

# Deployment Guide

*Sun™ ONE Identity Server*

**Version 6.1**

816-6775-10  
December 2003

Copyright © 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents> and one or more additional patents or pending patent applications in the U.S. and in other countries.

THIS PRODUCT CONTAINS CONFIDENTIAL INFORMATION AND TRADE SECRETS OF SUN MICROSYSTEMS, INC. USE, DISCLOSURE OR REPRODUCTION IS PROHIBITED WITHOUT THE PRIOR EXPRESS WRITTEN PERMISSION OF SUN MICROSYSTEMS, INC.

U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

This distribution may include materials developed by third parties.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and in other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Java, Solaris, JDK, Java Naming and Directory Interface, JavaMail, JavaHelp, J2SE, iPlanet, the Duke logo, the Java Coffee Cup logo, the Solaris logo, the SunTone Certified logo and the Sun ONE logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon architecture developed by Sun Microsystems, Inc.

Legato and the Legato logo are registered trademarks, and Legato NetWorker, are trademarks or registered trademarks of Legato Systems, Inc. The Netscape Communications Corp logo is a trademark or registered trademark of Netscape Communications Corporation.

The OPEN LOOK and Sun(TM) Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Products covered by and information contained in this service manual are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

---

Copyright © 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. détient les droits de propriété intellectuels relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plus des brevets américains listés à l'adresse <http://www.sun.com/patents> et un ou les brevets supplémentaires ou les applications de brevet en attente aux Etats - Unis et dans les autres pays.

CE PRODUIT CONTIENT DES INFORMATIONS CONFIDENTIELLES ET DES SECRETS COMMERCIAUX DE SUN MICROSYSTEMS, INC. SON UTILISATION, SA DIVULGATION ET SA REPRODUCTION SONT INTERDITES SANS L'AUTORISATION EXPRESSE, ECRITE ET PREALABLE DE SUN MICROSYSTEMS, INC.

Cette distribution peut comprendre des composants développés par des tierces parties.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, Java, Solaris, JDK, Java Naming and Directory Interface, JavaMail, JavaHelp, J2SE, iPlanet, le logo Duke, le logo Java Coffee Cup, le logo Solaris, le logo SunTone Certified et le logo Sun[tm] ONE sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

Legato, le logo Legato, et Legato NetWorker sont des marques de fabrique ou des marques déposées de Legato Systems, Inc. Le logo Netscape Communications Corp est une marque de fabrique ou une marque déposée de Netscape Communications Corporation.

L'interface d'utilisation graphique OPEN LOOK et Sun(TM) a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui, en outre, se conforment aux licences écrites de Sun.

Les produits qui font l'objet de ce manuel d'entretien et les informations qu'il contient sont regis par la législation américaine en matière de contrôle des exportations et peuvent être soumis au droit d'autres pays dans le domaine des exportations et importations. Les utilisations finales, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes biologiques et chimiques ou du nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers des pays sous embargo des Etats-Unis, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exclusive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont regis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.

# Contents

<b>List of Figures</b> .....	<b>9</b>
<b>List of Tables</b> .....	<b>11</b>
<b>List of Code Examples</b> .....	<b>13</b>
<b>About This Guide</b> .....	<b>15</b>
Audience for This Guide .....	15
Identity Server 6.1 Documentation Set .....	16
Identity Server Core Documentation .....	16
Identity Server Policy Agent Documentation Set .....	17
Your Feedback on the Documentation .....	17
Documentation Conventions Used in This Guide .....	18
Typographic Conventions .....	18
Terminology .....	18
Related Information .....	19
<b>Chapter 1 Introduction</b> .....	<b>21</b>
What is Identity Management? .....	21
The Identity Management Infrastructure .....	22
The Life Cycle of an Identity Profile .....	23
Sun ONE Identity Server .....	24
Access Management .....	24
Single Sign-On (SSO) .....	24
Pluggable Authentication .....	24
Policy Evaluation .....	25
Federation Management .....	25
Liberty Alliance Project .....	25
Security Assertion Markup Language (SAML) .....	26
Identity Management .....	26
User Profile Management .....	26

