manager Environment

In an upgrade scenario, you can import a previously existing Identity Manager environment after the upgrade.

To import an Identity Manager environment

1. Click Environments in the Management Console.
The Identity Manager environments screen appears with a list of Identity Manager Environments.
2. Click the Import button.
The Import Environment screen appears.
3. Browse for the ZIP file required to import an environment.
Note: This ZIP file is created when you export an environment. For more information on exporting environments, see the upgrade section of the *Installation Guide*.
4. Click Finish.

The environment is imported into Identity Manager.

Restart an Identity Manager Environment

To start an Identity Manager environment

1. Click Environments in the Management Console.
The Identity Manager environments screen appears with a list of Identity Manager environments.
2. Click the name of the Identity Manager environment to start.
The Identity Environment Properties screen appears.

3. Select one of the following options:

Restart

Stops and starts an environment

Stop

Stops an environment that is currently running

Start

Starts the environment after you stop it

Delete an Identity Manager Environment

Use this procedure to remove an Identity Manager environment.

Note: If Identity Manager integrates with SiteMinder for advanced authentication, Identity Manager also deletes the SiteMinder policy domain that protects the environment and the default authentication schemes created for the environment.

To delete an Identity Manager Environment

1. In the Environments screen, select the check box for the Identity Manager Environments to delete.
2. Click Delete.
Identity Manager displays a confirmation message.
3. Click OK to confirm the deletion.

Optimize Policy Rule Evaluation

Policy rules, which dynamically identify a set of users, are used in the evaluation of role member, admin, and owner policies, and identity policies. The evaluation of these rules can take significant time in large Identity Manager implementations.

Note: For more information about member, admin, owner, and identity policies, see the *Administration Guide*.

To reduce the evaluation time for rules that include user-attributes, you can enable the in-memory evaluation option. When the in-memory evaluation option is enabled, Identity Manager retrieves information about a user to be evaluated from the user store and stores a representation of that user in memory. Identity Manager uses the in-memory representation to compare attribute values against policy rules. This limits the number of calls Identity Manager makes directly to the user store.

You enable the in-memory evaluation option for an environment in the Management Console.

To enable the in-memory evaluation option

1. Open the Management Console.
2. Select Environments, *Environment Name*, Advanced Settings, Miscellaneous.
The User Defined Properties page opens.
3. Enter the following text in the Property field:
UseInMemoryEvaluation
4. Enter *one* of the following numbers in the Value field:
0
In-memory evaluation is disabled.
1
In-memory evaluation is enabled. When this option is specified, the attribute comparison is case-sensitive.
3
In-memory evaluation is enabled. When this option is specified, the attribute comparison is not case-sensitive.
5. Click Add.
Identity Manager adds the new property to the list of existing properties for the environment.
6. Click Save.

Role and Task Settings

From the Role and Task Settings screen in the Management Console, you can import or export screen, tab, role and task settings in an XML file. You can use the .XML file to apply these settings to multiple environments. To do this, you must perform the following:

- Configure screen, tab, task and role settings in one environment
- Export these settings to an XML file
- Import the XML file to the required environment

Export Role and Task Settings

Perform the following procedure to export role and task settings.

To export role and task settings

1. In the Management Console, click Environments.
A list of Identity Manager environments appears.
2. Click the name of the appropriate Identity Manager environment.
The Properties screen for that environment appears.
3. Click Role and Task Settings, and click Export.
4. Click Open to view the file in a browser window or Save to save the settings in an XML file.

Import Role and Task Settings

Perform the following procedure to import role and task settings.

To import role and task settings

1. In the Management Console, click Environments.
A list of Identity Manager environments appears.
2. Click the name of the Identity Manager environment where you want to import the role and task settings.
The Properties screen for that environment appears.
3. Click Role and Task Settings, and click Import.

4. Type the path and file name for the role definitions file to import or browse for the file.

5. Click Finish.

The status is displayed in the Role Configuration Output window.

6. Click Continue to exit.

Modify the System Manager Account

A system manager is responsible for setting up and maintaining an Identity Manager environment. Typically, a system manager's tasks include the following:

- Creating and managing the initial environment
- Creating and modifying admin roles
- Creating and modifying other administrator accounts

You create a system manager account when you create an Identity Manager environment. If this account is "locked out," for example, if the system manager forgets his or her password—you can re-create the account using the System Manager wizard.

The System Manager wizard guides you through the steps to assign a system management role to a user.

Note the following before modifying the System Manager account:

- If you are using an LDAP user store, and you configured a user container such as `ou=People`, in the directory configuration file (`directory.xml`) for your Identity Manager directory, ensure that the users you select when you configure the system manager exist in that container. Selecting a user account that does not exist in the user container may cause failures.
- When the Identity Manager environment manages a user directory with a flat or flat user structure, the profile for the user that you select must include the organization to which the user belongs. To ensure that the user's profile is configured correctly, add the name of the user's organization to the physical attribute that corresponds to the `%ORG_MEMBERSHIP%` well-known attribute in the [directory.xml file](#) (see page 82). For example, when the physical attribute description is mapped to the `%ORG_MEMBERSHIP%` well-known attribute in the `directory.xml` file, and the user belongs to the Employees organization, the user's profile should contain the attribute/value pair `description=Employees`.

To specify the system manager

1. At the Identity Manager environments screen, click the name of the appropriate Identity Manager environment.
That environment's properties screen appears.
2. Click System Manager.
The System Manager wizard appears.
3. Type the unique name for the user that will have the System Manager role as follows:
 - For relational database users, type the unique identifier for the user or the value that is mapped to the %USER_ID% well-known attribute in the directory configuration file.
 - For LDAP users, type the relative DN of the user. For example, if the user's DN is uid=Admin1, ou=People, ou=Employees, ou=NeteAuto, type Admin1.

Note: The System Manager should *not* be the same user as the administrator of the user store.

4. Click Validate to display the user's full identifier.
5. Click Next.
6. In the second page of the wizard, select a role to assign to the user as follows:
 - If you want to assign the System Manager role, do the following:
 - a. Select the radio button next to System Manager role.
 - b. Click Finish.
 - If you want to assign a role other than the System Manager role, do the following:
 - a. Select a condition in the first list.
 - b. Type a partial or complete role name or an asterisk (*) in the second list box. Click Search.
 - c. Select the role to assign from the search results list.
 - d. Click Finish.

The System Manager Configuration Output screen displays status information.

7. Click Continue to close the System Manager wizard.

Access the Status of an Identity Manager Environment

Identity Manager includes a status page that you can use to verify the following:

- The Identity Manager directory is loaded correctly
- Identity Manager can connect to the user store
- The Identity Manager environment loads correctly

To access the status page, type the following URL in a browser:

`http://domain_name/idm/status.jsp`

domain_name

Determines the fully qualified domain name of the server where Identity Manager is installed, for example, myserver.mycompany.com.

If the Identity Manager environment starts correctly and all of the connections are running successfully, the status page resembles the following illustration:

Environment	Directory	Status
test1	Admin	OK
test2	NeteAuto	OK

Troubleshooting Identity Manager Environments

The following table describes possible error messages and the troubleshooting process:

Message	Description	Troubleshooting
Not Loaded	The Identity Manager Directory associated with the environment was not loaded when Identity Manager started.	1. Verify that the user store is running. If Identity Manager integrates with SiteMinder, verify that SiteMinder can connect to the

Message	Description	Troubleshooting
Not OK	Identity Manager cannot connect to the Identity Manager Directory.	user store. In the Policy Server user interface, you can verify the connection by opening the properties page for the SiteMinder User Directory connection associated with the user store, and clicking the View Contents button. If you can see the contents of the user store, SiteMinder can connect successfully. For more information about the Policy Server, see the CA <i>SiteMinder Web Access Manager Policy Server Configuration Guide</i>. 2. Restart Identity Manager and the Policy Server.
SM connection is not OK	Identity Manager cannot connect to the SiteMinder Policy Server (for implementations that include SiteMinder)	1. Verify the following: ■ The Policy Server is running. ■ The Web agent is protecting resources. You can verify that the Web agent is running correctly by accessing the Policy Server user interface. If you are prompted for credentials, the Web agent is functioning correctly. 2. Restart Identity Manager and the Policy Server.

Message	Description	Troubleshooting
IMS is not available now	An error has occurred in Identity Manager.	Check the application server log for error details.
Windows 500 error message	The status page is not displayed when it is accessed while removing the connectivity with the LDAP user directory.	Set the Internet browser option "Show friendly error message" to off to view the status page.

Chapter 8: Advanced Settings

The Advanced Settings window in the Management Console lets you do the following:

- Access screens for configuring advanced settings
- Import and export advanced settings as described in [Import/Export Custom Settings](#) (see page 182)

This section contains the following topics:

[Auditing Logs](#) (see page 169)
[Business Logic Task Handlers](#) (see page 170)
[Email Notifications](#) (see page 170)
[Event Listeners](#) (see page 171)
[Identity Policies](#) (see page 171)
[Logical Attributes Handlers](#) (see page 172)
[Miscellaneous Properties](#) (see page 172)
[Notification Rules](#) (see page 173)
[Organization Selectors](#) (see page 173)
[Provisioning](#) (see page 174)
[User Console](#) (see page 177)
[Web Services](#) (see page 178)
[Workflow Properties](#) (see page 179)
[Work Item Delegation](#) (see page 181)
[Workflow Participant Resolvers](#) (see page 181)
[Import/Export Custom Settings](#) (see page 182)

Auditing Logs

Auditing logs maintain records of the operations performed in an Identity Manager environment. You can use the data in audit logs to monitor system activity.

Identity Manager audits *events*. An event is an operation that is generated by an Identity Manager task. A task can generate multiple events. For example, the CreateUser task may generate the CreateUserEvent and the AddToGroupEvent events.

To control the type and amount of event information that Identity Manager records, you can do the following:

- Enable auditing for Identity Manager admin tasks.
- Enable auditing for some or all of the Identity Manager events generated by admin tasks.

- Record event information at specific states, for example, when an event completes or is cancelled.
- Log information about attributes involved in an event. For example, you can log attributes that change during a `ModifyUserEvent`.
- Set the audit level for events and attributes.

For a description of auditing in Identity Manager, see [Audit Data](#) (see page 229). For information about importing or exporting audit settings, open the online help for the Auditing screen in the Management Console.

Business Logic Task Handlers

A Business Logic Task Handler performs custom business logic before an Identity Manager task is submitted for processing. Typically, the custom business logic validates data. For example, a business logic task handler may check a group's membership limit before Identity Manager adds a new member to the group. When the group membership limit is reached, the business logic task handler displays a message informing the group administrator that the new member could not be added.

You can use predefined business logic task handlers or create custom handlers using the Business Logic Task Handler API.

Note: For information about creating custom business logic, see *Programming Guide for Java*.

The Business Logic Task Handlers screen contains a list of existing global business logic task handlers. The list includes predefined handlers shipped with Identity Manager and any custom handlers defined at your site. Identity Manager executes the handlers in the order in which they appear in this list.

Global business logic task handlers can be implemented only in Java.

Email Notifications

Identity Manager can send email notifications when a task or event completes, or when an event under workflow control reaches a specific state. For example, an email can inform an approver that an event requires approval.

To specify the content of email notifications, you can use predefined email templates or customize the templates to suit your needs.

Using the Management Console, you can do the following:

- Enable email notifications for an Identity Manager environment
- Specify the template sets for creating email messages
- Indicate the events and tasks for which email notifications are sent

Event Listeners

An Identity Manager task is made up of one or more actions, named events that Identity Manager performs during the execution of the task. For example, the Create User task may include the following events:

- CreateUserEvent—Creates a user profile in an organization
- AddToGroupEvent—(Optional) Adds the user as a member of a group
- AssignAccessRole—(Optional) Assigns an access role to the user

An *event listener* "listens" for a specific event, and then performs custom business logic at a specific point in the event's lifecycle. For example, after a new user is created in Identity Manager, an event listener may add the user's information to a database used by another application.

Note: For more information on configuring event listeners, see the *Programming Guide for Java*.

Identity Policies

An identity policy applies a set of business changes to users who meet certain rules or conditions. You can use identity policies to do the following:

- Automate certain identity management tasks, such as assigning roles and group membership, allocating resources, or modifying user profile attributes
- Enforce segregation of duties. For example, you can create an identity policy that prohibits members of the Check Signer role from having the Check Approver role.
- Enforce compliance. For example, you can audit users who have a certain title and make more than \$100,000.

You create and manage identity policy sets in the User Console. For more information on identity policies, see the *Administration Guide*.

Before you use identity policies, use the Management Console to do the following:

- Enable identity policies for an Identity Manager environment
- Set the recursion level (optional)

Logical Attributes Handlers

Identity Manager logical attributes let you display user store attributes (named *physical attributes*) in a user-friendly format on task screens. Identity Manager administrators use task screens to perform functions in Identity Manager.

Logical attributes do not exist in a user store. Typically, they represent one or more physical attributes to simplify presentation. For example, the logical attribute *date* can represent the physical attributes *month*, *day*, and *year*.

Logical attributes are processed by logical attribute handlers, which are Java objects that are written using the Logical Attribute API. For example, when a task screen is displayed, a logical attribute handler might convert physical attribute data from the user store to logical attribute data, which is displayed in the task screen.

You can use predefined logical attributes and logical attribute handlers included with Identity Manager or create new ones using the Logical Attribute API.

Note: For more information, see the *Programming Guide for Java*.

Miscellaneous Properties

User-defined properties defined on this screen apply to the entire Identity Manager environment. They are passed as name/value pairs to the `init()` method of every custom Java object that you create with the Identity Manager APIs. A custom object can use this data in any way that the object's business logic requires.

When any user-defined properties are also defined for a particular custom object, for example, when user-defined properties are defined in the Properties screen for an event listener named `MyListener`, the object-specific user-defined properties and the environment-wide properties defined in the Miscellaneous screens are passed in a single call to `MyListener.init()`.

To add a user-defined property, specify a property name and value, and click Add.

To delete one or more user-defined properties, select the check box next to each name/value pair to delete, and click Delete.

When finished making changes, click Save. You need to restart the application server for the changes to take effect.

Note: All miscellaneous properties are case-sensitive. Therefore, if you define a property named SelfRegistrationLogoutUrl and another property named selfregistrationlogouturl, both properties are added.

Notification Rules

A notification rule determines which users receive email notification when a task completes or when an event in a task reaches a certain state, such as pending approval, approved, or rejected.

Note: For more information about the email notification feature, see the *Administration Guide*.

Identity Manager includes the following predefined notification rules:

ADMIN_ADAPTER

Sends an email message to the administrator who initiates the task

USER_ADAPTER

Sends an email message to the user affected by the task

USER_MANAGER

Sends an email to the manager of the user in the current context

To create custom notification rules, use the Notification Rule API.

Note: For more information about notification rules, see the *Programming Guide for Java*.

Organization Selectors

An organization selector is a custom logical attribute handler that determines where Identity Manager creates the profile of a self-registered user, based on information the user provides during registration. For example, the profile for users who provide a promotional code when they register may be added to a Promotional Users organization.

Provisioning

Use this screen when you are using Identity Manager with provisioning.

Note: For full details on setting up provisioning for an Identity Manager environment, see the *Provisioning Guide*.

The Provisioning Properties options are as follows:

Enabled

Specifies the use of two user stores, one for Identity Manager and one (called the Provisioning Directory) for provisioning accounts.

Use Session Pool

Enables the use of a session pool.

Session Pool Initial Sessions

Defines the minimum number of sessions that are available in the pool at startup.

Default: 8

Session Pool Maximum Sessions

Defines the maximum number of sessions in the pool.

Default: 32

Enable Password Changes from Endpoint Accounts

Defines the setting for the Enable Password Synchronization Agent for each user in the Provisioning Server. This will allow password synchronization between Identity Manager users and associated endpoint accounts.

Enable Accumulation of Provisioning Role Membership Events

Enables Identity Manager to sequence the provisioning role membership events, `AssignProvisioningRoleEvent` and `RevokeProvisioningRoleEvent`, so that Add operations (`AssignProvisioningRoleEvent`) are executed before Remove operations (`RevokeProvisioningRoleEvent`).

Some endpoints require this execution sequence.

When this option is selected, Identity Manager accumulates the Add and Remove operations into a single event, `AccumulatedProvisioningRolesEvent`. When this event executes, all of the Add actions are combined into a single operation and sent to the Provisioning Server as a single set of changes. Once these complete, all of the Remove operations are combined into a single operations and also sent to the Provisioning Server as a single set of changes.

Organization for Creating Inbound Users

Defines the fully qualified path to the user store used by Identity Manager. This field appears only when the user store includes an organization.

Inbound Administrator

Defines an Identity Manager administrator account that can execute tasks mapped to inbound mappings. These tasks are included in the Provisioning Synchronization Manager role. The administrator must be able to execute each task on any Identity Manager user.

Provisioning Directory

The Provisioning Directory is a repository for provisioning information including domain, global users, endpoint types, and endpoints. When you select it, other options appear for mapping the Identity Manager user store to the Provisioning Directory.

Enable Session Pooling

To improve performance, Identity Manager can pre-allocate a number of sessions to be pooled when communicating with the Provisioning Server.

If the Session Pools option is disabled, Identity Manager will create and destroy sessions as needed.

For a new environment, Session Pools are enabled by default. For existing environments, you can enable Session Pools.

To enable Session Pooling

1. In the Management Console, choose Advanced Settings, Provisioning.
2. Select Use Session Pool.
3. Define the minimum number of sessions in the pool at startup.
4. Define the maximum number of sessions in the pool.
5. Click Save.
6. Restart the Application Server.

The Session Pool is enabled per the defined settings.

Enable Password Synchronization

The Provisioning Server allows password synchronization between Identity Manager users and associated endpoint user accounts. This means when a user who has provisioning roles is created or modified in Identity Manager, the provisioning user will be set to allow password changes from endpoint accounts.

Note: When you enable this feature in the Management Console, *all* users in the environment will be set to allow password changes from endpoint accounts.

To enable password synchronization

1. In the Management Console, choose Advanced Settings, Provisioning.
2. Check Enable Password Changes from Endpoint Accounts.
3. Click Save.
4. Restart the Application Server.

Attribute Mappings

Attribute mappings associate the user attributes in provisioning-related admin tasks, such as Provision Create User, with the corresponding attributes in the Provisioning Server.

Default mappings exist for the attributes in the default tasks, which are listed in the Inbound Mappings section. If you modify one of these admin tasks, so that different attributes are used, update the attribute mappings as needed.

Inbound Mappings

Inbound mappings map events, generated by the Provisioning Server, to an admin task. These mappings are preset and cannot be modified.

Outbound Mappings

Outbound Mappings associate events, which are generated by admin tasks, with events that are applied to the Provisioning Directory. Default mappings exist for the events that affect user attributes.

User Console

You access an Identity Manager environment using the User Console, a web application that allows users to perform admin tasks. You define certain properties for an environment's user console in the User Console page in the Management Console.

The User Console page includes the following fields:

General Properties

Define properties that apply to an Identity Manager environment.

Show Recently Completed Tasks

Determines whether Identity Manager displays a status message when a task completes.

When this option is selected, users must click OK to clear the status message that Identity Manager displays.

To disable the message and prevent users from having to click OK when each status message appears, deselect this option.

Job Timeout

Determines the amount of time that Identity Manager waits after a task is submitted before displaying a status message.

When the task completes within the specified amount of time, Identity Manager displays the following message:

"Task has been submitted for processing on *current date*"

If the task takes longer to complete or is under workflow control, Identity Manager displays the following message:

Task has been submitted for processing on *current date*

Note: Changes may not take effect immediately.

Theme Properties

Let you customize the icon and title of the User Console in an Identity Manager environment. For example, you can add a company logo and the company name to User Console screens.

Theme properties include the following settings:

Icon (URI)

Defines the icon using a URI to an image available to the application server.

Example: `http://myserver.mycompany.com/images/front/logo.gif`

Title

Specifies custom text, which is displayed next to the icon at the top of the User Console.

Note: If you defined a custom skin, you can specify an icon or title by referencing a skin's properties file. For example, if the entry for the icon image in the .properties file for a custom skin is `image/logo.gif`, you can enter that same string in the Icon field.

Login Properties

Specify the authentication method and location of the login page to which users are directed when they access an Identity Manager environment.

Authentication Provider module class name

Specifies the class name of the authentication provider module.

Login Page

Specifies the page to which users are directed when they access an Identity Manager environment.

Web Services

The Identity Manager Task Execution Web Service (TEWS) enables third-party client applications to remotely submit Identity Manager tasks to Identity Manager for execution.

The Web Services Properties screen lets you configure the TEWS for an environment. On this screen, you can do the following:

- Enable TEWS for an Identity Manager environment
- Generate task-specific Web Services Definition Language (WSDL) documents
- Configure SiteMinder to secure the web services URL, if Identity Manager integrates with SiteMinder

For information on issuing remote requests to Identity Manager through the Task Execution Web Service, see the *Programming Guide for Java*.

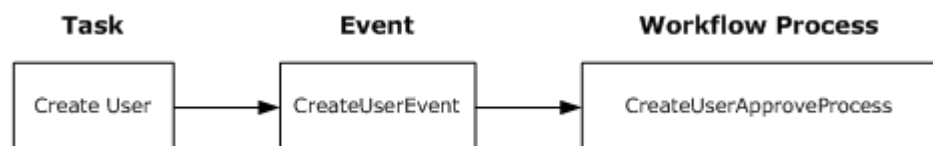
Workflow Properties

If enabled, the workflow feature controls the execution of an Identity Manager task that is associated with a workflow process.

A workflow process is a set of steps that are performed to accomplish a business objective, such as creating a new user account. Typically, one of these steps involves approving or rejecting the task.

An Identity Manager task is associated with one or more events, which may trigger one or more workflow processes. After the workflow processes complete, Identity Manager performs or rejects the task based on the results of the workflow processes.

The following illustration shows the relationship between an Identity Manager task, an associated event, and a workflow process:



Workflow Properties

Use the check box to enable or disable workflow for the Identity Manager environment.

Event Mappings

Use this section to map Identity Manager events to workflow processes.

Important: These are global mappings. The mapped workflow process executes whenever the corresponding event is generated by any task in the environment.

Note: For more information about workflows and event mappings, see the *Administration Guide*.

Global Process to Event Mapping

This table shows the default global workflow process and event mappings, specified in the Management Console.

Important: These are global mappings. The mapped workflow process executes whenever the corresponding event is generated by any task in the environment.

Workflow Process	Mapped Event
CertifyRoleApproveProcess	CertifyRoleEvent
CreateGroupApproveProcess	CreateGroupEvent
CreateOrganizationApproveProcess	CreateOrganizationEvent
CreateUserApproveProcess	CreateUserEvent
DeleteGroupApproveProcess	DeleteGroupEvent
DeleteOrganizationApproveProcess	DeleteOrganizationEvent
DeleteUserApproveProcess	DeleteUserEvent
ModifyAccessRoleMembershipApproveProcess	AssignAccessRoleEvent RevokeAccessRoleEvent
ModifyAdminRoleMembershipApproveProcess*	
ModifyGroupMembershipApproveProcess*	
ModifyIdentityPolicySetApproveProcess	ModifyIdentityPolicySetEvent
ModifyOrganizationApproveProcess	ModifyOrganizationEvent
ModifyUserApproveProcess	ModifyUserEvent
SelfRegistrationApproveProcess	SelfRegisterUserEvent

Note: Workflow processes marked with an asterisk (*) are not mapped to events by default.

Work Item Delegation

If enabled, work item delegation allows a participant (the delegator) to specify that another user (the delegate) be allowed to approve tasks in the delegator's work list. A participant can assign work items to another approver during periods when the delegator is "out of the office." Delegators retain full access to their work items during the delegation period.

Delegation uses the following wellknown attribute:

%DELEGATORS%

This wellknown attribute stores the names of users who are delegating to the user with the attribute, as well as the time when the delegation was created.

Note: For more information, see the *Administration Guide*.

Workflow Participant Resolvers

The activities in a workflow process, such as approving or rejecting a task, are performed by *participants*.

You use the Workflow Participant Resolvers screen to map a custom participant resolver to a fully qualified participant resolver Java class.

A custom *participant resolver* is a Java object that determines a workflow activity's participants and returns a list to Identity Manager. Identity Manager then passes the list to the workflow engine.

Typically, you write a custom participant resolver only if none of the standard participant resolvers can provide the list of participants that an activity requires.

Note: For information about developing custom participant resolvers, see the *Programming Guide for Java*. For information about standard participant resolvers, see the *Administration Guide*.

Import/Export Custom Settings

From the Advanced Settings screen in the Management Console, you can apply advanced settings to multiple environments, as follows:

- Configure advanced settings in one environment.
- Export the advanced settings to an XML file.
- Import the XML file to the required environments.

Chapter 9: Customizing the User Console

This section contains the following topics:

[Skins](#) (see page 183)

[Logout Pages](#) (see page 197)

[Custom Help](#) (see page 199)

Skins

Identity Manager users perform most admin tasks in a user console. Identity Manager includes the following user consoles:

- **ca12**—The default user console
- **imcss**—An alternate user console that you can customize for your business needs

The console name is added to the URL for each console page. For example, the following URLs are for the Create User task in the ca12 and imcss consoles, respectively:

- `http://myserver.ca.com:8080/idm/neteauto/ca12/index.jsp?task.tag=CreateUser`
- `http://myserver.ca.com:8080/idm/neteauto/imcss/index.jsp?task.tag=CreateUser`

Note: Identity Manager generates these URLs automatically, once an administrator logs into the user console.

You can change the appearance of a console with skins. A *skin* is a set of components that you can use to customize the appearance of the user interface (UI) for a given audience.

You can change the following aspects:

- Images
- Colors and fonts
- Page headers and footers

In an Identity Manager environment, you can have more than one skin, and you can create different skins for different audiences. For example, employees may see one skin and partners may see another skin.

Components of a Skin

The components of a skin are as follows:

Cascading Style Sheet

Defines the CSS files that contain color and font definitions.

Location:

*IdentityMinder.ear/user_console_war/app/imcss/skin_name/fileName.css **

Images

Defines a graphic file that can be displayed on a web page, such as a .gif, .jpeg, or .png file.

Location:

*IdentityMinder.ear/user_console_war/app/imcss/skin_name/image/**

.properties file

Defines the text file that defines the components for the skin.

Location:

IdentityMinder.ear/user_console_war/app/imcss/skin_name/

IdentityMinder.ear

Defines one of the following installed locations of the IdentityMinder.ear directory:

- WebLogic:

*WebLogic_home\bea\user_projects\domains\Identity Manager
Domain name\applications\IdentityMinder.ear*

- JBoss:

`JBoss_home\jboss-version\server\default\deploy\IdentityMinder.ear\`

- WebSphere:

`was_im_tools_dir\WebSphere-ear\IdentityMinder.ear\`

`was_im_tools_dir`

Defines the installed location of the Identity Manager Tools for WebSphere.

Note: After making changes in the WebSphere-ear\IdentityMinder.ear directory, you must package the directory into an Enterprise Archive (EAR) file and install the EAR file in the WebSphere Application Server. For more information, see the *Installation Guide*.

Identity Manager Skins

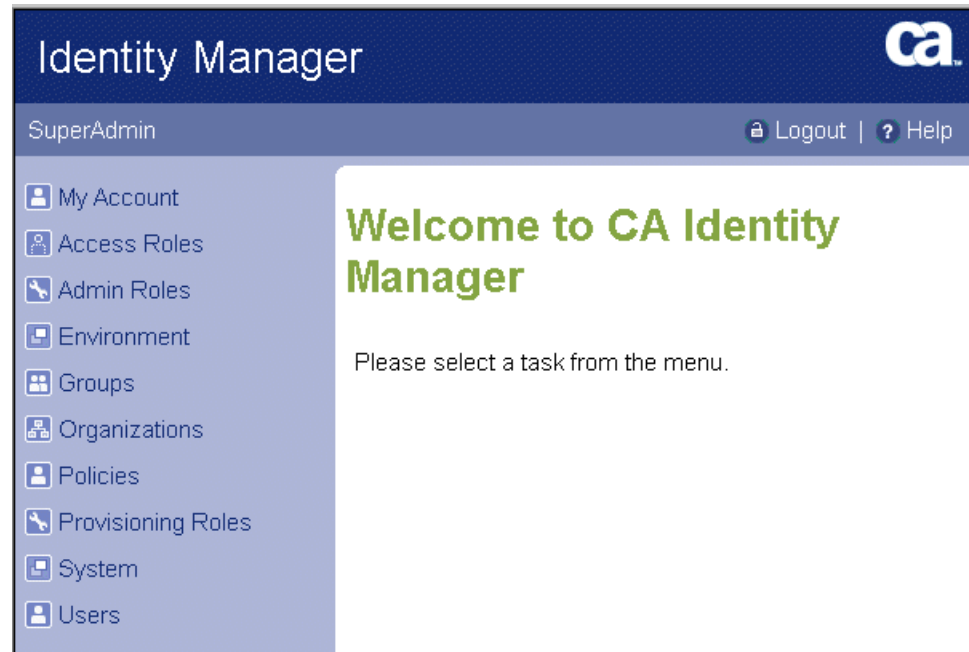
Identity Manager includes the following skins:

- `idm`—Defines the components of the default `idm` User Console. Skin components that are not defined by custom skins default to the `idm` components. For example, if a skin does not include a custom CSS file, Identity Manager uses the default CSS file from the `idm` skin.

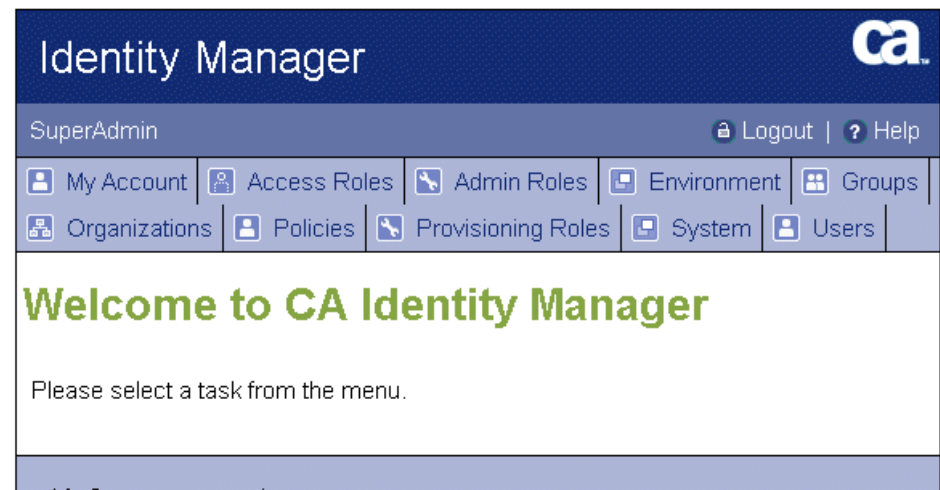
This skin is installed in the following folder:

IdentityMinder.ear/user_console_war/app/imcsss/idm

The following illustration shows the *idm* User Console:



- horizontal—Defines the components of the *horizontal* User Console, as shown in the following illustration:

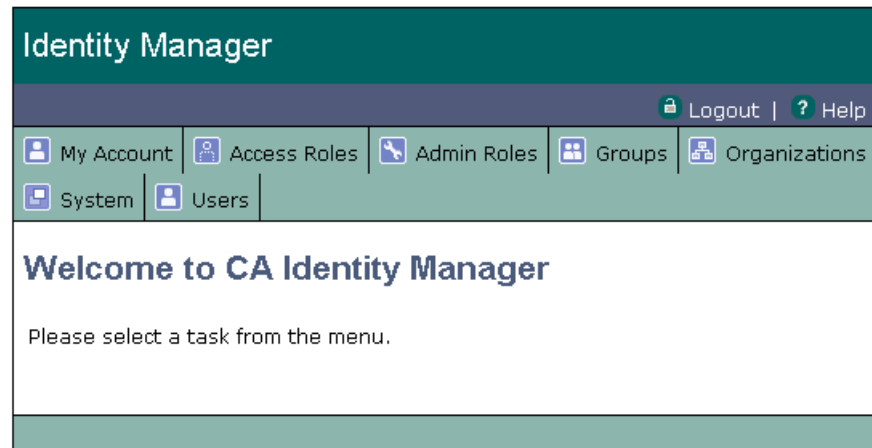


This interface looks similar to the *idm* skin but displays menu options in a horizontal format.

This skin is installed in the following folder:

IdentityMinder.ear/user_console_war/app/imcss/horizontal

- horizontal2—Defines the components of the *horizontal2* User Console, as shown in the following illustration:

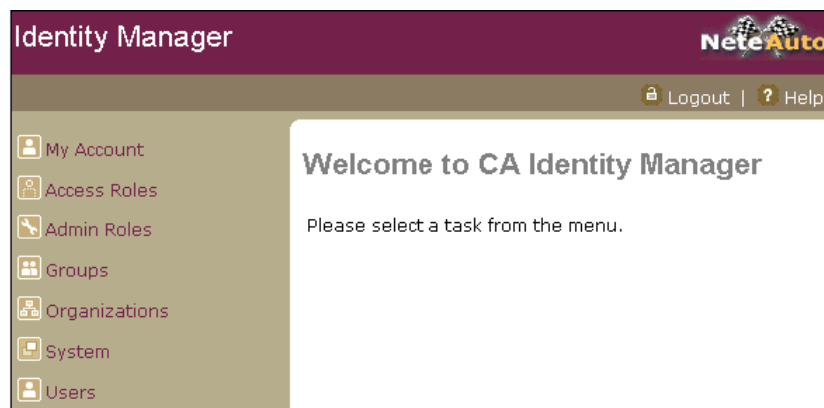


This interface looks similar to the *horizontal* skin but displays green and purple colors in the same horizontal format.

This skin is installed in the following folder:

IdentityMinder.ear/user_console_war/app/imcss/horizontal2

- neteauto—Defines the components of the *neteauto* User Console, as shown in the following illustration:



This interface has the same layout as the *idm* skin but displays red and light brown colors and the NeteAuto image.

This skin is installed in the following folder:

IdentityMinder.ear/user_console_war/app/imcss/neteauto

This chapter uses the *neteauto* skin for demonstration purposes.

Note: The skins in the various illustrations apply to the imcss console.

Skin Associations

A skin can inherit properties from a *parent skin*. Therefore, you can define a component once and reuse it in multiple skins. For example, a company logo, defined in a parent skin, can propagate to all the company skins.

When Identity Manager renders a user console, it checks for the appropriate skin for the user. When a parent skin is specified, its components are added.

For example, suppose the *corporate* skin includes only the company logo. A *sales* skin includes a custom footer JSP and points to the parent skin, *corporate*. When salespersons access Identity Manager, they see the footer defined in the *sales* skin, and the logo, colors, and fonts defined in the *corporate* skin.

How to Create a Skin

Creating a skin involves the following steps:

1. Copying an existing skin
2. Updating the skin's images
3. Updating the skin's colors and fonts
4. Modifying the .properties file
5. Modifying the index.jsp file

Copy an Existing Skin

Perform the following procedure to copy an existing skin.

To copy an existing skin

1. Look at the illustrations in Identity Manager Skins, and decide which of the following skins is similar to the one that you want to create for your company:

- *idm* skin
- *horizontal* skin
- *horizontal2* skin
- *neteauto* skin

For illustrative purposes, this section assumes that you chose the look of the *neteauto* skin and will call it *yourskin*. The name *yourskin* is the name of your company's skin.

2. Copy the following file:

IdentityMinder.ear/user_console_war/app/imcss/neteauto

Give it the following name:

IdentityMinder.ear/user_console_war/app/imcss/yourskin

3. In the *yourskin* directory, rename the files in the following table:

Current Name	New Name
<i>neteauto.css</i>	<i>yourskin.css</i>
<i>neteauto_task.css</i>	<i>yourskin_task.css</i>
<i>neteauto.properties</i>	<i>yourskin.properties</i>

To simplify administration, the name of the *.properties* file and the *.css* files should correspond to the name of the skin.

Note: Skin names are case-sensitive.

Update the Skin Images

Update any of the images (*.gif*, *.jpg*, or *.png*) in the following directories with the images that match the look of your company's skin:

- *IdentityMinder.ear/user_console_war/app/imcss/yourskin/image*
- *IdentityMinder.ear/user_console_war/app/imcss/yourskin/image/category*
- *IdentityMinder.ear/user_console_war/app/imcss/yourskin/image/tab*
- *IdentityMinder.ear/user_console_war/app/imcss/yourskin/image/tasks*

For example, to put your company's logo in the *yourskin* skin, replace `logo.jpg` from the image directory with your company's logo.

We recommend using the same image file names because many of these images are already mapped with the appropriate paths in the `yourskin.properties` file.

Update the Skin Colors and Fonts

Update the *yourskin* skin's cascading style sheets with the fonts and menu, background, margin, and other colors of your choice.

The paths to the style sheets are as follows:

- `IdentityMinder.ear/user_console_war/app/imcss/yourskin/yourskin.css`
- `IdentityMinder.ear/user_console_war/app/imcss/yourskin/yourskin_task.css`

Modify the .properties File

Perform the following procedure to modify the `.properties` file.

To modify the .properties file

1. In the `yourskin.properties` file, replace every *neteauto* reference with *yourskin*.
2. Make the following additional modifications in the `yourskin.properties` file:
 - a. If it is not already specified, set the following in the parent section:

`parent=/app/imcss/idm/idm.properties`

The parent section specifies the location to the properties file of the parent skin. In this example, the parent skin is *idm*.

- b. Set the cascading style sheets as follows:

`stylesheet/skin.css=yourskin/yourskin.css`

`stylesheet/skin_task.css=yourskin/yourskin_task.css`

- c. Proceed as follows:

If you replaced the images in the skin without renaming them, go to Step d. If you renamed images, do the following:

- In the header images section, set the path to header images from [Update the Skin Images](#) (see page 189).

You can add multiple image entries to the `.properties` file by adding an entry like the following:

`image/logo.gif=yourskin/image/logo.jpg`

- In the standard tabs section, set the path to tab images from [Update the Skin Images](#) (see page 189).
 - In the task body buttons section, set the path to task body button images from [Update the Skin Images](#) (see page 189).
- d. Save the file.

Modify the index.jsp File

Perform the following procedure to modify the index.jsp file.

To modify the index.jsp file

1. Open the index.jsp file in a text editor. This file is located in the following folder:
`<IdentityMinder.ear>/user_console_war/app/imcss`
2. Under the `<skin:update>` section, add the following as the first entry:
`<skin:skin name="yourskin" filename="/app/imcss/yourskin/yourskin.properties" />`
3. Save the file.

Identity Manager selects the default skin for the Identity Manager User Console based on which skin tag is listed first in the `<skin:update>` section of the index.jsp file.

As noted in the following `<skin:update>` section from the index.jsp file, Identity Manager will display the user console for the *yourskin* skin:

```
<skin:update>
  <skin:skin name="yourskin" filename="/app/imcss/yourskin/yourskin.properties" />
  <skin:skin name="idm" filename="/app/imcss/idm/idm.properties" />
  <skin:skin name="neteauto" filename="/app/imcss/neteauto/neteauto.properties" />
  <skin:skin name="horizontal"
    filename="/app/imcss/horizontal/horizontal.properties" />
  <skin:skin name="horizontal2"
    filename="/app/imcss/horizontal2/horizontal2.properties" />
</skin:update>
```

When you want to use a different skin, you need to change the order of the skin tags listed.

Recompile the .jsp Files and Restart the Application Server

After you have created a skin, you must recompile the .jsp files and restart the application server as follows:

1. Configure your application server to recompile all the console.jsp files. [How to Edit Console Pages](#) (see page 194) has instructions for editing console pages. For more information about recompiling .jsp files, see your application server's documentation.
2. Restart the application server so that the *yourskin* skin's changes can take effect.

How to Use Multiple Skins

You may want different sets of users to see different skins. For example, the skin for employees may include the company logo and colors. The skin for suppliers could include a different logo and color scheme, and display a company-specific footer.

Note: This functionality requires integration with SiteMinder, an optional component.

If Identity Manager integrates with SiteMinder, you can use a SiteMinder response to associate the *yourskin* skin with a set of users. The response is paired with a rule in a policy, which is associated with a set of users. When the rule fires, it triggers the response to pass information about the skin to Identity Manager, to build the user console.

The following steps are involved:

1. A user tries to access an Identity Manager environment. (An Identity Manager environment is protected by SiteMinder.)
2. SiteMinder authenticates and authorizes the user. The rule in the policy fires, triggering a response that contains information about the skin.
3. The response is passed as an HTTP header variable to Identity Manager through the Web agent.
4. Identity Manager builds the User Console based on the skin.

Prioritize Skins

Multiple policies (and therefore multiple skins) may apply to a single user. To make sure that the user sees the correct skin, assign priority numbers to skins.

When you create a skin response, add the priority number to the header variable name. The priority number can be from 1 to 999, with 1 having the highest priority.

For example, the header variable name in the response for the *sales* skin is *skin3*. For the *management* skin, the header variable name is *skin2*. When a user logs in to Identity Manager as a sales manager, he or she will see the management skin because it has a higher priority.

Create a Skin Response

Perform the following procedure to create a skin response.

To create a skin response

1. In the Domains tab in the Policy Server User Interface, expand the policy domain for the Identity Manager environment to which the skin applies. The domain name has the following format:

*environment_name*Domain

environment_name

Defines the name of the Identity Manager environment.

For example, when the Identity Manager environment name is *employees*, the domain name is *employeesDomain*.

Note: The policy domain must be associated with the user directory that contains the Identity Manager users who will use the skin.

2. Under the domain, expand Realms. You see one of the following realms depending on the features you configured for the environment:
 - *environment_name_ims_realm*—Protects the Identity Manager environment.
 - *environment_name_pub_realm*—Enables support for public tasks, such as self-registration and forgotten password tasks.

This realm appears only when you have configured a public alias.

Create a rule in each of the realms. Specify the following:

- Resource: */**
- Actions: GET, POST

To simplify administration, include the skin in the rule name. For example, when the skin name is *yourskin*, create a rule named *yourskin* skin.

3. Create a response as follows:
 - a. Expand the Policy Domain from Step 1. Right-click Responses and click Create Response.
The Response Properties dialog opens.
 - b. Click Create.
The Response Attribute dialog opens.
 - c. Make sure WebAgent-HTTP-Header-Variable is selected.
 - d. Select Static in the Attribute Kind box.
 - e. In the Variable Name field, type one of the following variable names:
 - skin—Use this variable when all Identity Manager users see the same skin.
 - skinx—Use this variable if you have multiple skins. The **x** represents a priority value where 1 is the highest priority and 999 is the lowest priority.
 - f. In the Variable Value field, specify the name of the skin.
4. Create a policy in the Policy Domain from Step 1. Specify the following:
 - In the Users tab, specify users who should see the skin. For example, if members of the sales organization should see the skin, be sure to include ou=sales in the policy.
 - In the Rules tab, add the rules you created in Step 3. Associate each rule with the response you created in Step 3.

How to Edit Console Pages

To change the layout of the User Console or to add links to a page's headers or footers, you can edit the following console pages:

- [index.jsp](#) (see page 196)—Hides or displays headers and footers in the main console. Also defines the available Identity Manager skins.

The index.jsp file is located in `<IdentityMinder.ear>/user_console_war/app/imcss/`

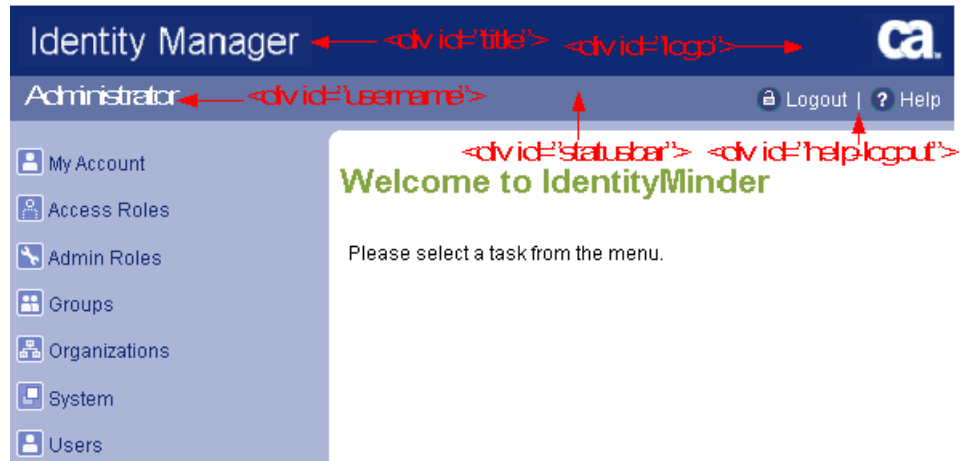
- [head.jsp](#) (see page 195)—Contains the code for the header in the User Console main page.

- home.jsp—Contains the confirmation message for a particular task.