Policy Configuration .....	26
Service Management .....	26
Auditing .....	27
Policy Agents .....	27
Identity Server Console .....	27
Programmatic Interfaces .....	27
Sun ONE Directory Server .....	28
Deploying Identity Server .....	28
Integrating Identity Server .....	28
Solaris™ Operating System .....	29
Windows 2000 Server .....	29
Linux .....	29
HPUX 11 .....	30
Deployment Road Map .....	30
Deployment Guide Chapters .....	31
Related Identity Server Documentation .....	32
<b>Chapter 2 Planning The Deployment .....</b>	<b>33</b>
Defining Resources .....	33
Human Resources .....	34
Executive Sponsors .....	34
Team Lead .....	35
Project Management .....	35
Systems Analyst .....	35
LOB Application Administrators .....	35
System Administrators .....	36
Independent Software Vendors .....	37
Third Party Affiliates .....	37
Funding .....	37
Setting Goals .....	38
Gathering Information .....	38
Business Processes .....	39
IT Infrastructure .....	40
Virtual Data .....	41
Evaluating Applications .....	42
Platform Information .....	43
Security Models .....	43
Lifecycle Of A Session .....	44
Customization And Branding .....	44
Categorizing Data .....	44
Mapping To Authentication .....	45
Mapping To Authorization .....	46
Building Timelines .....	47

Deployment Design .....	47
Proof-of-Concept .....	47
Early Adoption .....	48
General Participation .....	48
Production Environment .....	48
<b>Chapter 3 Identity Server Architecture .....</b>	<b>49</b>
Overview .....	49
Integration Points .....	50
Policy Agents .....	51
Web And Proxy Server Agents .....	51
J2EE Agents .....	51
Identity Server SDK .....	52
SSO API .....	52
Authentication API And Authentication SPI .....	53
Policy API .....	53
Identity Management SDK .....	53
Logging API and Logging SPI .....	53
Service Management SDK .....	53
Client Detection API .....	54
SAML SDK .....	54
Federation Management API .....	54
Functional Processes .....	54
Authentication and User Sessions .....	54
HTML Over HTTP(S) Interface .....	55
XML Over HTTP(S) Interface .....	56
Integrated Policy .....	57
Integrated Client Detection .....	58
CDSSO, SAML and Federation .....	58
CDSSO .....	58
SAML .....	59
Federation .....	59
Extending Identity Server .....	59
Web Containers .....	59
Multiple Directory Server Instances .....	60
LDAP Load Balancers .....	60
<b>Chapter 4 Pre-Deployment Considerations .....</b>	<b>61</b>
Deployment Options .....	61
Security .....	62
High Availability .....	62
Clustering .....	63

Scalability .....	63
Hardware Requirements .....	63
Software Requirements .....	64
Operating System Requirements .....	64
Patch Clusters for Solaris .....	65
Java™ Requirement .....	66
Resource Web Server Requirements .....	67
Web Browser Requirements .....	67
Understanding the Identity Server Schema .....	67
Marker Object Classes .....	72
Administrative Roles .....	72
Schema Limitations .....	74
People Containers .....	74
Only One Identity Server Organization Allowed .....	74
Unsupported Directory Trees .....	76
<b>Chapter 5 Deployment Scenarios .....</b>	<b>77</b>
Reference Deployment Scenario .....	77
To Install Multiple Identity Servers With ammultiserverinstall .....	78
Web Deployment .....	79
Java Application Deployment .....	80
Multiple JVM Environment .....	81
Replication Considerations .....	81
Configuring For Replication .....	82
Configuring With a Load Balancer .....	85
Replication Caveats .....	88
Implementing Federation Management .....	88
<b>Appendix A Installed Product Layout .....</b>	<b>89</b>
The Sun Java Enterprise System 2003Q4 Base Directory .....	89
The SUNWam Directory .....	90
/opt/SUNWam/agents/ .....	91
/opt/SUNWam/bin/ .....	91
/config ---> /etc/opt/SUNWam/config/ .....	92
/opt/SUNWam/console.war .....	94
/opt/SUNWam/docs .....	94
/opt/SUNWam/dtd .....	94
/opt/SUNWam/ldaplib .....	95
/opt/SUNWam/ldif .....	96
/opt/SUNWam/lib .....	96
/opt/SUNWam/locale .....	98
/opt/SUNWam/migration .....	100
/opt/SUNWam/password.war .....	100