The head.jsp and home.jsp files are located in `<IdentityMinder.ear>/user_console_war/app/imcss/component`

How to Modify the Header in the Identity Manager User Console

You use the `head.jsp` file (located in `<IdentityMinder.ear>/user_console_war/app/imcss/component`) to modify the header in the User Console. The header can contain images, links, title, colors, and fonts. The following illustration shows the `<div id>` markers in the `head.jsp` file, which generate the header for the *idm* skin:



To update the header, modify the code for the following entries in the `head.jsp` file:

- `<div id="title">`
- `<div id="logo">`
- `<div id="statusbar">`
- `<div id="username">`
- `<div id="help-logout">`

For example, to change the link users are directed to when they click on the logo image that appears in the header for the User Console, replace the URL in the following code with the URL for your company's home page.

```
<div id="logo">
<a href="http://www.yourcompany.com/"><skin:img altKey="console.logo"
titleKey="console.logo" src="logo.gif" width="93" height="33" border="0" /></a>
</div>
```

Note: After modifying the `head.jsp` file, be sure to [recompile the .jsp files](#) (see page 192).

How to Modify and Show the Footer

To modify and show the footer, you can do the following:

- Insert the footer's HTML code into the `index.jsp` file (located in *IdentityMinder.ear/user_console_war/app/imcss*).
- Modify the cascading style sheet file from one of the skins—*idm*, *horizontal*, *horizontal2*, and *neteauto*—so that Identity Manager displays the footer.

Insert the Footer's HTML into the `index.jsp` File

Perform the following procedure to insert the footer's HTML into the `index.jsp` file.

To insert the footer's HTML into the `index.jsp` file

1. Open the `index.jsp` file in a text editor.
2. Under the `<div id="foot">` section at the end of the file, add the HTML for *yourskin* skin's footer between the `<div>` and `</div>` markers.
3. Save the file.

Modify the Cascading Style Sheet to Display a Footer

Use the cascading style sheet file from each skin—*idm*, *horizontal*, *horizontal2*, and *neteauto*—to configure Identity Manager to display the footer or not. These cascading style sheet files are stored in the following folder:

IdentityMinder.ear/user_console_war/app/imcss/

By default, the footer shows in the *horizontal* and *horizontal2* skins but not in *idm* or *neteauto*.

Presently, *yourskin* skin does not display the footer because it is a copy of *neteauto*, which uses the `idm.css` file to display footers. The `neteauto.css` and `yourskin.css` files contain color overrides only and import the `idm.css` file to handle other display functions like showing footers. In the *idm* skin, shown as follows, the `#foot` section in the `idm.css` file shows the footer as being disabled:

```
#foot {  
    display: none;
```

In the horizontal skin, shown as follows, the `#foot` section in the `horizontal.css` file shows the footer as enabled:

```
#foot {  
    border: 1px solid black;  
    background-color: #AEB6D8;  
    padding: 4px;  
    display: block; /* to override parent */
```

The `display: block` reference means that Identity Manager displays the footer.

To get *yourskin* skin to display the footer, do the following:

- In `yourskin.css`, add the following:

```
/*CONSOLE FOOTER*/  
#foot {  
    display: block;  
}
```

In this section, you can also modify the footer's border, background color, and padding.

Logout Pages

A logout page is a page to which a user is directed after performing an action in certain Identity Manager task screens, such as clicking a Logout link from the User Console.

For self-service tasks, such as self-registration or forgotten password tasks, users are redirected to a logout page when they click Cancel to exit the task, or when they click OK in a confirmation or error message.

You can configure a custom logout page for the following Identity Manager screens:

- User Console
- Self-registration tasks
- Forgotten Password tasks

Important! If Identity Manager integrates with SiteMinder, configure the SiteMinder Web Agent to [terminate the user session](#) (see page 288) after the user logs out of Identity Manager. If you do not configure the Web Agent, SiteMinder may reopen the user session.

Configuring Logout Pages

To configure custom logout pages

1. Create one or more custom logout pages.

To ensure that an HTML logout page is loaded from the Web server and not from the browser's cache, set up the logout page so that it cannot be cached in the browser. For example, for HTML pages, you can add the following meta tags to the page:

```
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
```

```
<META HTTP-EQUIV="Expires" CONTENT="-1">
```

Important! Meta tags may not always work with an Internet Explorer browser. If not, use a cache-control HTTP header.

2. In the Identity Manager environments screen, click the name of the appropriate environment.

The Environment Properties screen appears.

3. Click Advanced Settings, and click Miscellaneous.

The Miscellaneous Properties screen appears.

4. In the Property field, type one of the following properties:

- **MainConsoleLogoutUrl**—Overrides the default logout URL in the main console.

This URL is also displayed for self-registration and forgotten password tasks if you do not specify custom logout pages using the *tasktagLogoutUrl* property.

- **tasktagLogoutUrl**—Specifies a logout page for a public task.

In this property, *tasktag* identifies the task for which you are configuring a custom logout page.

For example, to configure a logout page for the default self-registration task, enter the following in the Property field:

SelfRegistrationLogoutUrl

You can define multiple *tasktagLogoutUrl* properties to configure different logout pages for different tasks. For example, when you have different self-registration pages for customers and suppliers, you can define a different logout page for each task.

Note: You specify the task tag when you configure a task in the User Console. For more information, see the *Administration Guide*.

5. In the Value field, type the URL that users are redirected to at logout.
6. Click Save.

Custom Help

Identity Manager allows you to create your own custom help for tasks and tabs that you have customized in the User Console. To implement custom help, you can create a context-sensitive help system with custom HTML help files or Wiki pages and redirect help links within the Identity Manager User Console to access your custom help.

This feature also allows you to translate any of the default help (written in English) into another language.

How Customized Help Works

Identity Manager uses resource bundles to override default help files and to provide the ability to link to custom context-sensitive help.

Using the defined format, you can create a resource bundle and place it in the IdentityMinder.ear. When a user clicks on a help link, Identity Manager will search through any applicable resource bundles for matches to custom help. If there is no match to custom help, Identity Manager will provide the default help to the user.

If an international user clicks on a help link and a language-specific resource bundle has been created, Identity Manager will check the user's browser locale settings and open the language-specific help link. If there is no match within the language-specific resource bundle, the user will be directed to the default English help.

Note: Content on the custom pages will not show up when you search the default help or view the help index.

Custom Help Format

The help link resource bundle uses a key/value pair to determine which help page to direct the user to. For custom help, tags are used for Key IDs and the custom web page URL is the value. The syntax for the help link resource bundle may be one of the following:

TaskTag.PageTag=Help URL

TaskTag.@PageDefinitionTag=Help URL

A tag is the unique identifier for a task, screen, or tab. The parameters for the key/value pair are defined as follows:

Task Tag

The active task tag.

Page Tag

The active search or tab tag.

Page Definition Tag

The active search definition or tab definition tag.

Help URL

The help URL is either an absolute URL (<http://www.neteauto.com>) or a relative URL that points to content on the Identity Manager server (</idm/help/customhelp.html>)

Example:

`ModifyUser.Profile=/idm/userprofile.html`

Custom Help Expressions

The following expressions have specific meaning when used in custom help resource bundles:

Expression	Meaning
<code>*</code>	Used to match any task tag, page tag, or page definition tag.
<code>\${task}</code>	Used in the help URL. Replaced with the task tag of the active page.
<code>\${page}</code>	Used in the help URL. Replaced with the page tag of the active page.
<code>\${pagedef}</code>	Used in the help URL. Replaced with the page definition tag of the active page.

Example:

`*.*=http://www.help.com/Wiki.jsp?page=${task}_${page}`

The `$task` and `$page` expressions are replaced with the task tag and page tag for the current web page where the user clicked the help link. For example, if a user clicks the help link from the Profile tab of the Create Group task, the help URL would open the following help page:

`http://www.help.com/Wiki.jsp?page=CreateGroup_Profile`

How Help Determines Which Link To Use

When a user clicks on a help link, Identity Manager performs checks to determine which help URL to use. These checks are made for the most specific case and then become more general. The bundle with the best match to the user's locale will be used.

The order of checks for a help link are as follows:

1. [TaskTag].[PageTag]
2. [TaskTag].@[PageDefinitionTag]
3. *. [PageTag]
4. *.@[PageDefinitionTag]
5. [TaskTag].*
6. *. *

Example:

.=http://www.help.com/Wiki.jsp?page=\${task}_\${page}

This matches anything without a more specific match and includes active task and page tags in the generated help link.

Example:

ModifyUser.Profile=/idm/userprofile.html

This matches the Profile tab on the Modify User task and directs the user to the relative URL /idm/userprofile.html

Example:

*.@Profile=/idm/profile.html

This matches all tabs derived from the Profile tab definition without a more specific match and directs the users to the relative URL /idm/profile.html.

How to Customize the Help

Custom help allows you to provide specific help pages for your users when accessing highly-customized or localized Identity Manager environments.

To customize the help

1. Create custom help pages and host them on a web site.
2. Create a resource bundle with key IDs mapped to the custom help pages. Name the resource bundle as follows:
`help_EnvironmentName_languageidentifier.properties`

Note: The language identifier is an optional, two-character abbreviation for a specific language.

3. Place the .properties file in the
IdentityMinder.ear/config/com/netegrity/config directory.

Note: If you are localizing the help to more than one language, create a resource bundle for each language.

4. Restart the Identity Manager server.

Your custom help will now override the default help when your users click on the help links in the Identity Manager User Console.

For internationalized help, users with their [browser locale preferences set to another language](#) (see page 260) will be directed to the appropriate custom help.

Examples of How to Use Custom Help

This section provides some examples of how custom help can address your business needs and localization requirements.

Example: Customize the Help

A customer has deployed a Identity Manager environment with heavily customized screens for Users, Groups, and Organizations. Unfortunately, the help content is static and pertains only to the default environment. They want to write their own help content to reflect the customization in their environment.

To customize the help

1. Write custom help pages for each custom task and tab.
2. Host the pages on a web site.

3. Create a resource bundle and place the .properties file in the IdentityMinder.ear/config/com/netegrity/config directory.
4. Restart the Identity Manager server.

For example, if the environment name is neteauto, create the following resource bundle named help_neteauto.properties:

*.UserProfile=http://www.neteauto.com/imhelp/user.html

*.GroupProfile=http://www.neteauto.com/imhelp/group.html

*.OrgProfile=http://www.neteauto.com/imhelp/org.html

The help links for the neteauto environment will access the default help except for the Profile tabs in the Modify User, Modify Organization, and Modify Group tasks.

Example: Create Wiki Help

A customer has a highly-customized Identity Manager environment and would like to provide their users with access to a rich context-sensitive help system. They want to create a Wiki for their environment that is available to users through the help links.

To create Wiki help

1. Write the Wiki content for your customized environment.
2. Create a resource bundle and place the .properties file in the IdentityMinder.ear/config/com/netegrity/config directory. The resource bundle should have a single entry:

.=http://www.neteauto.com/wiki.jsp?page=\${task}-\${page}

The help links will launch the Wiki with a page specific to the task and tab.

3. (Optional) If the page has not been created yet, the user can enter the details for the task and tab.
4. Restart the Identity Manager Server.

Example: Localize the Help

A customer in Japan has purchased a localized version of Identity Manager. Unfortunately, all of the help is in English. They want to write their own help in Japanese and implement it for their User Console.

To localize the help

1. Write custom help pages for each task and tab.
2. Host the custom help pages on a web site.

3. Create a resource bundle and place the .properties file in the IdentityMinder.ear/config/com/netegrity/config directory.

For example, if the environment is named neteauto, create the following resource bundle named help_neteauto.properties:

```
ModifyUser.*=http://www.neteauto.jp/modifyuser.html
```

```
ModifyGroup.*=http://www.neteauto.jp/modifygroup.html
```

4. Restart the Identity Manager Server.

The help links in the User Console will redirect to the language-specific, custom help pages that are hosted on the www.neteauto.jp web site.

Example: Internationalize the Help

An international company has purchased Identity Manager and must support users in English, Spanish, and French. They are able to internationalize their Identity Manager environments, but the help is written in English. They want to write a version of help in each language and make the correct help available depending on what language the user requires.

To internationalize the help

1. Write custom help pages for each task and tab, and in each language.
2. Host the custom help pages on a web site.
3. Create two locale-specific resource bundles and place the .properties files in the IdentityMinder.ear/config/com/netegrity/config directory.

For example, if the environment is named neteauto, create two resource bundles such as help_neteauto_es.properties (for Spanish) and help_neteauto_fr.properties (for French).

4. Restart the Identity Manager Server.

Users with their [browser locale preferences set to Spanish or French](#) (see page 260) will be directed to the appropriate custom help in their language. All other users will be directed to the default English user help.

Chapter 10: Validation Rules

This section contains the following topics:

[Introduction](#) (see page 205)

[About Validation Rules](#) (see page 205)

[Using Default Validation Rules](#) (see page 209)

[How to Implement Custom Validation Rules](#) (see page 212)

[How to Configure Validation Rules](#) (see page 221)

[How to Initiate Validation](#) (see page 228)

[Sample Implementations](#) (see page 228)

Introduction

Values are assigned to data store attributes through task screen fields or programmatically. Attribute validation rules help ensure that the values users type in task screen fields or that are supplied programmatically meet certain requirements, as in the following example:

- User directory requirements, such as enforcing data type, or ensuring that an entry such as a date is formatted in a particular way.
- Data integrity. Does an entry make sense in the context of other information about the task screen or according to site-specific business rules?

A validation rule can be directly associated with a task screen field, or be indirectly associated with the field by being associated with a managed object attribute that is configured for the field.

All validation rules directly or indirectly associated with a task screen's fields must be satisfied before Identity Manager can begin processing the task. When a supplied value is invalid, a message associated with the violated rule is displayed, and the user can then correct the entry and resubmit the task.

About Validation Rules

Validation rules enforce requirements, as shown in the following examples:

- A Quantity field must contain only numeric characters.
- A Telephone Number field must be formatted as nnn-xxx-nnnn.
- An Employee ID field must contain a number no higher than 9999.

- The value typed in a Zip Code field must be appropriate for the values typed in the City and State field.
- Does the value typed in a Title field qualify the user for the security clearance typed in Security Level?

In addition to verifying a user entry, a validation rule can *change* an entry so that the entry conforms to the rule's requirements without further user intervention, as in the following examples:

- A validation rule for a Telephone Number field requires that telephone numbers be formatted as nnn-xxx-nnnn. If a user types the value 9785551234, the validation rule automatically changes the entry to the correct format, 978-555-1234.
- A validation rule for a Department Number field requires that the number be prefixed with a three-character code representing the name typed in the Region field. When the prefix is missing or incorrect, the validation rule supplies the correct prefix.

Changing an entry through a validation rule is named *transformation*.

Types of Validation Rules

There are two types of validation rules as follows:

- **Task-level validation.** Validates an attribute value against other attributes in the task. For example, you might verify that the area code in a user-supplied telephone number is appropriate for the user's city and state.

During task-screen configuration, task-level validation rules are directly associated with task screen fields.

You can use this type of validation to enforce data integrity.

- **Directory-level validation.** Validates the attribute value itself, and not in the context of other attributes in the task. For example, you might verify that a user-supplied telephone number matches the nnn-xxx-nnnn format used in the directory.

In `directory.xml`, directory-level validation rules are mapped to a managed object attribute through a rule set. The rules in the rule set are applied to any task screen field configured with the attribute.

You can use this type of validation to enforce user directory requirements.

Identity Manager executes task-level validation rules before directory-level validation rules.

Example: Comparing Directory-Level Validation and Task-Level Validation

In this example, a telephone attribute is mapped in `directory.xml` to a directory-level validation rule requiring phone numbers to be formatted as `nnn-nnn-nnnn`. All fields configured with the phone attribute are validated against the `nnn-nnn-nnnn` format whether or not the field appears in a Create User task screen, a Create Supplier task screen, or any other task screen.

If a Telephone Number field appears on a Create Customer task screen, like telephone number fields in other task screens, this field is configured with the telephone attribute that requires the `nnn-nnn-nnnn` telephone number format. However, because some of the company's customers are located in other states, the Telephone Number field on the Create Customer task screen is also associated with the following task-level validation logic:

- Check the value in the State field.
- When the customer is located out of state, make sure that the area code of the customer's telephone number is appropriate for the customer's state.

Validation Rule Sets

With directory-level validation, one or more validation rules are assigned to a rule set, and the rule set is associated with a managed object attribute.

Rule sets let you define and apply rules in a granular way, as in the following examples:

- A rule can be used in different rule sets
- Rules can be executed in different combinations

When a rule in a rule set fails (for example, a Java or JavaScript rule returns `False`), any exception messages associated with the rule are presented to the user. All validation rules associated with the attribute must be satisfied before the attribute is considered validated.

Order of Execution

Rules are executed in the order in which they are listed in the rule set.

Identity Manager executes each rule in a rule set separately, and transparently continues to each subsequent rule in the rule set unless a rule fails.

Because validation rules are executed in a predictable order, you can implement rules whose actions are dependent upon the outcome of previous rules, as in the following examples:

- One rule's output can become input to the next rule.
- When a field value is changed during validation, the new value can be evaluated in subsequent rules.

Basics of Validation Rule Definition

You perform the following basic operations when defining custom validation rules:

- **Implement a validation rule.** You implement a validation rule in any of the following ways:

- Regular expression
- JavaScript
- Java class

Note: If you cannot achieve the validation functionality you need using these implementations, you can use the Identity Manager APIs to create custom validation objects. For information, see *Programming Guide for Java*.

- **Integrate a validation rule with Identity Manager through a task screen or directory.xml.** You do so either inline (directly in the task screen or directory.xml file) or by reference (referencing a JavaScript source file or compiled Java class file), as shown in the following table:

	Inline	By Reference
Regular Expression	directory.xml or task screen	—
JavaScript	directory.xml or task screen	Source file referenced in directory.xml

	Inline	By Reference
Java	—	Class file referenced in directory.xml or task screen

- **Associate one or more validation rules with a task screen field.** You do so in either or both of the following ways:

- With task-level validation, you assign a validation rule directly to a field on a particular task screen.

Task-level validation has task-specific scope—that is, it can be used only in the context of the particular task screen where it is assigned.

- With directory-level validation, you map a rule set to a managed object attribute in directory.xml. Any task screen field that is configured with the attribute is validated against the rules in the rule set.

Directory-level validation has global scope. This means that directory-level validation can be used on any field configured with the managed object attribute, regardless of the task screen that contains the field, and regardless of the Identity Manager environment that includes the task screen.

Using Default Validation Rules

Identity Manager is shipped with the following types of default validation rules:

- Data validation of task screen fields
- Predefined validation rules defined in directory.xml

Default Data Validations

By default, Identity Manager checks certain data when an administrator submits a task for processing. When the data is invalid, Identity Manager stops processing the task and displays an error message. The data validations that Identity Manager performs are based on the type of task, as shown in the following table:

Data Validations According to Task	
Tasks	Validation
All tasks	Required fields must have a value.

Data Validations According to Task	
Tasks	Validation
Create User	An administrator cannot create an object with the same name as an existing object of the same type. For example, an administrator cannot create two access roles with the same name. Note: For users and groups, Identity Manager checks only the current organization.
Create Group	
Create Organization	
Create Access Role	
Create Access Task	
Create Admin Role	
Create Admin Task	
Create User	An administrator cannot create a user, group or organization with a name that contains any of the following characters: ■ comma (,) ■ single quote (') ■ double quote (") ■ asterisk (*) ■ ampersand (&) ■ slash (/) ■ back slash (\) ■ less than sign (<) ■ greater than sign (>) ■ equal to sign (=) ■ plus sign (+) ■ semicolon (;) ■ pound sign (#) ■ leading or trailing spaces Note: Organization names can contain a comma (,) or an ampersand (&).
Create Group	
Create Organization	

Data Validations According to Task

Tasks	Validation
All Create and Modify tasks	Attributes with Read/Write permission (excluding passwords) cannot contain the following characters: ■ comma (,) ■ percent sign (%) ■ less than sign (<) ■ greater than sign (>) ■ semicolon (;) These characters are vulnerable to cross-site scripting attacks.
Create User Self-register Change My Password Reset User Password Any custom task that collects and stores user passwords	If you are using the SiteMinder's Password Services feature to enforce password rules (such as minimum length), user passwords are validated against these rules. If the password does not satisfy the password policy, the password is not accepted. (See the <i>CA SiteMinder Web Access Manager Policy Server Configuration Guide</i> for more information.)
Modify User	Administrators cannot give themselves a role or the ability to assign a role.
Forgotten Password	If a user profile does not have a password hint and answer, that user cannot use the forgotten password feature.
Delete User Enable/Disable User	Administrators cannot delete their own profile or change the status of their account.
Delete Organization	Administrators cannot delete the organization where they are assigned the role that contains the Delete Organization task. Consider an administrator who is assigned the Organization Manager role in the Dealers organization. The Organization Manager role enables this user to delete organizations. This administrator can delete suborganizations of Dealers, but cannot delete Dealers.
Modify Organization	Administrators cannot modify the organization where they are assigned the role that contains the Modify Organization task.

Predefined Validation Rules

Identity Manager is shipped with the following validation rules predefined in `directory.xml`. Predefined validation rules are used for directory-level validation only, as shown in the following table:

Predefined Rule Name	Description
Phone pattern	Enforces the following format for telephone numbers: +nn nnn- ~~nnn~~ -nnnn
Set international	Adds the prefix +1 to an international telephone number.
Valid User	Verifies that the specified User object exists in the directory.
Valid Group	Verifies that the specified Group object exists in the directory.
Valid Organization	Verifies that the specified Organization object exists in the directory.

Predefined validation rules and custom validation rules can appear in the same rule set.

How to Implement Custom Validation Rules

You can implement a validation rule for one of the following:

- Regular expression
- JavaScript
- Java class

Regular Expression Implementation

A validation rule can be based on regular expression pattern matching. For example, you can do the following:

- Specify a list of invalid characters or values for an attribute
- Restrict the user from typing invalid constructs, such as an improperly formed DN or telephone number

The following JavaScript example enforces telephone number format as +nn nnn-xxx-xxxx:
phone=/\+\d{1,3} \d{3}-\d{3}-\d{4}/;

Regular expressions defined in XML should be wrapped in CDATA, as in the following example:

```
<ValidationRule name="Phone pattern" description="+nn nnn-xxx-xxxx"
messageid="4001">
  <RegularExpression>
    <![CDATA[ ((\+|\d)*+(\s*|\x2D))?\d\d\d-\d\d\d-\d\d\d\d]]>
  </RegularExpression>
</ValidationRule>
```

Validation rules based on regular expressions must comply with the requirements defined in the java.util.regex package of J2SE v1.4.