/opt/SUNWam/public_html .....	100
/opt/SUNWam/samples .....	100
/opt/SUNWam/services.war .....	101
/opt/SUNWam/share .....	101
/opt/SUNWam/web-apps .....	102
<b>Appendix B The User Session Life Cycle .....</b>	<b>105</b>
Overview .....	105
The Request .....	106
The Authentication .....	107
The Session Token .....	109
The Policy .....	113
The Requested Page .....	115
Single Sign-On Requests .....	116
Thread One: Single Sign-On .....	116
Thread Two: Cross Domain Single Sign-On .....	120
Terminating a Session .....	123
<b>Appendix C Authenticate Against Active Directory .....</b>	<b>127</b>
Overview .....	127
Point to Existing LDAP Authentication Module .....	128
Create New Active Directory Authentication Module .....	128
Multiple LDAP Sub-Configurations .....	128
Setting Up Active Directory Authentication .....	128
Troubleshooting .....	131
Quick Access To Identity Server .....	131
Reconfigure Using Directory Server .....	131
<b>Appendix D Load Balancer Configuration .....</b>	<b>133</b>
Load Balancer Overview .....	133
Sticky Sessions .....	134
Resonate Central Dispatch Installation .....	134
Configuring The Load Balancer .....	135
To Configure Central Dispatch for setcookie .....	136
To Configure Identity Server for setcookie .....	141
To Configure Central Dispatch with Load Balancer Cookies .....	141
To Configure Identity Server with Load Balancer Cookies .....	143
Confirming The Configuration .....	144
<b>Appendix E Authenticate Against RADIUS Servers .....</b>	<b>145</b>
Overview .....	145
RADIUS Server Configuration .....	145

Identity Server Configuration .....	146
<b>Appendix F Installing in a chroot Environment .....</b>	<b>149</b>
Overview .....	149
Before Creating chroot .....	149
Creating a chroot Environment .....	150
Installing Identity Server Under chroot .....	155
Starting Identity Server In chroot .....	156
Identity Server Log Files In chroot .....	157
<b>Index .....</b>	<b>159</b>



# List of Figures

Figure 1-1	Building Blocks Of An Identity Management Solution . . . . .	23
Figure 2-1	Security Requirements Of Data And Services . . . . .	45
Figure 3-1	Identity Server 6.1 Functional Architecture . . . . .	50
Figure 4-1	Unmanageable Directory Tree . . . . .	75
Figure 4-2	One Organization Rule Exception . . . . .	75
Figure 4-3	Two Organizational Units Allowed . . . . .	75
Figure 4-4	Scenario 1: Three LDAP Organization Attributes not Allowed . . . . .	76
Figure 4-5	Scenario 2: Three LDAP Organization Attributes not Allowed . . . . .	76
Figure 5-1	Multiple Identity Servers Against One Directory Server . . . . .	78
Figure 5-2	Simple Web Deployment Scenario . . . . .	80
Figure 5-3	Java Application Deployment . . . . .	81
Figure 5-4	Single Supplier Replication . . . . .	83
Figure 5-5	Multi Supplier Configuration (also known as Multi-Master Replication) . . . . .	83
Figure 5-6	Multi-supplier Replication With Load Balancer . . . . .	86
Figure 5-7	Identity Server Configuration With Load Balancer . . . . .	134
Figure 5-8	Creating Nodes With Resonate . . . . .	136
Figure 5-9	Creating a new Virtual IP address . . . . .	137
Figure 5-10	Configuring HTTP Scheduling Rules . . . . .	138
Figure 5-11	Configuring Nodes With CDMaster . . . . .	139
Figure 5-12	Configuring a Cookie Persistence Scheduling Rule . . . . .	140



# List of Tables

Table 0-1	Where to Find Related Sun ONE Resources .....	19
Table 4-1	Recommended Patch List For Identity Server Installation .....	65
Table 4-2	Default And Dynamic Roles And Their Permissions .....	73
Table A-1	Identity Server Command Line Utilities .....	91
Table A-2	XML Service and Configuration Files .....	92
Table A-3	DTD Files in Identity Server .....	95
Table A-4	Shared Object Files .....	95
Table A-5	LDIF Files .....	96
Table A-6	Directories For Functional Samples .....	101
Table 5-1	Resonate Central Dispatch Terms Defined .....	135