JavaScript Implementation

A JavaScript-based validation rule must implement the relevant interface, depending on whether the rule is used for task-level validation or directory-level validation.

At validation time, Identity Manager calls validate() and passes the value to be validated.

JavaScript Interface for Task-Level Validation

The definition of the JavaScript interface for task-level validation is as follows:

Syntax

```
public boolean validate(
    BLTHContext context,
    String attributeValue,
    StringRef changedValue,
    StringRef errorMessage
);
```

Parameters

context

Input parameter

Specifies an object that contains methods for retrieving information in the current task session.

For information about BLTHContext, see the Identity Manager Javadoc.

attributeValue

Input parameter

Specifies the value of the attribute being validated.

changedValue

Output parameter

Provides an optional transformation value that replaces the user-supplied value being validated. If no transformation is necessary, pass back null.

errorMessage

Output parameter

Provides a message that displays to the user if validation fails.

The message is displayed through `AttributeValidationException`. Identity Manager generates this exception when the method returns false.

Comments

The output parameters *changedValue* and *errorMessage* are of data type `StringRef`. This is a predefined data type that contains the field *reference* to which you assign a value, as shown in the following examples:

- Add a 1 prefix for a properly formatted telephone number:
`changedValue.reference="+1 " + phoneNumber;`
- Provide an error message for an improperly formatted number:
`errorMessage.reference="Phone number " + phoneNumber +
" does not match the format nnn-xxx-nnnn.";`

Returns

- True. The implementation considers the value in *attributeValue* to be valid, or it passes back a transformed value in *changedValue*.
- False. The implementation considers *attributeValue* to be invalid. Identity Manager generates an `AttributeValidationException` that includes *errorMessage*.

JavaScript Interface for Directory-Level Validation

The definition of the JavaScript interface for directory-level validation is as follows:

Syntax

```
public boolean validate(  
    String attributeValue,  
    StringRef changedValue,  
    StringRef errorMessage  
);
```

Parameters

attributeValue

Input parameter

Specifies the value of the attribute being validated.

changedValue

Output parameter

Provides an optional transformation value that replaces the user-supplied value being validated. If no transformation is necessary, pass back null.

errorMessage

Output parameter

Provides a message that displays to the user if validation fails.

The message is displayed through `AttributeValidationException`. Identity Manager generates this exception if the method returns false.

Comments

The output parameters *changedValue* and *errorMessage* are of data type `StringRef`. This is a predefined data type that contains the field *reference*, to which you assign a value, as shown in the following examples:

- Add a 1 prefix for a properly formatted telephone number:
`changedValue.reference="+1 " + phoneNumber;`
- Provide an error message for an improperly formatted number:
`errorMessage.reference="Phone number " + phoneNumber +
 " does not match the format nnn-xxx-nnnn.";`

Returns

- True. The implementation considers the value in *attributeValue* to be valid, or it passes back a transformed value in *changedValue*.
- False. The implementation considers *attributeValue* to be invalid. Identity Manager generates an *AttributeValidationException* that includes *errorMessage*.

Java Implementation

A Java-based validation rule must implement the relevant interface, depending on whether the rule is used for task-level validation or directory-level validation.

At validation time, Identity Manager calls *validate()* and passes the value to be validated.

Java Interface for Task-Level Validation

The definition of the JavaScript interface for task-level validation is as follows:

Syntax

```
public interface TaskValidator {
    public class StringRef {
        public String reference = new String();
        public String toString(){return reference;}
    }
    public boolean validate(
        BLTHContext ctx,
        String attrValue,
        StringRef updatedValue,
        StringRef errorMessage
    ) throws AttributeValidationException;
}
```

Parameters

ctx

Input parameter

Specifies an object that contains methods for retrieving information in the current task session.

For information about *BLTHContext*, see the Identity Manager Javadoc.

attrValue

Input parameter

Specifies the value of the attribute being validated.

updatedValue

Output parameter

Provides an optional transformation value that replaces the user-supplied value being validated. When no transformation is necessary, pass back null.

errorMessage

Output Parameter

Provides a message that displays to the user if validation fails.

Comments

Note: For more information about Java validation rules and on managed objects, see the Identity Manager Javadoc and *Programming Guide for Java*.

Returns

- True—The implementation considers the value in *attributeValue* to be valid, or it passes back a transformed value in *changedValue*.
- False—The implementation considers *attributeValue* to be invalid.

Throws

AttributeValidationException

Java Interface for Directory-Level Validation

The definition of the Java interface for directory-level validation is as follows:

Syntax

```
public interface IAttributeValidator {
    public class StringRef {
        public String reference = new String();
        public String toString(){return reference;}
    }
    public boolean validate(
        Object attributeValue,
        StringRef changedValue,
        StringRef errorMessage
    ) throws AttributeValidationException;
}
```

Parameters

attributeValue

Input parameter

Specifies the value of the attribute being validated.

changedValue

Output parameter

Provides an optional transformation value that replaces the user-supplied value being validated. When no transformation is necessary, pass back null.

errorMessage

Output parameter

Provides a message that displays to the user if validation fails.

Comments

If the validation operation requires managed objects from the directory, use `AttributeValidator`. This abstract class implements the `IAttributeValidator` interface, and includes a method for retrieving the managed object providers.

Note: For more information about Java validation rules and on managed objects, see the Identity Manager Javadoc and *Programming Guide for Java*.

Returns

- `True`. The implementation considers the value in `attributeValue` to be valid, or it passes back a transformed value in `changedValue`.
- `False`. The implementation considers `attributeValue` to be invalid.

Throws

`AttributeValidationException`.

Exceptions

`AttributeValidationException` is thrown when a validation rule cannot validate an attribute value supplied in a task screen field or programmatically. The exception contains one or more messages that are presented to the user, enabling the user to correct the entry and resubmit the task.

How this exception is thrown and how the error messages are presented for the exception depends on whether the rule is implemented as JavaScript, a Java class, or a regular expression.

Exceptions with Task-Level Validation

With task-level validation errors, `AttributeValidationException` is thrown as shown in the following table:

Rule Type	How Thrown	Error Message Source
Regular expression	By Identity Manager if the regular expression validation fails.	Identity Manager uses a generalized exception message.
JavaScript	By Identity Manager if the <code>validate()</code> method returns <code>False</code> .	The <code>errorMessage</code> parameter of the <code>validate()</code> method.

Rule Type	How Thrown	Error Message Source
Java	By the custom validation rule or by Identity Manager. Identity Manager throws the exception when the custom rule does not and the custom rule's <code>validate()</code> method returns <code>False</code> .	One of the following sources: ■ If the custom validation rule throws the exception, the exception's constructor. The constructor lets you specify the ID of a message in a resource bundle and the text of an additional message. ■ If Identity Manager throws the exception, the <i>errorMessage</i> parameter of the <code>validate()</code> method.

If the validation rule implementation does not provide an error message, Identity Manager uses a generalized error message.

Exceptions with Directory-Level Validation

With directory-level validation errors, `AttributeValidationException` is thrown in the same circumstances as shown in [Exceptions with Task-Level Validation](#) (see page 219).

Exception messages for directory-level validation errors come from two sources:

- A resource bundle. In `directory.xml`, definitions of all types of validation rules (Java, JavaScript, and regular expression) include the attribute `messageid`. This ID maps to a custom exception message in the resource bundle `IMSEExceptions.properties`. When `AttributeValidationException` is thrown, Identity Manager includes the mapped message with other error information that may be defined for the validation rule.
- Custom validation rule code. Java and JavaScript implementations can define additional exception messages for the rule. If a validation error occurs in the Java or JavaScript rule, the message is presented to the user along with the message that is mapped to the rule in the resource bundle.

The sources of these Java and JavaScript exception messages are those that are defined in the previous table.

This feature does not apply to directory-level validation rules implemented as regular expressions.

Note: For information about exception messages in resource bundles, see `AttributeValidationException` in the Identity Manager Javadoc, and see the *Programming Guide for Java*.

AttributeValidationException Constructor

When you create an `AttributeValidationException` object for a Java `validate()` method, use the following constructor:

Syntax

```
public AttributeValidationException(String attrName,  
                                   String attrValue,  
                                   String messageid,  
                                   String message);
```

Parameters

attrName

Specifies the name of the managed object attribute being validated.

attrValue

Specifies the value to validate.

messageid

Provides the ID associated with the message to display if the value cannot be validated. The ID corresponds to a message in the resource bundle `IMSEExceptions.properties`.

message

Provides an additional message that can be displayed to the user. This gives you an opportunity to display a more specific message than the one in the resource bundle, or a message from a custom resource bundle.

For more information about `AttributeValidationException`, see the Identity Manager Javadoc.

How to Configure Validation Rules

You configure a validation rule by integrating it with Identity Manager and by directly or indirectly associating it with a task screen field.

How you configure a validation rule determines whether you want the rule to be applied to a field in a particular task screen (task-level validation) or a field in any task screen (directory-level validation), as follows:

- With task-level validation, you make a direct association between the rule and a field in a particular task screen. Validation is performed on the field in the context of that task screen only.

- With directory-level validation, the association between the rule and the task screen field is indirect, as follows:
 - In `directory.xml`, you specify the validation rule, add the rule to a rule set, and associate the rule set with a managed object attribute.
 - In the User Console, a field that is configured with the managed object attribute is validated against the rule set mapped to the attribute.

Validation is performed on any field configured with the attribute, regardless of the task screen that contains the field, and regardless of the Identity Manager environment that contains the task screen.

How to Configure Task-Level Validation

You configure a task-level validation in the User Console, when defining field properties on a profile task screen. The basic steps are as follows:

1. Navigate to the Field properties section of the profile configuration screen containing the field to be validated.

Note: For information about field properties, see the *Administration Guide* and the Identity Manager online help.

2. Specify a value in one of the following fields, depending on how the validation rule is to be implemented:
 - Validation Expression. Contains a regular expression that performs the validation.
 - Validation Java Class. Contains the fully-qualified name of a Java class that performs the validation, for example:

`com.mycompany.MyJavaValidator`

Identity Manager expects the class file to be located in the root directory designated for custom Java class files.

Key Attributes in [Integration of Directory-Level Validation with Identity Manager](#) (see page 223) provides more information.

- Validation JavaScript. Contains the complete JavaScript code that performs the validation.

You must provide JavaScript code in this field. With task-level validation, you cannot reference a file containing JavaScript code.

For information about defining other field properties on a profile configuration screen, click the Help button on the screen.

How to Configure Directory-Level Validation

You configure directory-level validation in `directory.xml` and in a task screen. The basic steps are as follows:

- In `directory.xml`, do the following:
 - Specify a validation rule in the `ValidationRule` element.
 - Specify a rule set in the `ValidationRuleSet` element. A rule set contains one or more predefined rules, custom validation rules or rules of both types.
 - Associate a rule set with a managed object attribute in the `ImsManagedObjectAttr` element.
- In a task screen, the field to be validated must be configured with the attribute mapped to the rule set.

Integration of Directory-Level Validation with Identity Manager

You define validation rules and rule sets to Identity Manager through the `ImsManagedObjectAttrValidation` element of `directory.xml`.

The schema for the `ImsManagedObjectAttrValidation` element is as follows:

```
<xs:element name="ImsManagedObjectAttrValidation" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ValidationRule" minOccurs="0"
                  maxOccurs="unbounded">
        <xs:complexType>
          <xs:choice>
            <xs:element name="Java">
              <xs:complexType>
                <xs:attribute name="class" type="xs:string"
                              use="required"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="JavaScript">
              <xs:complexType>
                <xs:simpleContent>
                  <xs:extension base="xs:string"/>
                </xs:simpleContent>
              </xs:complexType>
            </xs:element>
            <xs:element name="JavaScriptFile">
              <xs:complexType>
                <xs:attribute name="file" type="xs:string"
                              use="required"/>
              </xs:complexType>
            </xs:element>
          </xs:choice>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="RegularExpression">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string"/>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
</xs:choice>
<xs:attribute name="name" type="xs:string"
              use="required"/>
<xs:attribute name="description" type="xs:string"
              use="optional"/>
<xs:attribute name="messageid" type="xs:string"
              use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="ValidationRuleSet" minOccurs="0"
              maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ValidationRule"
                  maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="name" type="xs:string"
                      use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string"
                  use="required"/>
    <xs:attribute name="description" type="xs:string"
                  use="optional"/>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
```


The following elements are described:

ValidationRuleSet

Consists of one or more predefined or custom validation rules. A validation rule is specified in the ValidationRule element.

Both predefined rules and custom rules can appear in the same rule set. Also, a rule set can contain any combination of Java, JavaScript, and regular expression implementations.

Validation rules are performed in the order in which they appear in ValidationRuleSet. This allows for cascading validation, where output from one rule is used as input to the next.

ValidationRuleSet is associated with a managed object attribute in the ImsManagedObjectAttr element of directory.xml.

ValidationRule

Specifies a validation rule for use in a ValidationRuleSet.

ValidationRule must contain only one of the following subelements:

- **Java.** References the Java class file that implements the rule.
- **JavaScript.** Contains the inline JavaScript code that implements the rule.
- **JavaScriptFile.** References the JavaScript source file that implements the rule.
- **RegularExpression.** Contains the inline regular expression that implements the rule. The regular expression should be wrapped in CDATA, as shown on Unsupported "page" cross-reference.

Key Attributes

Most of the attributes of the previously described elements are self-explanatory. However, the following attributes require explanation:

- **Attribute class of element <Java>**

With Java validation rules, the Java class must be deployed in the following root location within your application server:

IdentityMinder.ear\custom

Class files in this root location must be fully qualified, but need no other path information, for example, com.mycompany.MyJavaImpl.

- Attribute file of element <JavaScriptFile>

With a validation rule implemented in a JavaScript source file, the file must be deployed in the following root location within your application server:

IdentityMinder.ear\custom\validationscripts

JavaScript source files in this root location can be referenced by name only, for example, MyJavaScriptImpl.js.

- Attribute messageid of element <ValidationRule>

The message id specified in this attribute maps to an error message in the resource bundle IMSEExceptions.properties.

All types of validation rules (Java, JavaScript, JavaScriptFile, and RegularExpression) contain a messageid attribute.

Note: For information about the use and location of resource bundles, and for more information about the deployment of Java class files and JavaScript files, see the *Programming Guide for Java*.

Example: Inline Regular Expression

The following example shows the predefined Phone pattern validation rule, which is included in the rule set Phone format. The rule is implemented inline as a regular expression:

```
<ValidationRule name="Phone pattern" description="+nn nnn-xxx-xxxx"
                                     messageid="4001">
  <RegularExpression>
    <![CDATA[ ((\+|\d)*+(\s*|\x2D))?\d\d\d-\d\d\d-\d\d\d\d]]>
  </RegularExpression>
</ValidationRule>
<ValidationRuleSet name="Phone format" description=
  "Verify format +nn nnn-xxx-xxxx">
  <ValidationRule name="Phone pattern" />
</ValidationRuleSet>
```

In the preceding example, messageid="4001" maps to the following line in IMSEExceptions.properties:

```
4001=Attribute Validation: {0} value must match regular expression
nnn-xxx-xxxx.
```

Example: Reference to JavaScript File

The following example specifies the rule EndWithZ_js. This rule is implemented in JavaScript, and the script is located in the file EndWithZ.js. The rule set that includes the rule is not shown in the example:

```
<ValidationRule name="EndWithZ_js" messageid="custom-5001">
  <JavaScriptFile file="EndWithZ.js" />
</ValidationRule>
```

In the preceding example, the JavaScript file is assumed to be in the default location, IdentityMinder.ear\custom\validationscripts.

Association of a Validation Rule Set with a Managed Object Attribute

You associate a validation rule set with a managed object attribute through the `ImsManagedObjectAttr` element of `directory.xml`.

In the following example, the validation rule set Phone format is associated with the managed object attribute telephonenumber:

```
<ImsManagedObjectAttr physicalname="telephonenumber" displayname="Business Phone"
description="Business Phone" valuetype="String" required="false"
multivalued="false" maxlength="0" validationruleset="Phone format" />
```

Note: When a managed object attribute is associated with a validation rule set, the rule set name is displayed in the Attribute Properties screen of the Management Console.

Association of a Validation Rule Set with a Task Screen Field

With directory-level validation, you associate a rule set with a task screen field indirectly, as follows:

1. Associate the rule set with a managed object attribute, as described in the previous section.
2. Ensure that the task screen field to be validated is configured with the managed object attribute associated with the rule set. At runtime, a field value supplied by an end user is validated against the rules in the rule set.

Typically, task screen fields are already configured with attributes. However, you may want to add a field to a task screen, or you may want to change the attribute assigned to a field. In those cases, if you want the value supplied to the field to be subject to directory-level validation, you must configure the field with an attribute that is mapped in `directory.xml` to the appropriate rule set.

Note: For information about configuring a task screen field with an attribute, see the information about logical attributes in the *Programming Guide for Java*.

How to Initiate Validation

At runtime, validation is initiated in any of the following ways:

User submits a task

Validates the fields on the submitted task screen that are associated with validation rules.

User navigates to a different task screen tab

Validates the fields in the vacated tab that are associated with validation rules.

User clicks a Validate button on a tab

Validates the fields in the current tab that are associated with validation rules.

The Validate button also executes Logical Attribute Handlers that include the validate method.

User changes a value in a field who's Validate on Change property is yes

Validates the fields in the current tab that are associated with validation rules.

For example, if Validate on change is enabled for an Employee Type field, and the field value is changed from Non-exempt to Exempt, all fields on the tab that are associated with validation rules are validated. One rule might require that a Salary field now have a value, and another rule might automatically change an Hourly Rate field to 0.

Custom code uses a setAttribute... method in AttributeCollection or a tab handler to set a managed object attribute value

The field is configured with the managed object attribute being set.

Sample Implementations

Sample JavaScript implementations of validation rules are located in the samples directory of your Identity Manager installment, as follows:

Identity Manager\samples\validationscripts

Chapter 11: Auditing an Identity Manager Environment

This section contains the following topics:

[Audit Data](#) (see page 229)

[How to Configure Auditing](#) (see page 229)

[How to View Audit Data in eTrust Security Command Center](#) (see page 243)

[Clean Up the Audit Database](#) (see page 243)

Audit Data

To audit an Identity Manager environment, configure Identity Manager to log audit data. Audit data provides a historical record of operations that occur in an Identity Manager environment. Some examples of audit data include the following:

- System activity for a specified period of time
- The tasks that a specific user performs
- A list of objects that were modified during a specific period of time
- The roles assigned to a user
- The operations performed for a particular user account

Audit data is generated for Identity Manager *events*. An event is an operation that is generated by an Identity Manager task. For example, the Create User task may include an AssignAccessRoleEvent event.

Note: For more information about events, see the *Administration Guide*.

How to Configure Auditing

The steps to configure auditing are as follows:

1. Install support for auditing.

Note: For more information, see the information on configuring auditing in the *Installation Guide*.

2. Configure an [audit settings file](#) (see page 230).
3. [Enable auditing](#) (see page 242) for the tasks that Identity Manager will audit.

Note: Using SiteMinder, you can also monitor access control data. See the *CA SiteMinder Web Access Manager Policy Server Configuration Guide*.

Configure Audit Settings

You configure audit settings in an audit settings file. An audit settings file determines the amount and type of information that Identity Manager audits. You can configure an audit settings file to do the following:

- Enable auditing for an Identity Manager environment.
- Enable auditing for some or all of the Identity Manager events generated by admin tasks.
- Record event information at specific states, such as when an event completes or is cancelled.
- Log information about attributes involved in an event. For example, you can log attributes that change during a ModifyUserEvent event.
- Set the audit level for attribute logging.

To configure audit settings

1. Export the current audit settings to an audit settings XML file.
2. Configure audit settings in the XML [file](#) (see page 231) that you exported in the previous step by adding, modifying or deleting elements in the file or changing the level of information that is recorded for each event.
3. Import the modified audit settings XML file.

Note: For more information, see the Management Console Online Help.

Audit Settings File

The audit settings file is an XML file that you create by exporting audit settings. The file has the following schema:

```
<Audit enabled="" auditlevel="" datasource="">
  <AuditEvent name="" enabled="" auditlevel="">
    <AuditProfile objecttype="" auditlevel="">
      <AuditProfileAttribute name="" auditlevel="" />
    </AuditProfile>
    <EventState name="" severity="" />
  </AuditEvent>
</Audit>
```

The schema contains the following elements:

- [Audit](#) (see page 231)
- [AuditEvent](#) (see page 233)
- AuditProfile
- AuditProfileAttribute
- [EventState](#) (see page 239)

Audit Element

Audit elements define general audit settings. The Audit element contains one or more AuditEvent elements.

An Audit element includes the following parameters:

- **enabled**—Determines the status of auditing in the current environment. The valid values are as follows:
 - **True**—Auditing is enabled for this environment.
 - **False**—There is no auditing for this environment.

Note: Use lowercase letters when you specify true or false.

- **auditlevel**—Indicates the type of information recorded for attributes involved in the task or event.

The audit level that you specify for the Audit element applies to all attributes, unless a different audit level is defined in the [AuditEvent](#) (see page 233), AuditProfile, or AuditProfileAttribute elements. Audit levels set in these elements override the audit levels defined in an Audit element.

AuditLevel Values

The valid values for AuditLevel are as follows:

- NONE—No attribute information is recorded.
- OLD—For modification events, Identity Manager records the value of an attribute before the modification event occurs. Identity Manager audits the attribute whether or not the value is directly affected by the event.

For other types of events, such as CreateUserEvent, no information is recorded.

- OLDCHANGED—For modification events, Identity Manager records the value for an attribute before the modification event only when the value changes to a new value.

For other types of events, such as CreateUserEvent, no information is recorded.

- NEW—For modification events, Identity Manager records the value of an attribute *after* the modification event occurs. Identity Manager audits the attribute whether or not the value is directly affected by the event.

For other types of events, Identity Manager records existing values.

- NEWCHANGED—For modification events, Identity Manager records the value for an attribute after the modification event only when the value changes to a new value.

For example, during a ModifyUserEvent event, a user's title changes from Assistant Manager to Manager. Identity Manager audits the value Manager, but does not audit the user's name and address, which did not change.

- BOTH—For modification events, Identity Manager records the value of an attribute before and after a modification event, whether or not that value is affected by the modification event.
- BOTHCHANGED—For modification events, Identity Manager records the old and new value for an attribute after the modification event only when the value changes to a new value.
- datasource—The JNDI name for the data source configured in the application server that points to the audit database.

Specify the following JNDI name:

auditDbDataSource

Note: For JBoss application servers, the JNDI name must appear as follows:

java:/auditDbDataSource

AuditEvent Element

Audit Event elements specify events to audit. For a list of Identity Manager events, see the *Administration Guide*.

The AuditEvent element may contain multiple AuditProfile and AuditProfileAttribute elements. The database stores member, administrator, and owner policies in compiled XML format. This format is different from the user interface where each policy appears as an expression element.

The AuditEvent element includes the following parameters:

name

Defines the name of the event to audit.

To audit or exclude an attribute for all events, specify ALL for the event name. For example, to prevent passwords from being audited, regardless of the event, specify the following:

```
<AuditEvent name="ALL" auditlevel="">
  <AuditProfile objecttype="User"      auditlevel="">
    <AuditProfileAttribute name="%PASSWORD%" auditlevel="NONE"/>
  </AuditProfile>
</AuditEvent>
```

enabled

Determines whether the event is audited. The valid values are as follows:

- True indicates that Identity Manager audits this event
- False indicates that Identity Manager does not audit this event.

Note: Use lowercase letters when you specify true or false.

auditlevel

Indicates the type of information recorded for an attribute in the audit event.

[AuditLevel Values](#) (see page 232) lists the valid values for the AuditLevel element.

Note: Settings in the AuditProfile and AuditProfileAttribute elements take precedence over global settings in the AuditEvent element.

AuditProfile Element

AuditProfile elements indicate the type of objects involved in the events to audit. For example, if you enable auditing for the PARENTORG object in a CreateUserEvent event, Identity Manager logs information about the organization in which the user is created.

The AuditProfile element can contain multiple AuditProfileAttribute elements.

The AuditProfile element includes the following parameters:

objecttype

Defines the type of object for which to record audit information. Object types are as follows:

- ACCESS_ROLE
- ACCESS_TASK
- ADMIN_ROLE
- ADMIN_TASK
- GROUP
- ORGANIZATION
- PARENTORG
- RELATIONSHIP

The RELATIONSHIP object describes container relationships, when an object includes one or more objects. Examples of container relationships include nested groups, group and role membership, and hierarchical organizations.

In the RELATIONSHIP object the parent object and the objects contained in the parent object are represented.

- USER
- NONE

auditlevel

Indicates the type of information recorded for an attribute in the object's profile.

The audit level that you specify for the AuditProfile element applies to all attributes in the object's profile unless a different audit level is defined in an AuditProfileAttribute element. Audit levels set in these elements override the audit levels defined in an AuditProfile element.

[AuditLevel Values](#) (see page 232) lists the valid values for the AuditLevel element.

AuditProfileAttribute Element

AuditProfileAttribute elements indicate the attributes that Identity Manager audits. The attributes apply to the object specified in the AuditProfile element.

Note: If there are no audit profile attributes specified, all the attributes for the object specified in the AuditProfile element are logged.

The AuditProfileAttribute element includes the following parameters:

name

Defines the name of the attribute to audit.

Specify a profile attribute for the object in the corresponding AuditProfile element. For example, if the AuditProfile element specifies the Organization object, specify the name of an organization attribute as the value for the name parameter.

Note: You must define the profile attribute in the directory configuration file for the Identity Manager directory.

auditlevel

Indicates the type of information recorded for an attribute.

[AuditLevel Values](#) (see page 232) lists the valid values for the AuditLevel element.

The following table shows the valid attributes for Identity Manager object types:

Valid Attributes for Identity Manager Object Types	
Object Type	Valid Attributes
ACCESS_ROLE	■ name—User-visible name for the role ■ description—An optional comment about the purpose of the role ■ members—The users who can use the role ■ administrators—The users who can assign role member or administrators ■ owners—The users who can modify the role ■ enabled—Indicates whether or not the role is enabled ■ assignable—Indicates whether the role can be assigned by an administrator ■ tasks—The access tasks associated with the role
ACCESS_TASK	■ name—User-visible name for the task ■ description—An optional comment about the purpose of the task ■ application—The application that is associated with the task ■ tag—The unique identifier for the task ■ reserved1, reserved2, reserved3, reserved4—The values of the reserved fields for the task

Valid Attributes for Identity Manager Object Types

Object Type	Valid Attributes
ADMIN_ROLE	■ name—User-visible name for the role ■ description—An optional comment about the purpose of the role ■ members—The users who can use the role ■ administrators—The users who can assign role member or administrators ■ owners—The users who can modify the role ■ enabled—Indicates whether or not the role is enabled ■ assignable—Indicates whether the role can be assigned by an administrator ■ tasks—The tasks associated with the role

Valid Attributes for Identity Manager Object Types

Object Type	Valid Attributes
ADMIN_TASK	■ name—User-visible name for the task ■ description—An optional comment about the purpose of the task ■ tag—The unique identifier for the task ■ category—The category in the Identity Manager user interface where the task appears ■ primary_object—The object on which the task operates ■ action—The operation performed on the object ■ hidden—Indicates whether the task does <i>not</i> appear in menus ■ public—Indicates whether the task is available to users who have not logged in to Identity Manager ■ auditing—Indicates whether the task enables the recording of auditing information ■ external—Indicates whether the task is an external task ■ url—The URL where Identity Manager redirects the user when an external task executes ■ workflow—Indicates whether the Identity Manager events associated with the task trigger workflow ■ webservice—Indicates whether the task is one for which Web Services Description Language (WSDL) output can be generated from the Identity Manager Management Console
GROUP	Any valid attribute that is defined for the GROUP object in the directory configuration file (directory.xml)
ORGANIZATION	Any valid attribute that is defined for the Organization object in the directory configuration file (directory.xml)
PARENTORG	

Valid Attributes for Identity Manager Object Types

Object Type	Valid Attributes
RELATIONSHIP	■ %CONTAINER%—Unique identifier of the parent object. For example, if the RELATIONSHIP object describes role membership, the container would be the role. ■ %CONTAINER_NAME%—User-visible name of the parent group ■ %ITEM%—Unique identifier of the object that is contained in the parent object. For example, if the RELATIONSHIP object describes role membership, the items would be the role members. ■ %ITEM_NAME%—User-visible name for the nested group
USER	Any valid attribute that is defined for the USER object in the directory configuration file (directory.xml)
NONE	No attributes

Note: The following applies to the preceding table:

- Enabled, assignable, auditable, workflow, hidden, webservice, and public are logged as true or false.
- When auditing tasks for roles, the user visible name is logged.
- The database stores member, administrator, and owner policies in compiled XML format. This format is different from the user interface where each policy appears as an expression

EventState Element

EventState elements indicate when to record information about events. Identity Manager can log information at several points, or *states*, during an event's life cycle.

The EventState element includes the following parameters:

name

Defines the name of the event state to audit. The event states that you can specify are the following:

AUDIT

Records special events that exist only to audit information. These events only go to the AUDIT state and do not execute.

BEGIN

Audits the set of attributes populated by the user interface and custom handlers, including business logic task handlers, logical attribute handlers and attribute validation implementations.

This state also audits attributes populated by TEWS.

PRE

Audits attributes that are affected by event listeners that execute during the BEGIN state.

APPROVED

Audits changes to attributes during the approval process.

REJECTED

Records status information when an event under workflow control is rejected.

Note: After an event is rejected, it proceeds to the Cancelled state.

EXECUTE

Records information when an event executes.

POST

Audits any changes that an event listener makes to an attribute in the POST state.

INVALID

Records status information when Identity Manager encounters an invalid event.

COMPLETE

Records status information when an event completes.

CANCELLED

Records status information when an event is cancelled.

Note: Specify the value of the name parameter using capital letters only.

severity

Defines the severity level of the event (for integration with eTrust Security Command Center [SCC]).

In SCC, you can define policies that trigger certain actions, such as sending an email when a task or event with a certain severity level occurs.

For example, in the Identity Manager audit configuration file, you can set the severity level for a CreateUserEvent in the INVALID state to WARNING as follows:

```
<AuditEvent name="CreateUserEvent" enabled="true" auditlevel="BOTH">  
    <AuditProfile objecttype="USER" auditlevel="BOTH"/>  
    <EventState name="COMPLETE" severity="WARNING"/>  
</AuditEvent>
```

In SCC, you can then create a policy that sends an email to a system administrator when an event with the severity level WARNING is audited.

For more information about SCC, see the documentation installed with SCC.

You can define the following severity levels:

- NONE—No severity level applies.
- WARNING—Functionality of the event or task may be affected.
- CRITICAL—Immediate action is required
- FATAL—The system has become unstable.

How to Enable Auditing For a Task

Once you have configured auditing for Identity Manager, you can decide which tasks should log auditing data. To generate audit data for a task, create or modify the task in the User Console, and select the Enable Auditing check box, as shown in the following illustration:

The screenshot shows a web form titled "Create Admin Task:". Below the title is a tabbed interface with five tabs: "Profile", "Scope", "Tabs", "Fields", and "Events". The "Profile" tab is currently selected. The form contains the following fields and controls:

Field Name	Value / Control	Notes
Name*	Create Contractor	
Tag*	CreateContractor	
Description		
Category*	☒ Admin Roles ☐	
Primary Object*	User	
Action*	Create	
User Synchronization	Off	(user primary object only)
Account Synchronization	Off	(user primary object only)
Hide In Menus	☐	
Public Task	☐	
Enable Auditing	☒	

How to View Audit Data in eTrust Security Command Center

When Identity Manager integrates with SCC, you can view and print audit data in the SCC Log Viewer. The audit data displays in a tabular format, as shown in the following illustration:

Log Viewer Using Profile: CA Identity Manager Events (Node: dlim.ca.com)										
Options Help										
Message	Entry ID	Timestamp	...	User	Log ...	Source	Co... 	Category	Type	Event ID
Taxonomy ...	26079	8/23/05 4:43 ...	t...	N/A	env1	eTrust IdentityM...	abansal	Identity Manage...	73	0
Taxonomy ...	26074	8/23/05 4:43 ...	t...	N/A	env1	eTrust IdentityM...	abansal	Identity Manage...	73	0
Taxonomy ...	26073	8/23/05 4:43 ...	t...	N/A	env1	eTrust IdentityM...	abansal	Identity Manage...	73	0
Taxonomy ...	26071	8/23/05 4:43 ...	t...	N/A	env1	eTrust IdentityM...	abansal	Identity Manage...	73	0
Taxonomy ...	26068	8/23/05 4:43 ...	t...	N/A	env1	eTrust IdentityM...	abansal	Identity Manage...	73	0
Taxonomy ...	26065	8/23/05 4:43 ...	t...	N/A	env1	eTrust IdentityM...	abansal	Identity Manage...	73	0
Taxonomy ...	26060	8/23/05 4:43 ...	t...	N/A	env1	eTrust IdentityM...	abansal	Identity Manage...	73	0
Taxonomy ...	26059	8/23/05 4:43 ...	t...	N/A	env1	eTrust IdentityM...	abansal	Identity Manage...	73	0
Taxonomy ...	26058	8/23/05 4:43 ...	t...	N/A	env1	eTrust IdentityM...	abansal	Identity Manage...	73	0
Taxonomy ...	26057	8/23/05 4:42 ...	t...	N/A	env1	eTrust IdentityM...	abansal	Identity Manage...	73	0
Taxonomy ...	26056	8/23/05 4:42 ...	t...	N/A	env1	eTrust IdentityM...	abansal	Identity Manage...	73	0
Taxonomy ...	26055	8/23/05 4:42 ...	t...	N/A	env1	eTrust IdentityM...	abansal	Identity Manage...	73	0
Taxonomy ...	26054	8/23/05 4:42 ...	t...	N/A	env1	eTrust IdentityM...	abansal	Identity Manage...	73	0
Taxonomy ...	26053	8/23/05 4:42 ...	t...	N/A	env1	eTrust IdentityM...	abansal	Identity Manage...	73	0
Taxonomy ...	26052	8/23/05 4:42 ...	t...	N/A	env1	eTrust IdentityM...	abansal	Identity Manage...	73	0

For information about using the Log Viewer, see *eTrust Security Command Center Product Integration Kit Guide for CA Identity Manager*, which is available on the CA Support site.

Note: To view Identity Manager events in eTrust Security Command Center, you must install the iRecorder for CA Identity Manager. For more information on how to install iRecorder, see the *Installation Guide*.

Clean Up the Audit Database

The auditing database may eventually accumulate records that are no longer necessary. To remove these records, execute the following database procedure in the db\auditing directory:

```
garbageCollectAuditing environment-OID MM/DD/YYYY
```

environment-OID

Defines the object ID of the Identity Manager environment

MM/DD/YYYY

Defines the date before which the auditing records should be removed.

Chapter 12: Identity Manager Localization

This section contains the following topics:

[Introduction](#) (see page 245)
[How to Configure the System for Localization](#) (see page 246)
[How to Plan the Localization](#) (see page 247)
[How to Localize Resource Bundles](#) (see page 250)
[Translate Email Templates](#) (see page 257)
[Configure Multibyte Character Support \(SQL Server\)](#) (see page 258)
[How to Localize Branding Skins](#) (see page 258)
[Modify the Change My Account Task](#) (see page 259)
[Localizing Custom Code](#) (see page 259)
[How to Display Identity Manager in a Different Language](#) (see page 260)
[Sample Localized Identity Manager Environments](#) (see page 261)
[Set the imlanguage HTTP Header in a SiteMinder Response](#) (see page 264)

Introduction

You can localize the following elements in the User Console to a particular language:

- Prompts, buttons, text, tabs, and titles that appear in the User Console
- Validation, status, and error messages and instructions that are displayed to Identity Manager users
- Email notifications
- Graphics
- Custom features created with the Identity Manager APIs

How to Configure the System for Localization

To configure the system to use a translated version of Identity Manager, you must perform the following steps:

1. Make sure that you have installed support for internationalization.

Note: For instructions, see the *Installation Guide*.

2. Make sure that you have installed the software required to use the localization tools.
3. Configure the PATH Variable.

Prerequisites for Localization Support

Identity Manager includes tools that you use to localize a new Identity Manager environment. To use these tools, you need the following software:

Java SDK native2ascii tool

Translates to and from Unicode escape sequences. Because localization mainly involves the translation of property files, the tools process all files in ASCII with Unicode escape sequences.

To configure the localization tools, install the native2ascii tool by installing a JDK on the system where the Identity Manager Administrative Tools are installed.

For information about the native2ascii tool, go to the following location at the Java Web site:

<http://java.sun.com/j2se/1.4.2/docs/guide/intl/index.html>

Perl

Required to run the localization tools.

If Identity Manager integrates with SiteMinder, you can use the version of Perl that is installed with SiteMinder.

If Identity Manager does not integrate with SiteMinder, you download a free version of Perl from the Internet.

Configure the PATH Variable

To localize Identity Manager, you need to have Perl and the Java SDK on your path. You can add them with the following commands:

Windows:

```
PATH path to bin directory where Perl is installed;%PATH%  
PATH %JAVA_HOME%\bin;%PATH%
```

UNIX:

```
PATH=path to bin directory where Perl is installed:$PATH  
export PATH
```

Configure Perl Tools for Localization

If you are using the version of Perl that is included with SiteMinder, install an additional package to configure the Perl tools to work with the localized files.

To configure the Perl Tools for localization

1. Enter the following command from a command prompt:

```
mkdir "C:\Program Files\Netegrity\SiteMinder\CLI\html\lib"  
ppm
```

2. At the PPM prompt, enter the following command:

```
PPM> install XML::DOM
```

3. When prompted to install the XML-DOM package, enter y.

The installation displays the following messages:

```
Installing package 'XML-DOM'...  
Bytes transferred: 53571  
Writing C:\Program  
Files\netegrity\siteminder\CLI\site\lib\auto\XML\DOM\packlist  
exit
```

How to Plan the Localization

Localization is achieved by doing the following:

- Selecting the elements to translate
- Translating the resource bundles

Elements that Can Be Translated

To display Identity Manager in a different language, you translate the elements that constitute the user interface. The elements to translate and the places where you translate them are as follows:

Button text, Status messages, and Prompts

Translated in the IMSResources resource bundle

Error messages

Translated in IMSEExceptions resource bundle or resource bundles that you create for custom error messages

Text, such as the welcome message and the Logout and Help links, on the main screen in the User Console

Translated in the resources.properties file

Email notifications

Translated in [email templates](#) (see page 257)

Graphics

Translated in [skins](#) (see page 258)

Custom features created with the Identity Manager APIs

Translated in [Identity Manager APIs](#) (see page 259)

Resource Bundles

A *resource bundle* is an Identity Manager file that associates user interface elements with locale-specific text strings. In this file, a user interface element is assigned a key ID, which is paired to a locale-specific text string.

When Identity Manager encounters an element with a key ID on a user interface screen, such as an admin task profile screen, it displays the paired value from the resource bundle.

For example, the entry for a Submit button in an Identity Manager resource bundle is as follows:

```
global.button.submit=Submit
```


The key ID is `global.button.submit` and the value is `Submit`. When Identity Manager encounters the `global.button.submit` key ID in a user interface screen definition, it displays the word `Submit`, as shown in the following illustration:

Create Group:

Profile | Membership | Administrators | Groups

Group Name*

Group Description

Self Subscribing ☐

Dynamic Group Query

Copy from a group

Submit **Cancel**

Note the following about key IDs:

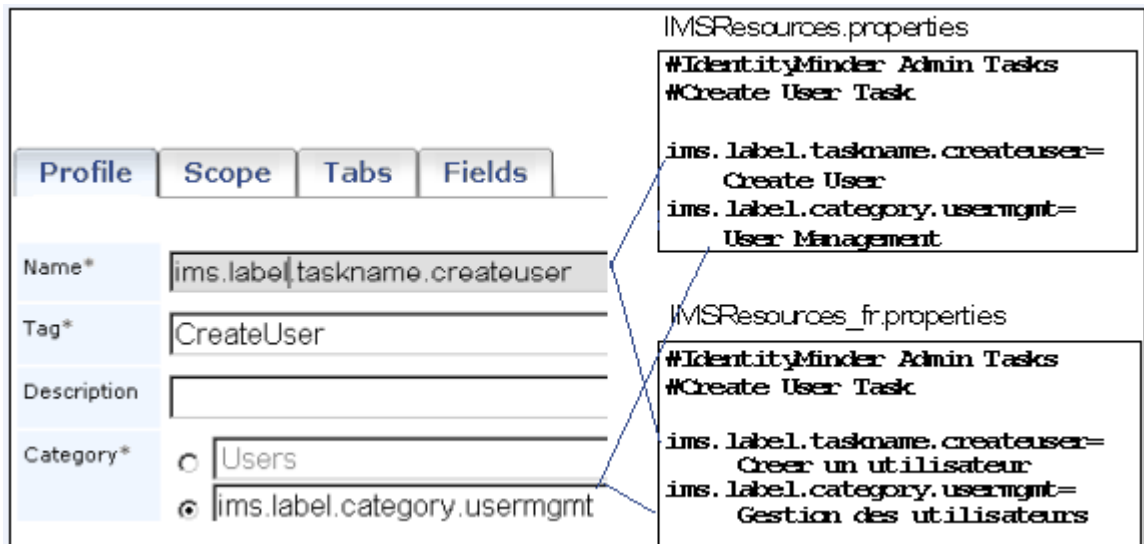
- Elements in the User Console can have only one key ID, and the key ID must be unique.
- Key IDs for elements in task screens must have the prefix `imtask.label`.
- Key IDs are case-sensitive, must contain only US-ASCII characters, and cannot contain spaces. However, key IDs may contain multiple strings separated by periods (.). For example, the key ID for the Create Organization task name may resemble the following:

`imtask.label.taskname.createorg`

With key ID/value pairs, you use a translation tool that changes the value for a different language. For example, to display the `Submit` button in French, the translation tool changes the value paired with the `global.button.submit` key ID in the resource bundle as follows:

`global.button.submit=Enter`

The following illustration shows how key IDs in a task screen definition map to key IDs in two IMSResources.properties files:



How to Localize Resource Bundles

Localizing resource bundles involves the following operations:

1. Export your roles into a RoleDefinitions.xml file, and do the following:
 - If you localize Identity Manager for a single language, go to the next step.
 - If you localize Identity Manager for multiple languages, tokenize the RoleDefinitions file.
2. Translate the RoleDefinitions.xml file.
3. Translate the resource bundles.
4. Install the localized files.

Tokenize the RoleDefinitions.xml File

To support multiple languages for an Identity Manager environment, you tokenize the RoleDefinitions.xml file, and include it in the resource bundle.

You can tokenize one of the RoleDefinitions.xml files that is installed with Identity Manager, or you can tokenize the RoleDefinitions.xml file for an existing environment that you have customized.

Tokenizing a RolesDefinition.xml file is only required if you are localizing an Identity Manager environment for more than one language.

Note: If you want to tokenize the RolesDefinition.xml file for an existing environment, you export the environment settings, complete the required steps, and then import a modified file into a *new* environment. If you import the modified file into the original environment, Identity Manager adds the tokenized definitions to the existing definitions instead of replacing them.

To tokenize the RoleDefinitions.xml file

1. If you are localizing an existing Identity Manager environment which you have customized, complete the following:
 - a. [Export the RoleDefinitions.xml](#) (see page 182) file.
 - b. Copy the file to the *im_admin_tools_dir*\samples\Localization\English directory.

2. Navigate to *im_admin_tools_dir*\samples\Localization\English at a command prompt.

3. Convert native characters in RoleDefinitions.xml to ASCII as follows:

```
native2ascii -encoding UTF8 RoleDefinitions.xml RoleDefinitions.tmp  
RoleDefinitions.xml
```

Represents the RolesDefinition.xml file that you want to tokenize.

If you are tokenizing the RoleDefinitions.xml file for a default environment (with no customizations), use the RolesDefintions.xml file in the *im_admin_tools_dir*\samples\Localization\English directory.

If you are tokenizing a localizing a custom environment, use the RolesDefintions.xml file that you copied to *im_admin_tools_dir*\samples\Localization\English in step 1.

4. Tokenize RoleDefinitions.xml as follows and generate the additional resources:

```
perl ..\Tools\tokenizeroledef.pl RoleDefinitions.tmp RoleDefinitions_tok.xml  
>tok.properties
```

5. Delete the following temporary file.

```
del RoleDefinitions.tmp
```

6. Append the properties to your English resources as follows:
`copy IMSResources.properties+tok.properties IMSResources_en.properties`
7. Import RoleDefinitions_tok.xml into a *new* environment.

Translate the RoleDefinitions.xml file

Perform the following procedure to translate the RoleDefinitions.xml file.