# List of Code Examples

Code Example 4-1	ds_remote_schema.ldif . . . . .	68
Code Example 4-2	sunone_schema2.ldif . . . . .	71
Code Example 5-1	serverconfig.xml Replication Modification . . . . .	85
Code Example 5-2	serverconfig.xml Load Balancer Modification . . . . .	87
Code Example 5-3	GET Request Header . . . . .	106
Code Example 5-4	GET Response With Redirect Information . . . . .	106
Code Example 5-5	GET Request Redirected To Authentication Service . . . . .	107
Code Example 5-6	Authentication Form Returned To User . . . . .	107
Code Example 5-7	POST Credentials Returned To Identity Server . . . . .	108
Code Example 5-8	Redirect Back To Originally Requested Resource . . . . .	108
Code Example 5-9	Redirected GET Request Header With Token . . . . .	109
Code Example 5-10	POST Request For Naming Information . . . . .	109
Code Example 5-11	Response With Naming Information . . . . .	110
Code Example 5-12	POST Request To Session Service For Session Validation . . . . .	111
Code Example 5-13	Session Service Response Asserting A Session's Validity . . . . .	112
Code Example 5-14	POST Request For Policy Information . . . . .	113
Code Example 5-15	Allow Policy Response . . . . .	114
Code Example 5-16	Logging Request From Policy Agent . . . . .	115
Code Example 5-17	Return Response Notifying Agent Of Log . . . . .	116
Code Example 5-18	Agent Allows Access To Requested Page . . . . .	116
Code Example 5-19	Second Request With A Valid Session Token . . . . .	117
Code Example 5-20	POST Request For Naming Service . . . . .	117
Code Example 5-21	POST Request To Session Service . . . . .	118
Code Example 5-22	POST Request To Policy Service . . . . .	118
Code Example 5-23	Response From Policy Service Negating Access . . . . .	119
Code Example 5-24	POST Request To Logging Service . . . . .	119
Code Example 5-25	HTML Page With Access Denied Message . . . . .	120

Code Example 5-26	GET Request For Protected Application In Different DNS Domain . . . . .	120
Code Example 5-27	Redirect To The CDSSO Controller Service Via The Browser . . . . .	121
Code Example 5-28	HTTP Redirect From Browser With Session Token . . . . .	121
Code Example 5-29	POST Reply To The Browser . . . . .	121
Code Example 5-30	Browser POST To Policy Agent . . . . .	122
Code Example 5-31	User Allowed Access To HTML Page With New Cookie . . . . .	123
Code Example 5-32	GET Request For Logout Service . . . . .	123
Code Example 5-33	Successful Logout HTML Page Returned To User . . . . .	124
Code Example 5-34	POST For Session Notification . . . . .	124
Code Example 5-35	RADIUS User Entry . . . . .	146
Code Example 5-36	RADIUS Client Entry . . . . .	146

# About This Guide

The *Sun™ ONE Identity Server Deployment Guide* offers technical information and simple scenarios to get an organization started a deployment of Sun ONE Identity Server. It also contains general details about identity management and how to select a deployment team. This preface to the *Deployment Guide* contains the following sections:

- Audience for This Guide
- Identity Server 6.1 Documentation Set
- Documentation Conventions Used in This Guide
- Related Information

## Audience for This Guide

This *Deployment Guide* is intended for use by IT administrators who implement an integrated identity management and web access platform using Sun ONE servers and software. It is recommended that administrators understand the following technologies:

- Lightweight Directory Access Protocol (LDAP)
- Java™
- JavaServer Pages™ (JSP)
- HyperText Transfer Protocol (HTTP)
- HyperText Markup Language (HTML)
- eXtensible Markup Language (XML)

Because Sun ONE Directory Server is used as the data store in an Identity Server deployment, administrators should also be familiar with the documentation provided with that product. The latest Directory Server documentation can be accessed online.

## Identity Server 6.1 Documentation Set

The Identity Server documentation set is separated into two core sets of manuals: the Sun ONE Identity Server 6.1 core application manuals and the Sun ONE Identity Server Policy Agents books.

### Identity Server Core Documentation

The Identity Server documentation set contains the following titles:

- *Product Brief* provides an overview of the Identity Server application and its features and functions.
- *Migration Guide* provides details on how to migrate existing data and Sun ONE product deployments to the latest version of Identity Server. For instructions on installing Identity Server, see the *Sun Java Enterprise System 2003Q4 Installation Guide*.
- *Administration Guide* describes how to use the Identity Server console as well as manage user and service data via the command line.
- *Customization and API Guide* documents how to customize an Identity Server installation. It also includes instructions on how to augment the application with new services using the public APIs.
- *Deployment Guide* (this book) provides information on planning an Identity Server deployment within an existing information technology infrastructure.
- The *Release Notes* will be available online after the product is released. They gather an assortment of last-minute information, including a description of what is new in this current release, known problems and limitations, installation notes, and how to report issues with the software or the documentation.

Updates to the *Release Notes* and links to modifications of the core documentation can be found on the Identity Server page at the Sun ONE documentation web site. Updated documents will be marked with a revision date.