To translate the RoleDefinitions.xml file

1. If you are localizing an existing Identity Manager environment which you have customized, complete the following:
 - a. [Export the RoleDefinitions.xml](#) (see page 182) file.
 - b. Copy the file to the *im_admin_tools_dir*\samples\Localization\English directory.

Note: If you are localizing an Identity Manager environment for multiple languages, you completed this step when you tokenized the RolesDefinitions.xml file.

2. Navigate to *im_admin_tools_dir*\samples\Localization\language at a command prompt.
3. Create a RoleDefinitions.tmp in ASCII format as follows:

```
native2ascii -encoding UTF8 ..\English\RoleDefinitions.xml  
..\English\RoleDefinitions.tmp
```

RoleDefinitions.xml

Represents the RolesDefinition.xml file that you want to tokenize.

If you are tokenizing the RoleDefinitions.xml file for a default environment (with no customizations), use the RolesDefinitions.xml file in the *im_admin_tools_dir*\samples\Localization\English directory.

If you are tokenizing a localizing a custom environment, use the RolesDefinitions.xml file that you copied to *im_admin_tools_dir*\samples\Localization\English in step 1.

4. Create a localized RoleDefinitions.tmp as follows by using the translation script with locale-specific input to replace xx:

```
perl ..\Tools\translateroledef.pl xx.dic ..\English\RoleDefinitions.tmp  
RoleDefinitions.tmp >missing.dic
```

5. Delete the following English and localized RoleDefinitions.tmp files:

```
del ..\English\RoleDefinitions.tmp  
del RoleDefinitions.tmp
```

6. If missing.dic file recorded missing words, add those entries to the xx.dic and repeat Steps 3 and 4.

7. When the missing.dic file is empty, create an Unicode-formatted RoleDefinitions.xml file from the RoleDefinitions.txt file as follows:

```
native2ascii -reverse -encoding UTF8 RoleDefinitions.txt
RoleDefinitions.xml
```

8. From the Management Console, import the localized RoleDefinitions.xml into a *new* environment.

Note: If you import the localized RoleDefinitions.xml file into an existing environment, Identity Manager adds the new localized task and screen definitions to the existing definitions instead of replacing them.

Translate Resource Bundles

The name of each resource bundle is made up of the family base name, such as IMSResources or IMSEExceptions, and a locale indicator, such as *fr* for French or *de* for German. For example, the name of the French IMSResources resource bundle becomes IMSResources_fr.properties.

To translate resource bundle families for Identity Manager

1. Navigate to *im_admin_tools_dir*\samples\Localization\language at a command prompt.
2. Decide what locale indicators to use in localized resource files that use the following convention:

resource_locale.properties

resource

Defines the name of the resource bundle.

locale

Defines the abbreviation for the language.

For example, for the Japanese language, IMSResources.properties becomes IMSResources_ja.properties

Note: For a list of ISO 639-compliant locale indicators, see the following URL:

<http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt>

3. Translate resource bundles using the dictionary as follows, replacing the <xx> with the appropriate language abbreviation:

```
perl ..\Tools\translateresourcebundle.pl <xx>.dic
..\English\IMSResources_en.properties IMSResources_<xx>.properties
>missing.dic

perl ..\Tools\translateresourcebundle.pl
<xx>.dic ..\English\IMSEExceptions_en.properties IMSEExceptions_<xx>.properties
>missing.dic

perl ..\Tools\translateresourcebundle.pl xx.dic
..\English\IdentityManagerExceptions_en.properties
IdentityManagerExceptions_xx.properties >missing.dic

perl ..\Tools\translateresourcebundle.pl xx.dic
..\English\feeder_en.properties feeder_xx.properties >missing.dic

perl ..\Tools\translateresourcebundle.pl xx.dic
..\English\improvising_en.properties improvising_xx.properties
>missing.dic

perl ..\Tools\translateresourcebundle.pl xx.dic
..\English\scheduling_en.properties scheduling_xx.properties >missing.dic

perl ..\Tools\translateresourcebundle.pl xx.dic
..\English\snapshottype_en.properties snapshottype_xx.properties >missing.dic

perl ..\Tools\translateresourcebundle.pl xx.dic
..\English\cal2_resources_en.properties cal2_resources_xx.properties
>missing.dic
```

4. If missing.dic is not empty, then you should add those entries to your xx.dic with the appropriate translations and re-execute the command.

To support multiple languages, you create a set or *family* of resource bundles. In a family, each resource bundle shares the same set of key IDs, but the values paired with the key IDs differ based on the locale of the file. For example, the value for the Submit button key ID may be "Submit" in the English resource bundle, and "Entrer" in the French resource bundle.

Install the Resource Bundles

Installing the localization files completes the localization process.

To install the localized files

To install the resource bundles for one or more languages, copy the resource bundles as follows:

```
copy IMSResources_xx.properties
```

```
IdentityMinder.ear\custom\resourceBundles\IMSResources_xx.properties
```

```
copy IMSEExceptions_xx.properties
```

```
IdentityMinder.ear\custom\resourceBundles\IMSEExceptions_xx.properties
```

```
copy IdentityManagerExceptions_xx.properties
```

```
IdentityMinder.ear\custom\provisioning\resourceBundles\IdentityManagerExceptions_
xx.properties
```

```
copy feeder_xx.properties
```

```
IdentityMinder.ear\custom\provisioning\resourceBundles\feeder_xx.properties
```

```
copy improvisioning_xx.properties
```

```
IdentityMinder.ear\custom\provisioning\resourceBundles\improvisioning_xx.properti
es
```

```
copy scheduling_xx.properties
```

```
IdentityMinder.ear\custom\provisioning\resourceBundles\scheduling_xx.properties
```

```
copy snapshottype_xx.properties
```

```
IdentityMinder.ear\custom\provisioning\resourceBundles\snapshottype_xx.properties
```

```
copy ca12_resources_xx.properties
```

```
IdentityMinder.ear\user_console.war\app\ca12\resources_xx.properties
```

Note: The resource bundles are copied to three different locations in IdentityMinder.ear\custom. The ca12_resources_xx.properties is copied to a different location and must be renamed to resources_xx.properties.

Install the Translated RoleDefinitions.xml Files

Use the Management Console to create a new environment or to update an existing environment with the tokenized or translated role definitions. Be sure to use the appropriate role definitions file for the type of environment you are creating

RoleDefinitions.xml

Contains role and task definitions for an organization-based user store without provisioning tasks.

eTrustAdmin-RoleDefinitions.xml

Contains role and task definitions for an eTrust Directory user store, which is also used as a Provisioning Directory.

Provisioning-RoleDefinitions.xml

Contains role and task definitions for an organization-based user store with a separate provisioning directory and provisioning tasks.

NoOrganization-RoleDefinitions.xml

Contains role and task definitions for a user store that does not include organizations or provisioning tasks.

NoOrganization-Provisioning-RoleDefinitions.xml

Contains role and task definitions for a user store that does not include organizations and uses a separate provisioning directory and provisioning tasks.

ProvisioningOnly-RoleDefinitions.xml

Contains role and task definitions for provisioning tasks only. May be used for adding provisioning tasks to existing environments with organization based user stores.

NoOrganization-ProvisioningOnly-RoleDefinitions.xml

Contains role and task definitions for provisioning tasks only. May be used for adding provisioning to existing environments with user stores that do not support organizations.

Note: The localized files can be made the defaults by deploying the localized files to `IM.ear/management_console.war/WEB-INF/Template/environment`. This will allow the localized environments to be deployed without selecting a `RoleDefinitions.xml` file.

Translate Email Templates

Identity Manager can send email notifications to interested users when certain events occur, such as when an administrator approves a request or when a task completes.

You can translate the templates to the language you need. You can specify one set of email templates for a given Identity Manager environment. Email notifications are generated from email templates.

Note: If the email contains internationalized characters, users must set the encoding for their email client to Unicode (UTF-8) to view the message correctly. For information about changing browser encodings, see the documentation included with the browser.

To translate email templates

1. Configure support for email notifications as described in the *Installation Guide*.
2. Complete the procedures for using email notification in the *Administration Guide*.

When you customize the email templates, translate the content to the appropriate language, and set the encoding for the template to UTF-8.

Note: If the email templates that you create contain non-ASCII characters, save them in a Unicode (UTF-8) encoded format.

Configure Multibyte Character Support (SQL Server)

Note: This procedure is required to support user IDs that contain multibyte characters *only* when Identity Manager integrates with SiteMinder. If Identity Manager does not integrate with SiteMinder, skip this procedure.

When creating a user with a user ID that contains multibyte characters, an incorrect "Task Failed" message appears in the Identity Manager User Console. Identity Manager creates the user account with the correct user ID. The error message appears because of an issue in a default SiteMinder SQL query object that is created for a relational database deployment. To prevent this error message from appearing, complete one of the following procedures:

Modify the Identity Manager directory configuration file (directory.xml):

1. In the Management Console, export the directory configuration file for the Identity Manager directory that will support multibyte characters.
2. Locate the Provider element in the directory .xml file and add a SiteMinderQuery element that looks like the following example:

```
<Provider ...>
    ...
    <SiteMinderQuery name="InitUser" query="SELECT tblUsers.loginid FROM tblUsers
    WHERE tblUsers.loginid=N&apos;%s&apos;"/>
</Provider>
...
```

The query you specify must comply with XML rules for displaying special characters. In the preceding example, %s' represents a single quote (').

3. Save the directory.xml file, and reimport it using the Identity Manager Management Console.

Update the default query in SiteMinder

In the Policy Server User Interface or Administrative UI (SiteMinder r12 only), edit the query object that has the same name as the user directory. Update all queries so that any entries that have the format: WHERE ... column1='value1' resemble the following: WHERE column1=N'value1'

How to Localize Branding Skins

Branding skins determine the graphics, colors, and fonts for an Identity Manager environment. Identity Manager includes a default skin and several sample skins that you can customize for a particular language.

Modify the Change My Account Task

For localized environments, the default consultation process has English role names in the configuration. This will cause the Change My Account task to fail. To fix this, do the following:

1. In the User Console, modify the Change my Account task.
2. Click the Events tab.
3. Select the language-specific User Manager and User Approver roles from the role list.
4. Save the changes.

Localizing Custom Code

When you create custom code using the Identity Manager APIs, you can do the following:

- Localize the exception messages that are displayed by Logical Attribute Handlers, Business Logic Task Handlers, and validation rules.
- Localize the display names for logical attributes.
- Specify the language that Identity Manager uses for all its communications with a third-party client when using the Remote Task Execution (TEWS) feature.
- Use the Localizer helper class to retrieve the localized version of a string for a given locale. For example, you can retrieve the localized version of a label on a task screen or a task name.

For more information about localizing custom code, see the following documents:

- The Localizer and IMSEException area of the Support Objects information in *Programming Guide for Java*.
- The class descriptions for the Localizer and IMSEException objects in the Javadoc.

How to Display Identity Manager in a Different Language

The language that a user sees in the User Console depends on the following:

- The language preference defined in the user's browser
- The availability of localized resource bundles

Note: Data that is stored in a user store is displayed in the User Console in the language in which it was originally typed in the user store. For example, if an address is typed in Japanese and the User Console is displayed in English, the address will appear in Japanese.

Identity Manager uses the following "best match" strategy when determining which language to display:

1. An Identity Manager user sets a language preference in the browser used to access Identity Manager.

For instructions on setting the language preference, see the online help for the Web browser.
2. The user tries to access an Identity Manager environment.
3. The browser sends the prioritized list of languages to the Identity Manager server as a request header.
4. The Identity Manager server reads the list of languages from the request header.

When the Identity Manager resource bundles that correspond to the first language in the list exist, Identity Manager displays the User Console in that language.

If no resource bundles correspond to the first language, Identity Manager continues through the list of languages until it finds a language for which it has the appropriate resource bundles.

If Identity Manager cannot find a match for the languages set in the browser, the User Console is displayed in the default locale.

Identity Manager users can set their language preferences in the browser that they use to access Identity Manager.

Sample Localized Identity Manager Environments

Identity Manager includes the files to create translated versions of the User Console. You can use these samples as defined, or use the samples as templates for creating a custom translated User Console. The following languages are supported:

- French
- Korean
- Japanese
- German
- Simplified Chinese
- Spanish
- Italian

These files that support these language versions are installed at the following location:

im_admin_tools_dir\samples\Localization\language

im_admin_tools_dir

Specifies the installed location of the Identity Manager Administrative Tools.

language

Specifies the language that you want to use.

For example, the default location of the Japanese files in a Windows environment is as follows:

C:\Program Files\CA\IAM Suite\Identity
Manager\tools\samples\Localization\Japanese

Install the Sample Resource Bundles

To use one of the translated versions of the Identity Manager User Console, install the resource bundles for the appropriate language.

To install the sample resource bundles

1. From a command prompt, navigate to the directory where the translated version of the help resides, as follows:

```
im_admin_tools_dir\samples\Localization\language
```

im_admin_tools_dir

Specifies the installed location of the Identity Manager Administrative Tools.

language

Specifies the language that you want to use.

For example, the default location of the Japanese files in a Windows environment is as follows:

```
C:\Program Files\CA\IAM Suite\Identity  
Manager\tools\samples\Localization\Japanese
```

2. Copy the resource bundles as follows:

```
copy IMSResources_xx.properties  
IdentityMinder.ear\custom\resourceBundles\IMSResources_xx.properties
```

```
copy IMSEExceptions_xx.properties  
IdentityMinder.ear\custom\resourceBundles\IMSEExceptions_xx.properties
```

```
copy IdentityManagerExceptions_xx.properties  
IdentityMinder.ear\custom\provisioning\resourceBundles\IdentityManagerExcepti  
ons_xx.properties
```

```
copy feeder_xx.properties  
IdentityMinder.ear\custom\provisioning\resourceBundles\feeder_xx.properties
```

```
copy improvisioning_xx.properties  
IdentityMinder.ear\custom\provisioning\resourceBundles\improvisioning_xx.prop  
erties
```

```
copy scheduling_xx.properties  
IdentityMinder.ear\custom\provisioning\resourceBundles\scheduling_xx.properti  
es
```

```
copy snapshottype_xx.properties  
IdentityMinder.ear\custom\provisioning\resourceBundles\snapshottype_xx.properties
```

```
copy ca12_resources_xx.properties  
IdentityMinder.ear\user_console.war\app\ca12\resources_xx.properties
```

Note: The resource bundles are copied to three different locations in IdentityMinder.ear\custom. The ca12_resources_xx.properties is copied to a different location and must be renamed to resources_xx.properties.

Import a Sample RoleDefinitions.xml File

The RoleDefinitions.xml file creates the tasks that appear initially in the User Console.

You must import the required sample RoleDefinitions.xml file to the required Identity Manager environment.

To import a RoleDefinitions.xml file

1. In the Management Console, click Identity Manager environments.
A list of Identity Manager environments appears.
2. Click the name of an Identity Manager environment.
The Properties screen for that environment appears.
3. Click Roles.
4. Type the path and file name for one of the RoleDefintions files, or browse for the file.

Note: For a description of the available RoleDefinitions file, see [Install the Translated RoleDefintions.xml Files](#) (see page 256).

5. Click Finish.
The status is displayed in the Role Configuration Output window.
6. Click Continue to exit.
7. Repeat Steps 1 through 6 for each Identity Manager environment.

Install the Online Help

Identity Manager includes translated online help for each translated version of the User Console.

To install a translated version of the online help

1. From a command prompt, navigate to the directory where the translated version of the help resides, as follows:

```
im_admin_tools_dir\samples\Localization\language
```

im_admin_tools_dir

Specifies the installed location of the Identity Manager Administrative Tools.

language

Specifies the language that you want to use.

For example, the default location of the Japanese files in a Windows environment is as follows:

```
C:\Program Files\CA\IAM Suite\Identity  
Manager\tools\samples\Localization\Japanese
```

2. Copy the im_help.zip file for the appropriate language as follows:

```
copy im_help.zip IdentityMinder.ear\user_console.war\
```

```
IdentityMinder.ear
```

The deployed location of the Identity Manager application (IdentityManager.ear) on the application server.

Note: Consider creating a backup copy of the default online help before replacing it with a translated version. The default online help is overwritten by the translated version.

3. Unzip the im_help.zip in the user_console.war directory.
4. Restart the Identity Manager environment.

The translated version of the online help is available for use.

Set the imlanguage HTTP Header in a SiteMinder Response

When Identity Manager integrates with SiteMinder, you can define a user's locale preference using an imlanguage HTTP header. In the SiteMinder Policy Server, you set this header within a SiteMinder response and specify a user attribute as the header's value. This imlanguage header acts as the highest priority locale preference for a user.

Note: The following procedure refers to the Policy Server User Interface, which is available with SiteMinder r6.0 SP 5 and SiteMinder r12. SiteMinder r12 also includes the Administrative UI, which is a browser-based console that you can use to manage the Policy Server. For information on using the Administrative UI, see the *CA SiteMinder Web Access Manager Policy Server Administration Guide*.

To set the imlanguage HTTP header in a SiteMinder response

1. Start the Policy Server User Interface.
2. Double-click the SiteMinder response for the protected Identity Manager environment.

The SiteMinder Response dialog appears.

3. Click Create.

The SiteMinder Response Attribute Editor dialog appears.

4. Select WebAgent-HTTP-Header-Variable.

This adds a new HTTP header to the response.

5. Select User Attribute as the attribute kind.

6. Type imlanguage for Variable Name.

7. Type the physical attribute from the user directory that contains the preferred locale as the Attribute Name, for example, employeetype.

The contents of the user attribute in the user directory should contain a language preference consisting of a two-letter language abbreviation followed by an optional two-letter country code, separated by a hyphen or underscore. The language preference setting in the user attribute is not case-sensitive, as in the following examples:

- ja
- en-US
- en_uk

8. Save the response.

The imlanguage HTTP header is now set in a SiteMinder response.

Chapter 13: Identity Manager in a Production Environment

This section provides step by step functional descriptions to migrate specific pieces of functionality. It should only be used if limited changes were made in the development environment and those changes are well understood.

This section contains the following topics:

[To migrate Admin roles and task definitions](#) (see page 267)

[To migrate Identity Manager skins](#) (see page 269)

[Update Identity Manager in a Production Environment](#) (see page 269)

[Migrate the IdentityMinder EAR for JBoss](#) (see page 272)

[Migrate the IdentityMinder EAR for WebLogic](#) (see page 273)

[Migrate the IdentityMinder EAR for WebSphere](#) (see page 273)

To migrate Admin roles and task definitions

You can customize Identity Manager roles and tasks to meet the specific needs of your company. The customization involves creating or modifying admin roles and tasks or by using a Create or Modify task for an admin role or task.

An alternative method, thought *not recommended*, is to modify roles and tasks in the roledefinition.xml file. Use this method for very limited changes because of the risk of errors in editing.

This process will only migrate administrative role and task definitions. If the roles were bound to organizations, consider migrating the entire Identity Manager environment.

Important! If you changed role or task definitions in the production environment, those changes are lost when you import role or task definitions from a development environment. Importing role and task definitions overwrites existing role and task definitions with the same names.

To export Admin role and task definitions

If changes were made directly to the `roledefinition.xml` file, this file can be directly imported into the production environment. Otherwise, to export the role and task definitions:

1. If you have a Policy Server cluster, check that only one Policy Server is running.
2. Stop all but one Identity Manager node.
3. Log into the Management Console.
4. Click Identity Manager environments.
5. Select the Identity Manager environment, from which to export the role and task definitions.
6. Click Roles, then click Export and supply a name for the file.
7. Follow the instructions in the next procedure to import this file.

To import Admin role and task definitions

To import the roles and task definitions into the production environment:

1. Copy the file created in the preceding procedure to the production environment.
2. Log into the Management Console in the production environment.
3. Click Identity Manager environments.
4. Select the appropriate Identity Manager environment.
5. Click Roles.
6. Click Import and specify the name of the XML file generated by the export.
7. If these steps succeeded, start any extra Policy Servers and Identity Manager nodes that you stopped.

Note: If you still need to make changes to an Identity Manager environment, omit this step until you are done.

To verify the role and task import

To verify that the roles and tasks were imported successfully, log into Identity Manager as an administrator account that can use the following tasks:

- Modify Admin Role
- Modify Admin Task

Execute these tasks and verify that the roles and tasks reflect the newly imported role definitions.

To migrate Identity Manager skins

Identity Manager skins can be customized to give the application a specific look and feel. If you have modified skins or created new skins for a set of users, use the following steps to migrate skins from the development to the production environment.

If you are modifying a skin, you only need to copy the files that have been modified.

To migrate skins

1. Copy new and modified files from the development to the production server such as image files, style sheets, properties files, and the console page (index.jsp).
2. If multiple skins are being used, configure SiteMinder response.

Note: For more information on using multiple skins, see the *Configuration Guide*.

To verify the migration of skins, log into the User Console and check that the skin appears correctly.

Update Identity Manager in a Production Environment

After migrating Identity Manager from development to production, you may need to perform incremental updates. To migrate new Identity Manager functionality from your development environment to your production environment, execute the following steps:

1. Migrate Identity Manager environments
2. Copy the IdentityMinder.ear
3. Migrate workflow process definitions

To migrate an Identity Manager environment

An Identity Manager environment is created from the Management Console. It includes a set of role and task definitions, workflow definitions, custom features created with Identity Manager APIs, and an Identity Manager directory.

To migrate an Identity Manager environment

1. If Identity Manager integrates with SiteMinder, and you have a Policy Server cluster, check that only one Policy Server is running.
2. Stop all but one Identity Manager node.
3. Export Identity Manager environments from the Management Console in the development environment.
4. Import the exported environments in the Management Console in the production environment.
5. If Identity Manager integrates with SiteMinder, reprotect the Identity Manager realms in the Policy Server User Interface.

The policy domain is not exported from the policy store when you export an Identity Manager environment.

6. Restart the Policy Server and Identity Manager nodes that you stopped.

When migrating an Identity Manager environment, the following occurs:

- The changes on the development server overwrite changes on the production server if the same object exists in both locations.
- If new objects are created on the development environment, they are added to the production server.
- If new objects are created on the production server, they are maintained.

To export an Identity Manager environment

To deploy an Identity Manager environment on a production system, you export the environment from a development or staging system and import that environment to the production system.

To export an Identity Manager environment

1. Click Environments in the Management Console.
The Identity Manager environments screen appears with a list of Identity Manager environments.
2. Select the environment that you want to export.
3. Click the Export button.
A File Download screen appears.
4. Save the ZIP file to a location that is accessible to the production system.
5. Click Finish.

The environment information is exported to a ZIP file that you can import into another environment.

To import an Identity Manager environment

After exporting an Identity Manager environment from a development system, you can import it into a production system.

To import an Identity Manager environment

1. Click Environments in the Management Console.
The Identity Manager environments screen appears with a list of Identity Manager Environments.
2. Click the Import button.
The Import Environment screen appears.
3. Browse for the ZIP file required to import an environment.
Note: This ZIP file is created when you export an environment. For more information on exporting environments, see the upgrade section of the *Installation Guide*.
4. Click Finish.

The environment is imported into Identity Manager.

To verify the Identity Manager environment migration

To verify that the Identity Manager environment was migrated correctly, confirm that the Identity Manager environment appears in the Policy Server User Interface for the Policy Server in the production environment.

In the Policy Server User Interface, verify the following:

- The Identity Manager user directory settings are accurate
- The new Identity Manager domain exists
- The correct authentication schemes protect the Identity Manager realms

Also, verify that when you log into the Management Console, the Identity Manager environment appears when you select Environments.

Migrate the IdentityMinder EAR for JBoss

Redeploy the IdentityMinder.ear each time functionality is migrated from the development environment to the production environment. By migrating the entire EAR, you ensure that your production environment is identical to your development environment.

To migrate the IdentityMinder EAR on JBoss Application Servers

1. Copy the IdentityMinder.ear from your development environment to a location accessible to your production environment.
2. In the copy of the IdentityMinder.ear, edit the Policy Server connection information, so that it reflects the production environment.

To achieve this change, copy the `jboss_home/server/default/IdentityMinder.ear/policyserver_rar/META-INF/ra.xml` from your production environment into the IdentityMinder.ear.

3. Replace the installed IdentityMinder.ear with the copy of the IdentityMinder.ear from your development environment from Step 2 as follows:
 - a. On the production server, delete the IdentityMinder.ear:
`cluster_node_jboss_home\server\default\deploy\IdentityMinder.ear`
 - b. Replace the file you deleted with the edited copy of the IdentityMinder.ear from the development environment.
4. Repeat these steps for each node in the cluster.

Migrate the IdentityMinder EAR for WebLogic

Redeploy the IdentityMinder.ear each time functionality is migrated from the development environment to the production environment. By migrating the entire EAR, you ensure that your production environment is identical to your development environment.

To migrate the IdentityMinder EAR on WebLogic Application Servers

1. Preserve Policy Server connection information.
Policy server connection information is stored in the ra.xml file in the policyserver_rar/WEB-INF directory. Copy this file to another location, so that it can be replaced in the IdentityMinder.ear before redeploying it.
2. Copy the IdentityMinder.ear to a location available to the WebLogic Admin Server.
3. Replace the Policy Server connection information.
In the IdentityMinder.ear replace the policyserver_rar/WEB-INF/ra.xml file with the one preserved from Step .
4. Redeploy the IdentityMinder.ear
 - a. Log in to the WebLogic console.
 - b. Go to Deployments, Application, IdentityMinder

On the Deploy Tab, select Deploy (Re-Deploy) Application.

Migrate the IdentityMinder EAR for WebSphere

To migrate the IdentityMinder EAR on WebSphere Application Servers

1. Copy the *imsInstall.jacl* script from *was_im_tools_dir\WebSphere-tools* to the *deployment_manager_dir \bin* directory where:
 - *was_im_tools_dir* is the directory on the development system where the Identity Manager Tools for WebSphere are installed
 - *deployment_manager_dir* is the location where the Deployment Manager is installed.
2. On the development system where you configured the Identity Manager application, copy *was_im_tools_dir \WebSphere-tools\imsExport.bat* or *imsExport.sh* to *was_home\bin*.
3. On the command line, navigate to *was_home\bin*.
4. Make sure the WebSphere application server is running.

5. Export the deployed Identity Manager application as follows:

For Windows, enter this command:

```
wsadmin -f imsExport.bat "path-to-exported-ear"
```

where *path-to-exported-ear* is the full path and file name that the imsExport utility creates.

For Windows systems, use forward slashes (/) instead of back slashes (\) when you specify the path to was_im.ear. For example:

```
imsExport.bat "c:/program files/CA/CA Identity Manager/  
exported_ear/IdentityMinder.ear"
```

For UNIX, enter this command:

```
./wsadmin -f imsExport.jacl -conntype RMI -port 2809 path to  
exported ear
```

where *path-to-exported-ear* is the full path including the file name of the exported EAR file.

6. Copy the exported EAR file from the location on the development system where you exported it to a location on the system where the Deployment Manager is installed.
7. Replace the *was_im_tools_dir*/WebSphere-ear/IdentityMinder.ear/policyserver_rar/META-INF/ra.xml with the one from the production environment.

The ra.xml file contains the Policy Server connection information.

8. On the system where the Deployment Manager is installed, deploy the IdentityMinder EAR:
 - a. From the command line, navigate to:

```
deployment_manager_dir \bin.
```
 - b. Make sure that the WebSphere application server is running.
 - c. Run the imsInstall.jacl script as follows:

Note: The imsInstall.jacl script may take several minutes to execute.

Windows:

```
wsadmin -f imsInstall.jacl "path-to-copied-ear" cluster_name
```

where *path-to-copied-ear* is full path including the file name for the IdentityMinder EAR that you copied to the Deployment Manager system.

For example:

```
wsadmin -f imsInstall.jacl "c:/Program Files/CA/CA Identity  
Manager/WebSphere-tools/was_im.ear" im_cluster
```

UNIX:

```
./wsadmin -f imsInstall.jacl path-to-copied-ear cluster_name
```

where *path-to-copied-ear* is full path including the file name for the IdentityMinder EAR that you copied to the Deployment Manager system.

For example:

```
./wsadmin -f imsInstall.jacl /CA/CA Identity Manager/websphere-  
tools/was_im.ear im_cluster
```

9. Verify the following:

- The SiteMinder agents can connect to your policy store.
- The Policy Server can connect to the user store.
- The Identity Manager domains have been created.

Chapter 14: Identity Manager Logs

This section contains the following topics:

[How to Track Problems in Identity Manager](#) (see page 277)

[How to Trace Components and Data Fields](#) (see page 279)

How to Track Problems in Identity Manager

Identity Manager includes the following methods for recording status and tracking issues:

View Submitted Tasks task

Displays the status of all events and tasks in an Identity Manager environment. Administrators use this task in the User Console.

View Submitted Tasks provides the following types of information:

- The list of events and tasks that occur in the environment
- The list of attributes associated with an event
- Successful and failed events
- Events that are in a pending or stalled state
- Rejected events, including the reason for rejection
- Account synchronization status
- Identity policy synchronization status
- Provisioning information (when provisioning is enabled)

Application Server logs

Displays information about all of the components in an Identity Manager installation, and provides details about all operations in Identity Manager.

The location and type of log file depends on which of the following types of application servers you are using:

- WebLogic—Identity Manager information is written to standard out. By default, standard out is the console window in which the server instance is running.
- JBoss—Identity Manager information is written to the console window where the server instance is running, and to `jboss_home\server\log\server.log`
- WebSphere—Identity Manager information is written to the console window where the server instance is running, and to `was_home\AppServer\logs\server_name\SystemOut`

See the documentation for your application server for more information.

Directory Server log file

Contains information about activity that occurs in the user directory.

The type of information that is recorded and the location of the log file depend on the type of directory server that you are using. See the directory server's documentation for more information.

Policy Server log file

Displays the following information when Identity Manager integrates with SiteMinder:

- SiteMinder connection issues
- SiteMinder authentication issues
- Information about Identity Manager managed objects in the SiteMinder policy store
- Password policy evaluation

For information about configuring SiteMinder logs, see the *CA SiteMinder Web Access Manager Policy Server Administration Guide*.

Policy Server Profiler

If Identity Manager integrates with SiteMinder, allows you to trace internal Policy Server diagnostics and processing functions, including functions related to Identity Manager.

For more information, see [How to Trace Components and Data Fields](#) (see page 279).

Web Agent log files

If Identity Manager integrates with SiteMinder, the Web agents write information to the following two logs:

- Error log file—contains program and operational-level errors, for example, the Web agent not being able to communicate with Policy Server.
- Trace log file—contains warning and informational messages, such as trace messages and flow state messages. It also includes data such as header details and cookie variables.

Note: For more information about Web Agent log files, see the *CA SiteMinder Web Access Manager Web Agent Configuration Guide*.

How to Trace Components and Data Fields

You can use the SiteMinder Policy Server Profiler to trace components and data fields in the Identity Manager extensions for the Policy Server. The Profiler lets you configure filters for the tracing output so that only specific values for a component or data field are captured.

Note: For instructions on using the Policy Server Profiler, see the *CA SiteMinder Web Access Manager Policy Server Administration Guide*.

You can enable tracing for the following components:

Function_Begin_End

Provides low-level trace statements when certain methods in the Identity Manager extensions for the Policy Server are executed.

IM_Error

Traces runtime errors in the Identity Manager extensions for the SiteMinder Policy Server.

IM_Info

Provides general tracing information for the Identity Manager extensions.

IM_Internal

Traces general information about internal Identity Manager operations.

IM_MetaData

Provides tracing information when Identity Manager processes directory metadata.

IM_RDB_Sql

Provides tracing information for relational databases.

IM_LDAP_Provider

Provides tracing information for LDAP directories.

IM_RuleParser

Tracks the process of parsing and evaluating member, owner, and admin policies, which are defined in an XML file that gets interpreted at runtime.

IM_RuleEvaluation

Traces the evaluation of member, admin, owner, and scope rules.

IM_MemberPolicy

Traces the evaluation of member policies, including membership and scope.

IM_AdminPolicy

Traces the evaluation of admin policies.

IM_OwnerPolicy

Traces the evaluation of owner policies.

IM_RoleMembership

Traces information relating to role membership, such as the list of roles a user has and the list of members in a certain role.

IM_RoleAdmins

Traces information relating to role administration, such as the list of roles a user can administer and the list of administrators for a certain role.

IM_RoleOwners

Traces information relating to role ownership, such as the list of roles a user owns and the list of owners for a certain role.

IM_PolicyServerRules

Traces evaluation of member rules, such as RoleMember, RoleAdmin, RoleOwner rules, which are resolved by the Policy Server, and scope rules, such as All and AccessTaskFilter rules for AccessTasks

IM_LLSDK_Command

Traces communication between the internal Identity Manager SDK and the policy server. This trace component is used by Technical Support.

IM_LLSDK_Message

Traces messages that are explicitly sent by Java code to the Policy Server from the internal Identity Manager SDK. This trace component is used by Technical Support.

IM_IdentityPolicy

Traces the evaluation and application of Identity Policies.

IM_PasswordPolicy

Traces the evaluation of password policies.

IM_Version

Provides information about the Identity Manager version.

IM_CertificationPolicy

Traces the evaluation of certification policies.

IM_InMemoryEval

Traces the processing of Identity Manager policies, including member, admin, owner, and Identity Policies. This trace component is used by Technical Support.

IM_InMemoryEvalDetail

Provides additional detail about the processing of Identity Manager policies, including member, admin, owner, and Identity Policies. This trace component is used by Technical Support.

The data fields for which you can configure tracing are listed in the *CA SiteMinder Web Access Manager Policy Server Administration Guide*.

Chapter 15: Integrating SiteMinder with Identity Manager

This section contains the following topics:

[SiteMinder Integration](#) (see page 283)

[SiteMinder Components](#) (see page 285)

[SiteMinder Configuration](#) (see page 286)

[Collect User Credentials Using a Custom Authentication Schemes](#) (see page 287)

[Configure the LogOff URI](#) (see page 288)

[Aliases in SiteMinder Realms](#) (see page 290)

[Modify a SiteMinder Password or Shared Secret](#) (see page 292)

[How to Configure Identity Manager Agent Settings](#) (see page 293)

[Configure an Identity Manager Environment to Use Different Directories for Authentication and Authorization](#) (see page 294)

[How to Improve the Performance of LDAP Directory Operations](#) (see page 296)

SiteMinder Integration

When Identity Manager integrates with SiteMinder, SiteMinder can add the following functionality to an Identity Manager environment:

Advanced Authentication

Identity Manager includes native authentication for Identity Manager environments by default. Identity Manager administrators enter a valid username and password to log into an Identity Manager environment. Identity Manager authenticates the name and password against the user store that Identity Manager manages.

When Identity Manager integrates with SiteMinder, Identity Manager uses SiteMinder basic authentication to protect the environment. When you create an Identity Manager environment, a policy domain and an authentication scheme are created in SiteMinder to protect that environment.

When Identity Manager integrates with SiteMinder, you can also use SiteMinder authentication to protect the Management Console.

Directory Mapping

An administrator may need to manage users whose profiles exist in a different user store from the one that is used for authenticating the administrator. In other words, when logging in to the Identity Manager environment, the administrator must be authenticated using one directory and authorized to manage users in a second directory.

When Identity Manager integrates with SiteMinder, you can configure an Identity Manager environment to use different directories for authentication and authorization.

Note: For more information, see the *Configuration Guide*.

Advanced Password Policies

Identity Manager enables you to create basic password policies that manage user passwords by enforcing rules and restrictions governing password expiration, composition, and usage.

If you configure Identity Manager to integrate with SiteMinder, you can create advanced password policies that enable you to define the additional rules and restrictions.

Note: For more information, see the *Administration Guide*.

Skins for Different Sets of Users

A skin changes the look of the User Console. When Identity Manager integrates with SiteMinder, you can enable different sets of users to see different skins. To accomplish this, you use a SiteMinder response to associate a skin with a set of users. The response is paired with a rule in a policy, which is associated with a set of users. When the rule fires, it triggers the response to pass information about the skin to Identity Manager, to build the User Console.

Note: For more information, see the *Configuration Guide*.

Locale Preferences for a Localized Environment

When Identity Manager integrates with SiteMinder, you can define a user's locale preference using an `imlanguage` HTTP header. In the SiteMinder Policy Server, you set this header within a SiteMinder response and specify a user attribute as the header's value. This `imlanguage` header acts as the highest priority locale preference for a user.

Note: For more information, see the *Configuration Guide*.

Note: For information on the components in an Identity Manager implementation that includes SiteMinder, see *Installation with SiteMinder Policy Server*.

SiteMinder Components

When Identity Manager integrates with SiteMinder, the following components are added to the Identity Manager architecture:

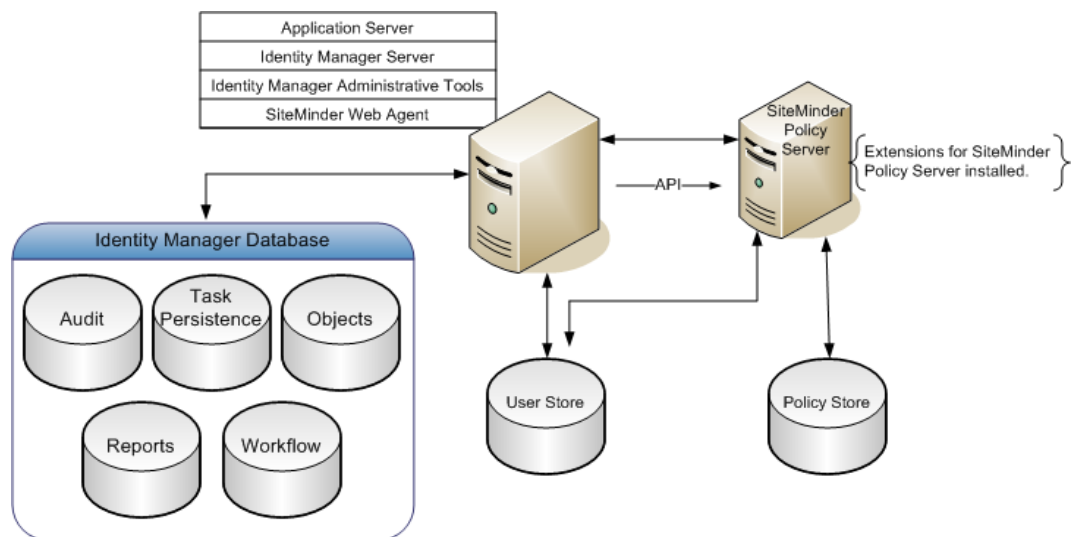
SiteMinder Web Agent

Allows communication between the Identity Manager Server and the SiteMinder Policy Server. Installed on the system with the Identity Manager Server.

SiteMinder Policy Server

Provides advanced authentication and authorization for Identity Manager, as well as other facilities such as Password Services, Single Sign-On, and so forth.

The following figure is an example of an Identity Manager installation with a SiteMinder Policy Server and Web Agent:



SiteMinder Configuration

The following sections discuss how to modify SiteMinder features, including policy domains and authentication schemes, to support Identity Manager:

[Collect User Credentials Using a Custom Authentication Schemes](#) (see page 287)

Changes the method that Identity Manager uses to collect credentials for users who try to access an Identity Manager environment.

[Configure the LogOff URL](#) (see page 288)

Prevents unauthorized access to an Identity Manager environment by enforcing a full logout.

[How to Update an Alias in SiteMinder Realms](#) (see page 290)

Updates the realms that protect an Identity Manager environment when you change the environment's alias.

SiteMinder Passwords

Lets you to change the password for the administrator account that Identity Manager uses to communicate with SiteMinder, and the shared secret for the SiteMinder agent that protects an Identity Manager environment.

How to Configure Identity Manager Agent Settings

Tunes the performance of the Identity Manager agent that communicates with the SiteMinder Policy Server.

[How to Use Different Directories for Authentication and Authorization](#) (see page 294)

Enables administrators who have profiles in one directory to manage users in a different directory.

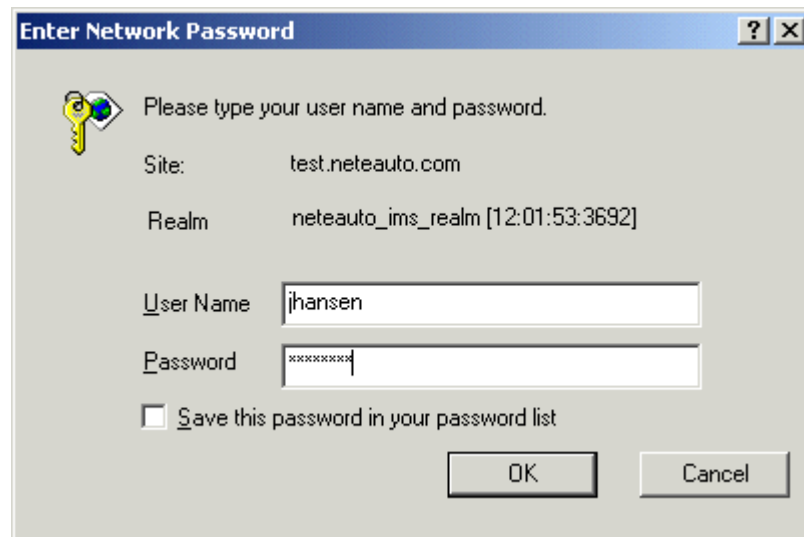
[How to Improve the Performance of LDAP Directory Operations](#) (see page 296)

Increases the throughput of Identity Manager requests to the user store by configuring SiteMinder to open multiple connections to the same directory.

Collect User Credentials Using a Custom Authentication Schemes

SiteMinder uses an authentication scheme to collect user credentials and determine a user's identity at login time. Once a user is identified, Identity Manager generates a personalized User Console based on the user's privileges.

When you create an Identity Manager environment, Identity Manager creates the basic authentication scheme, shown in the following illustration:



Note: You can implement any SiteMinder authentication scheme to protect an Identity Manager environment.

For example, you can implement an HTML Forms Authentication Scheme, which collects credentials in an HTML form. Using an HTML form lets you create a login page that may include branding elements, such as a company logo, and links to the self-registration and forgotten password pages.

Note: For information about authentication schemes, see the *CA SiteMinder Web Access Manager Policy Server Configuration Guide*.

To collect user credentials using a custom authentication scheme

1. In the Policy Server user interface, create an authentication scheme as described in the *CA SiteMinder Web Access Manager Policy Server Configuration Guide*.

2. Configure the policy that protects the appropriate Identity Manager environment to use the authentication scheme you created in Step 1 as follows:

- a. In the Domains tab, expand the policy domain for the appropriate Identity Manager environment. The domain name has the following format:

Identity Manager-environmentDomain

Identity Manager-environment

Defines the name of the Identity Manager environment. For example, when the environment name is *employees*, the domain name is *employeesDomain*.

- b. Under the domain, expand Realms. Double-click the realm that protects the Identity Manager environment. The realm name has the following format:

Identity Manager-environment_ims_realm

Note: If you configured support for public tasks, you will see an additional realm, *Identity Manager-environment_pub_realm*. This realm uses an anonymous authentication scheme to enable unknown users to use the self-registration and forgotten password features without supplying credentials. Do not modify the authentication schemes for these realms.

- c. In the Realm Properties dialog, select the authentication scheme you created in Step 1 from the drop-down list.
- d. Click OK to exit the Realm Properties dialog.

Configure the LogOff URI

To protect an Identity Manager environment, configure the SiteMinder Web Agent that protects the environment to terminate a user session after the user logs off Identity Manager.

The Web Agent terminates a user session by deleting the SiteMinder session and authentication cookies from the Web browser and instructing the Policy Server to remove any session information.

To terminate the SiteMinder session, configure logout functionality in the LogOffURI field in the Agent Configuration Object for the SiteMinder agent that protects the Identity Manager environment.

Notes:

- A SiteMinder agent has one LogOff URI. All applications protected by the agent use the same logout page.
- When you configure custom logout pages in the Management Console as described in *Configure Custom Logout Pages*, Identity Manager sends the logout request to the custom logout page *and* the LogOff URI. However, Identity Manager displays only the custom logout page to the user.

To configure the LogOff URI

1. In the System tab in the Policy Server user interface, click Agent Conf Objects.

A list of existing Agent Configuration Objects appears in the right pane.

2. Click the Agent Configuration Object for the agent that protects the Identity Manager environment.

Note: You select an Agent Configuration Object when you install the Web Agent. For more information, see the *CA SiteMinder Web Access Manager Policy Server Installation Guide*.

The Agent Configuration Properties dialog appears.

1. In the Configuration Values list, select #LogOffUri, and click Edit.

The Edit Parameter dialog appears.

In the Parameter Name field, remove the pound sign (#).

In the Value field, specify the following URI:

/idm/logout.jsp

2. Click OK twice to exit the Agent Configuration Object Properties dialog.
3. Restart the Web server.

Aliases in SiteMinder Realms

An *alias* is a unique string that is added to the URL for accessing an Identity Manager environment. For example, when an environment's alias is *employees*, the URL for accessing that environment is as follows:

`http://myserver.mycompany.org/idm/employees`

myserver.mycompany.org

Defines the fully qualified domain name of the server where Identity Manager is installed.