# Identity Server Policy Agent Documentation Set

Policy agents for Identity Server are available on a different schedule than the server product itself. Therefore, the documentation set for the policy agents is available outside the core set of Identity Server documentation. The following titles are included in the set:

- *Web Policy Agents Guide* documents how to install and configure an Identity Server policy agent on various web and proxy servers. It also includes troubleshooting and information specific to each agent.
- *J2EE Policy Agents Guide* documents how to install and configure an Identity Server policy agent that can protect a variety of hosted J2EE applications. It also includes troubleshooting and information specific to each agent.
- The *Release Notes* will be available online after the set of agents is released. There is generally one *Release Notes* file for each agent type release. The *Release Notes* gather an assortment of last-minute information, including a description of what is new in this current release, known problems and limitations, installation notes, and how to report issues with the software or the documentation.

Updates to the *Release Notes* and modifications to the policy agent documentation can be found on the Policy Agents page at the Sun ONE documentation web site. Updated documents will be marked with a revision date.

## Your Feedback on the Documentation

Sun Microsystems and the Identity Server technical writers are interested in improving the documentation and welcome any comments and suggestions. Please email these comments to [docfeedback@sun.com](mailto:docfeedback@sun.com).

# Documentation Conventions Used in This Guide

In the Identity Server documentation, certain typographic conventions and terminology are used. These conventions are described in the following sections.

## Typographic Conventions

This book uses the following typographic conventions:

- *Italic type* is used within text for book titles, new terminology, emphasis, and words used in the literal sense.
- Monospace font is used for sample code and code listings, API and language elements (such as function names and class names), filenames, pathnames, directory names, HTML tags, and any text that must be typed on the screen.
- *Italic serif font* is used within code and code fragments to indicate variable placeholders. For example, the following command uses *filename* as a variable placeholder for an argument to the `gunzip` command:

```
gunzip -d filename.tar.gz
```

## Terminology

Below is a list of general terms used in the Identity Server documentation set:

- *Identity Server* refers to Identity Server and any installed instances of the Identity Server software.
- *Policy and Management services* refers to the collective set of Identity Server components and software that are installed and running on a dedicated deployment container such as a web server.
- *Directory Server* refers to an installed instance of Sun ONE Directory Server.
- *Application Server* refers to an installed instance of Sun ONE Application Server.
- *Web Server* refers to an installed instance of Sun ONE Web Server.
- *IdentityServer\_base* is a variable place holder for the home directory where you have installed Identity Server.
- *DirectoryServer\_base* is a variable place holder for the home directory where you have installed Sun ONE Directory Server.

- *ApplicationServer\_base* is a variable place holder for the home directory where you have installed Sun ONE Application Server.
- *WebServer\_base* is a variable place holder for the home directory where you have installed Sun ONE Web Server.
- *Web container that runs Identity Server* refers to the dedicated J2EE container (such as Web Server or Application Server) where the Policy and Management Services are installed.

## Related Information

In addition to the documentation provided with Identity Server, there are several other sets of documentation that might be helpful. Table 0-1 lists these and additional sources of information.

**Table 0-1** Where to Find Related Sun ONE Resources

Information or Resource	Internet Location
Directory Server documentation	<a href="http://docs.sun.com/coll/S1_DirectoryServer_52">http://docs.sun.com/coll/S1_DirectoryServer_52</a>
Web Server documentation	<a href="http://docs.sun.com/coll/S1_websvr61_en">http://docs.sun.com/coll/S1_websvr61_en</a>
Web Proxy Server documentation	<a href="http://docs.sun.com/prod/s1.webproxys#hic">http://docs.sun.com/prod/s1.webproxys#hic</a>
Sun ONE Download Center	<a href="http://www.sun.com/software/download/">http://www.sun.com/software/download/</a>
Sun ONE Technical Support	<a href="http://www.sun.com/service/sunone/software/index.html">http://www.sun.com/service/sunone/software/index.html</a>
Sun ONE Professional Services Information	<a href="http://www.sun.com/service/sunps/sunone/index.html">http://www.sun.com/service/sunps/sunone/index.html</a>
Sun Enterprise Services, Solaris Patches and Support	<a href="http://sunsolve.sun.com/">http://sunsolve.sun.com/</a>
Developer Information	<a href="http://developers.sun.com/prodtech/index.html">http://developers.sun.com/prodtech/index.html</a>