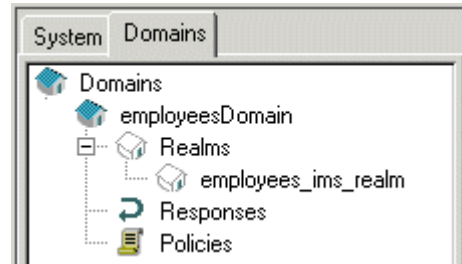
You specify at least one alias when you create an Identity Manager environment in the Management Console. (You may also specify a public alias.)

SiteMinder uses the environment name to name the objects that protect the environment. For example, when you specify the name *employees*, SiteMinder creates objects named *employeesobject_type*.

object_type

Defines the SiteMinder object, such as *employees_ims_realm*.

The following illustration shows two of the objects that SiteMinder creates:



Update an Alias in SiteMinder Realms

If you modify the protected or public alias in the Management Console, Identity Manager tries to update the alias names in the Policy Server. If Identity Manager cannot update the names, manually update them as described in the following procedure.

To update the SiteMinder realm in the policy domain

1. Log in to the Policy Server User Interface as an administrator with Domain privileges.
2. In the Domains tab, expand the policy domain for the appropriate Identity Manager environment. The domain name has the following format:

*Identity Manager-environment*Domain

Identity Manager-environment

Specifies the name of the environment you are modifying. For example, when the name is *employees*, the domain name is *employees*Domain.

3. Under the domain, expand Realms. You see one of the following realms depending on the features you configured for the environment:
 - *Identity Manager-environment_ims_realm*—Protects the Identity Manager User Console.
 - *Identity Manager-environment_pub_realm*—Enables support for public tasks, such as self-registration and forgotten password tasks.

Note: This realm appears only if you have configured a public alias.

4. In the Realm Properties dialog for the *Identity Manager-environment_ims_realm* realm, change the alias in the Resource Filter field as follows:

/idm/new_alias

Do not remove the */idm/* that precedes the alias in the resource filter.

5. If you changed the public alias in the Realm Properties dialog for the *Identity Manager-environment_pub_realm*, change the alias in the Resource Filter field as follows:

/idm/new_public_alias

6. Click OK to exit the Realm Properties dialog.

Note: Modify Identity Manager Environment Properties provides instructions on changing the alias in the Management Console.

Modify a SiteMinder Password or Shared Secret

When you install the Identity Manager Extensions to the Policy Server, you supply the password for the SiteMinder administrator account that Identity Manager uses to communicate with the Policy Server.

You can change the password; however, the password must be encrypted. To encrypt a password, use the password tool provided with Identity Manager.

Note: Before changing a SiteMinder password, be sure the `JAVA_HOME` variable is defined for your environment.

To modify a password

1. Encrypt the password as follows:
 - a. From the command line, navigate to `im_admin_tools-dir\PasswordTool`, where `im_admin_tools-dir` is the installed location of the Identity Manager administrative tools, as in the following examples:
 - **Windows:** `C:\Program Files\CA\IAM Suite\Identity Manager\tools\PasswordTool`
 - **UNIX:** `home_directory/CA/IAM Suite/Identity Manager/tools/PasswordTool`
 - b. Type the following command:

```
pwdtools new_password
```

In this command, `new_password` is the password to encrypt.

The password tool generates the encrypted password.
 - c. Copy the encrypted password.
2. Complete the relevant step as follows:
 - If Identity Manager is running on a WebLogic application server, do the following:
 - a. In the WebLogic console, edit the WebLogic resource adapter in the `policyserver_rar` connector descriptor.
 - b. Add the encrypted password as the value of the Password property.

- If Identity Manager is running on a JBoss application server, do the following:
 - a. Open ra.xml from `JBoss_home\server\default\deploy\IdentityMinder.ear\policyserver_rar\META-INF`.
 - b. Add the encrypted password as the value of the Password config-property.
 - If Identity Manager is running on a WebSphere application server, do the following:
 - a. In the WebSphere console, open ra.xml.
 - b. Add the encrypted password as the value of the Password config-property.
3. Restart the application server.

How to Configure Identity Manager Agent Settings

When Identity Manager integrates with SiteMinder, Identity Manager uses a built-in Identity Manager agent to communicate with the SiteMinder Policy Server. To tune the performance, configure the following connection settings for the Identity Manager agent.

1. Complete one of the following steps:
- If Identity Manager is running on a WebLogic or WebSphere application server, edit the resource adapter in the policyserver_rar connector descriptor in the application server's console.
 - If Identity Manager is running on a JBoss application server, open policyserver-service.xml from `<JBoss_home>\jboss-4.0.5\server\default\deploy\IdentityMinder.ear\policyserver_rar\META-INF`.
2. Configure the settings as follows:

ConnectionMax

Sets the maximum number of connections to the policy server, for example, 20.

ConnectionMin

Sets the minimum number of connections to the policy server, for example, 2.

ConnectionStep

Sets the number of additional connections to open when all the agent connections are in use.

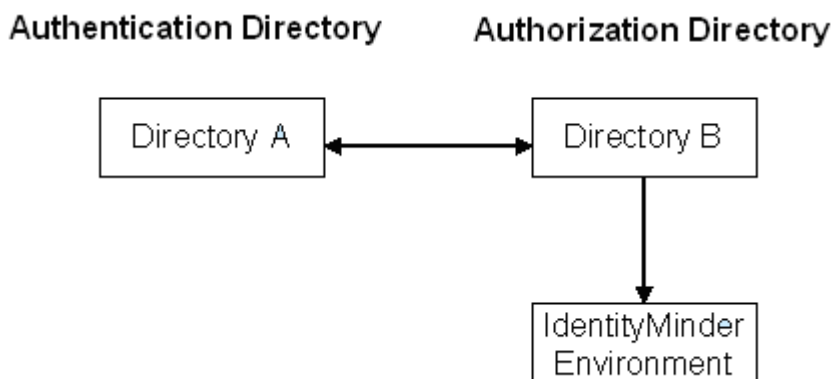
ConnectionTimeout

Specifies the amount of time that the connection should wait before timing out.

3. Restart the application server.

Configure an Identity Manager Environment to Use Different Directories for Authentication and Authorization

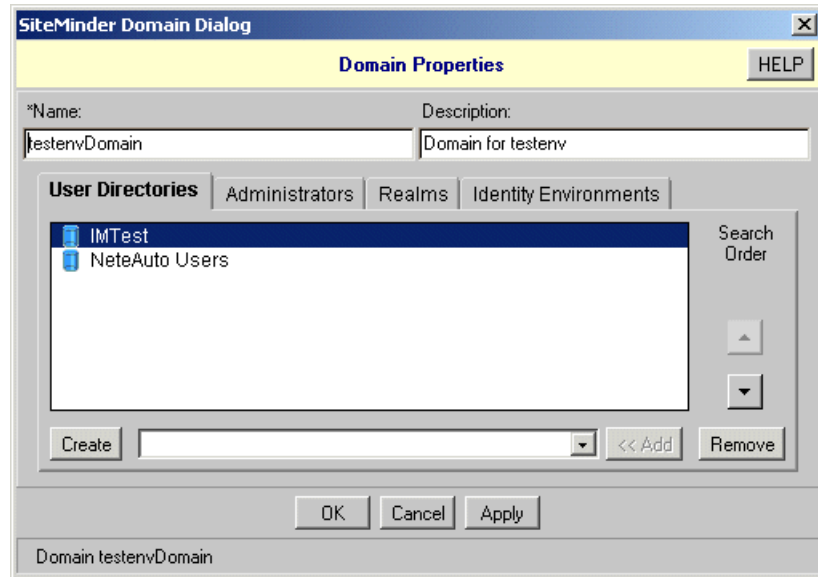
An administrator may need to manage users whose profiles exist in a different user store from the one that is used for authenticating the administrator. In other words, when logging in to the Identity Manager environment, the administrator must be authenticated using one directory and authorized to manage users in a second directory, as shown in the following illustration:



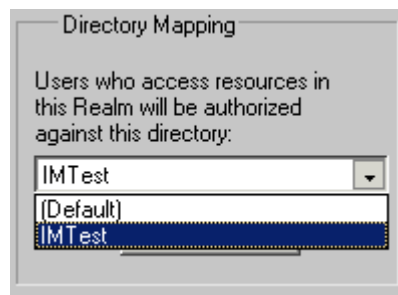
To configure an Identity Manager environment to use different directories for authentication and authorization

1. In the Policy Server user interface, create two user directories.
One directory references the authentication data (administrator profiles); the other directory references the authorization data (user profiles).
2. In the Management Console, create an Identity Manager environment.
Select the authorization directory as the Identity Manager directory.

3. In the Policy Server user interface, on the System tab, perform the following actions:
 - a. Double-click *Identity Manager-environmentDomain* to display the Domain Properties.
 - b. Add the authentication directory to the domain.
 - c. Make sure that this directory is first in the Search Order, as shown in the following illustration:



4. In the Domains tab, do the following:
 - a. Expand *Identity Manager-environmentDomain*.
 - b. Double-click *Identity Manager-environment_ims_realm* to display the Realm Properties.
 - c. Under Realm Properties, click the Advanced tab.
 - d. Click Edit Mappings to map the authorization directory to the authentication directory.
 - e. In the Advanced tab, under Directory Mapping, select the authorization directory, as shown in the following illustration:



f. Click OK to exit the Realm Properties dialog box.

5. Double-click Responses.

If you configured support for public tasks in the *<Identity Manager-environment>Domain*, the following responses display in the Response List:

- *Identity Manager-environmentresponse_ims*—The response for the User Console
- *Identity Manager-environmentresponse_ims*—The response for public tasks

6. Double-click the appropriate response to display the response properties.

- a. Click Create to add a response attribute.
- b. Type the values for the attribute in the following table:

Field	Value
Attribute	Web-Agent-HTTP-Header-Variable
Attribute Kind	user attribute
Variable Name	sm_userdn
Attribute Name	SM_USERNAME

How to Improve the Performance of LDAP Directory Operations

Directory operations may take longer to process because all Identity Manager requests for the LDAP user directory are routed through a fixed set of connections.

To increase the throughput of Identity Manager requests to the user directory, configure SiteMinder to open multiple connections to the same directory. To do this, add the LDAP server multiple times in the LDAP Directory Failover and Load Balancing Setup dialog box in the Policy Server user interface.

The number of times to enter the LDAP server (and the number of connections to create) will depend on the load on Identity Manager.

Appendix A: Global Database Task Persistence

This section contains the following topics:

[Modify the Task Persistence EJB Descriptor for WebLogic](#) (see page 297)

[Modify the Task Persistence EJB Descriptor for JBoss](#) (see page 298)

[Modify the Task Persistence EJB Descriptor for WebSphere](#) (see page 298)

Modify the Task Persistence EJB Descriptor for WebLogic

Perform the following procedure to modify the task persistence EJB descriptor in a WebLogic environment.

To modify the task persistence EJB descriptor

1. Open the following file in an editor:

```
<wl_install>\user-  
projects\domain\<your_domain>\applications\taskpersistence_ejb_jar\META-  
INF\ejb-jar.xml
```

2. Enable database task persistence by changing the value of the following entry to 1:

```
<env-entry id="EnvEntry_1">  
  <description>Flag to use db persistence</description>  
  <env-entry-name>USE_DB_PERSISTENCE</env-entry-name>  
  <env-entry-type>java.lang.String</env-entry-type>  
  <env-entry-value>1</env-entry-value>  
</env-entry>
```

3. Save the ejb-jar.xml file.
4. Restart WebLogic.

Modify the Task Persistence EJB Descriptor for JBoss

Perform the following procedure to modify the task persistence EJB descriptor in a JBoss environment.

To modify the task persistence EJB descriptor

1. In a text editor, open the following EJB descriptor:

```
jboss_home\server\default\deploy\IdentityMinder.ear\taskpersistence_ejb_jar\META-INF\ejb-jar.xml
```

2. Locate the following lines in the file:

```
<env-entry id="EnvEntry_1">
  <description>Flag to use db persistence</description>
  <env-entry-name>USE_DB_PERSISTENCE</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>0</env-entry-value>
</env-entry>
```

3. Change the value of the env-entry-value from 0 to 1, as in the following example:

```
<env-entry id="EnvEntry_1">
  <description>Flag to use db persistence</description>
  <env-entry-name>USE_DB_PERSISTENCE</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>1</env-entry-value>
</env-entry>
```

4. Restart JBoss.

Modify the Task Persistence EJB Descriptor for WebSphere

To enable database task persistence for a WebSphere deployment, you change the value of the USE_DB_PERSISTENCE setting to 1 in the ejb-jar.xml file. The USE_DB_PERSISTENCE setting in a sample ejb-jar.xml file is the following:

```
<env-entry id="EnvEntry_1">
  <description>Flag to use db persistence</description>
  <env-entry-name>USE_DB_PERSISTENCE</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>1</env-entry-value>
</env-entry>
```

The ejb-jar.xml is in `was_im_tools_dir\CA\CA Identity Manager\WebSphere-ear\IdentityMinder.ear\taskpersistence_ejb_jar\META-INF\ejb-jar.xml`

`was_im_tools_dir`

Specifies the installed location of the Identity Manager Tools for WebSphere.

Note: You must package the EAR and deploy it to the WebSphere application server as described in the *Installation Guide*.

Index

%

- %ADMIN_ROLE_CONSTRAINT% • 125
- %DYNAMIC_GROUP_MEMBERSHIP% • 79
- %ENABLED_STATE% • 76
- %FIRST_NAME% • 76
- %FULL_NAME% • 76
- %GROUP_MEMBERSHIP% • 79
- %GROUP_ADMIN% • 79
- %GROUP_ADMIN_GROUP% • 79
- %GROUP_DESC% • 79
- %GROUP_NAME% • 79
- %IDENTITY_POLICY% • 76, 125
- %LAST_CERTIFIED_DATE% • 76, 125
- %LAST_NAME% • 76
- %MEMBER_OF% • 76
- %NESTED_GROUP_MEMBERSHIP% • 79
- %ORG_DESCR% • 80
- %ORG_MEMBERSHIP% • 76, 79, 80, 125, 127
 - determining directory structure • 82
- %ORG_MEMBERSHIP_NAME% • 76
- %ORG_MEMBERSHIP_NAME% • 79, 80
- %ORG_NAME% • 80
- %PASSWORD% • 76
- %PASSWORD_DATA% • 76
- %PASSWORD_HINT% • 76, 125
- %SELF_SUBSCRIBING% • 79, 127
- %USER_ID% • 76

A

- advanced settings screen
 - about • 169
- attributes
 - See also validation rules
 - zzz • 205
 - validating • 205
- AttributeValidationException • 219
 - resource bundles • 220
- AttributeValidator • 217
- audit data
 - about • 229
- Audit element
 - about • 231
- audit settings file
 - about • 230

- AuditEvent element
 - about • 233
- auditing
 - about • 229
 - audit settings file • 230
 - enabling for tasks • 242
- auditlevel parameter
 - in audit elements • 231
- authentication schemes • 287

C

- cascading style sheets
 - in skins • 184
- CDATA • 212, 223
- class name • 123
- class of an object • 123
- ClassFilters • 83
- colors
 - updating
 - skins • 190
- configuring
 - validation rules • 221
- custom settings
 - importing and exporting • 182

D

- data validation, see validation rules, validation rule sets • 205
- datasource parameter
 - in audit elements • 232
- default user interface
 - idm skin • 185
- directory configuration files
 - configuring self-subscribing gro • 84, 130
- directory.xml
 - configuring directory-level validation • 223
 - configuring self-subscribing groups • 84, 130
 - conventions in • 53
 - managed object attributes • 227
 - validation rule definition • 208
- directory-level validation
 - managed object attributes • 227
 - vs. task-level validation • 207
 - about • 206

- rule sets • 207
- regular expressions • 212
- displayname
 - in `ImsManagedObjectAttr` statements • 113
- dynamic groups
 - configuring • 84

E

- email templates
 - translating • 257
- Enabled field
 - for identity policies • 171
- enabled parameter
 - in Audit elements • 231, 233
- enabling
 - identity policies • 171
- error messages
 - directory-level validation • 220
 - JavaScript validation rules • 213, 215
 - regular expression validation rules • 219
 - task-level validation • 219
- eTrust Directory
 - configuration considerations • 73
- events
 - specifying email notifications for • 170
- EventState element
 - about • 239
- exceptions with validation rules • 219
- exporting
 - custom settings • 182

F

- fonts
 - updating
 - skins • 190
- forgotten password task
 - accessing • 155

G

- group name • 123
- GroupClassFilters • 83
- groups
 - configuring dynamic groups • 84
 - configuring nested groups • 84
 - configuring self-subscribing groups • 130

H

- head.jsp
 - description • 195

- hidden
 - in `ImsManagedObjectAttr` statements • 113
- horizontal skin
 - description • 185
- horizontal2 skin
 - description • 185

I

- `IAttributeValidator` • 217
- Identity Manager directories
 - viewing validation rule sets • 144
- Identity Manager environments
 - accessing • 155
 - auditing • 229
 - configuring an authentication sc • 287
- Identity Manager User Console
 - accessing • 155
- identity policies
 - configuring settings for • 171
- Identity Policy Well Known Attribute field • 171
- idm skin
 - description • 185
- images
 - in skins • 184
 - updating
 - skins • 189
- importing
 - custom settings • 182
- `ImsExceptions` resource bundle
 - messageid • 223
 - validation rules • 220
- `ImsManagedObject` • 108
- `ImsManagedObjectAttr` • 113, 223, 227
- index.jsp file
 - modifying for skins • 191, 196

J

- Java validation rules
 - `AttributeValidator` • 217
 - configuring directory-level validation • 223
 - configuring task-level validation • 222
 - deployment • 223
 - error messages • 219
 - `IAttributeValidator` • 217
 - implementation • 216
 - `TaskValidator` • 216
 - where referenced • 208
- JavaScript validation rules
 - configuring directory-level valida • 223

- configuring task-level validation • 222
- error messages • 213, 215
- implementation • 213
- source file location • 223
- where implemented or referenced • 208, 223
- jsp files
 - recompiling
 - for skins • 192

K

- key IDs
 - in resource bundles • 248
 - understanding • 248

L

- LdapMatchUserDn • 83
- localization
 - and email templates • 257
 - and Identity Manager APIs • 258
 - and skins • 258
 - and validation exceptions • 219, 220
 - determining which language to display • 260
- login pages
 - customizing • 287
- LogOffURI
 - about • 287

M

- managed object attributes
 - validating • 205
 - validation rule sets and • 227
- Management Console
 - accessing • 16
 - viewing validation rule sets • 144
- management data
 - about • 229
- maxlength
 - in ImsManagedObjectAttr statements • 113
- maxrows • 58
- messageid • 223
- modifying
 - index.jsp file
 - for skins • 191, 196
 - properties file
 - for skins • 190
- multivalued
 - in ImsManagedObjectAttr statements • 113

N

- Name
 - ImsManagedObject parameter • 111
- name parameter
 - in AuditEvent elements • 233
 - in EventState elements • 239
- nested groups
 - configuring • 84
- neteauto skin
 - description • 185

O

- Objecttype
 - ImsManagedObject parameter • 111
- org parameter
 - for self-subscribing groups • 130
- OrgClassFilters • 83

P

- parent skins • 188
- permission
 - in ImsManagedObjectAttr statements • 113
- physicalname
 - in ImsManagedObjectAttr statements • 113
- Policy Server
 - creating a skin response • 193
- PolicyClassFilters • 83
- PolicyResolution • 83
- prioritizing
 - skins • 193
- properties file
 - modifying for skins • 190
- properties files
 - creating for skins • 190
 - in skins • 184
- public tasks
 - accessing • 155

Q

- query • 123

R

- READONLY • 113
- READWRITE • 113
- realms
 - configuring authentication schemes • 287
- recompiling
 - jsp files

- for skins • 192
- Recursion Level field • 171
- regular expression validation rules
 - configuring directory-level • 223
 - configuring task-level val • 222
 - error messages • 219
 - example • 223
 - implementation • 212
 - where implemented • 208
- required
 - in `ImsManagedObjectAttr` statements • 113
- resource bundles
 - validation rules • 220
- rules, see validation rules, validation rule sets
 - 205

S

- `sAMAccountName` • 74
- SCC
 - setting an event's severity level • 239
- self-registration
 - accessing • 155
- self-registration tasks
 - accessing • 155
- self-subscribing groups
 - configuring • 130
- severity attribute
 - in `EventState` element • 239
- SiteMinder
 - configuring authentication schemes • 287
 - creating a skin response • 193
- skins
 - about • 183
 - about parent skins • 188
 - applying to users • 193
 - creating a properties file • 190
 - localizing • 258
 - prioritizing • 193
 - updating
 - colors • 190
 - fonts • 190
 - images • 189
- SQL select • 123
- SQL update • 123
- string values • 123
- StringRef
 - Java declaration • 216, 217
 - use in JavaScript • 213, 215
- system

- in `ImsManagedObjectAttr` statements • 113

T

- task screen fields
 - task-level validation • 222
- task-level validation
 - vs. directory-level validation • 207
 - about • 206
 - regular expressions • 212
- tasks
 - enabling auditing • 242
 - specifying email notifications for • 170
- `TaskValidator` • 216
- timeout • 58
- transformation • 205
- translating
 - email templates • 257
- type parameter
 - for self-subscribing groups • 84, 130

U

- URLs
 - for the Management Console • 16
 - for accessing Identity Manager environments • 155
- user
 - attributes • 123
 - name • 123
- User Console
 - customizing • 183

V

- validation rule sets
 - about • 207
 - managed object attributes • 227
 - order of execution • 208
 - rule set element • 223
 - viewing • 144
- validation rules
 - about • 205
 - configuring • 221
 - default • 209
 - definition basics • 208
 - exceptions • 219
 - implementation types • 212
 - `ImsExceptions` • 220
 - Java • 216
 - JavaScript • 213
 - pre-defined • 212

- regular expression • 212
- resource bundles • 220
- rule element • 223
- See also directory-level validation, task-level
 - 205
- transformation • 205
- types • 206
- validation, initiating • 228
- ValidationRule element • 223
- ValidationRuleSet element • 223
- validationscripts • 223
- value
 - Java validation • 216, 217
 - JavaScript validation • 213, 215
- value of a property • 123
- valuetype
 - in ImsManagedObjectAttr statements • 113
- viewing
 - User Console in multiple languages • 260

W

- wellknown
 - in ImsManagedObjectAttr statements • 113
- write access • 123
- WRITEONCE • 113