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

## Related Information

# Introduction

Sun™ ONE Identity Server provides an infrastructure for an organization to administrate the processes used to manage the digital identities of customers, employees and partners who use their web-based services and non web-based applications. Because these resources may be distributed across a wide range of internal and external computing networks, attributes, policies and controls are defined to manage access. This introductory chapter describes the basic principles behind a deployment of Identity Server. It contains the following sections:

- What is Identity Management?
- Sun ONE Identity Server
- Deploying Identity Server

## What is Identity Management?

Modern enterprises maintain an advanced information technology infrastructure to facilitate the management of its daily operations. Integral parts of this infrastructure might include:

- Network servers running disparate operating systems
- Information data stores
- Human resources, payroll and contract management systems
- Line-of-business applications for accounting, supply chain management, and resource planning
- Customer relationship management (CRM) systems that integrate sales, marketing, customer service, field support and other client-related functions
- E-commerce applications for shopping, and secure credit card transactions

Because they are deployed individually, each of these systems separately tracks users, controlling what they can and cannot see and do. This tracking process generally includes the management of *identity data* such as a personal profile, authentication information and access controls. Identity management simplifies the administration of this duplicated and oftentimes contrary data.

## The Identity Management Infrastructure

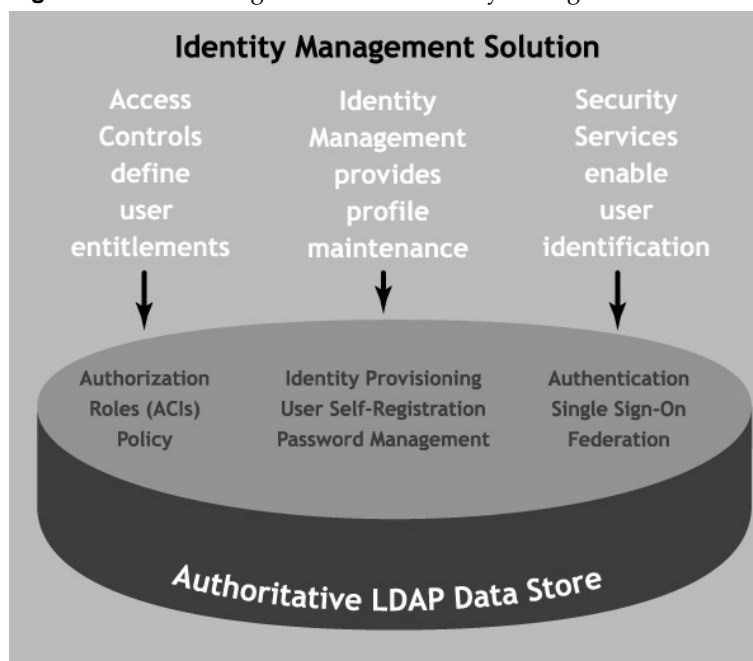
Implementing a single infrastructure to manage all users across an enterprise is the objective of an identity management system. One identity management system simplifies the administration of user profiles by eliminating repetition and therefore maintaining consistency. One system also streamlines, simplifies and automates the identity management processes. The building blocks of an identity management system include:

- Identity administration provides an infrastructure to support the creation and maintenance of identities and their corresponding attributes, credentials and entitlements. This function includes:
  - Identity provisioning (creation, modification and deletion of profiles)
  - User self-registration
  - Password management and password reset
- Security services enable user identification to remain consistent across a network. This function includes:
  - Authentication (proving the identity is who they claim to be)
  - Single sign-on function (authenticate once to access many resources)
- Access controls define a user's entitlements (how and when they can use disparate resources). These controls tend to specify the user's role within the organization. By creating and applying centralized policies and roles, an organization can delegate the responsibilities of its customers, employees and partners.
  - Authorization (determines whether an identity can perform the requested action)
  - Policy (defines authorization for access to protected resources)
  - Access control instructions (defines authorization for access to identity data)

- Federation services provide authentication and authorization between independent information systems.
- A corporate LDAP directory acts as the authoritative data store for all identity information as well as the configuration information for the identity management system itself.

These building blocks are further illustrated by Figure 1-1.

**Figure 1-1** Building Blocks Of An Identity Management Solution



## The Life Cycle of an Identity Profile

The challenges of identity management can be summed up by taking into account the life cycle of a typical identity profile. The three stages of an identity within an organization would be:

### 1. Creating A Profile

An identity profile is created when a user joins the organization. The profile might include personal information, employment data, password information and defined access privileges.

## 2. Maintaining The Profile

After setup, profiles must be managed. This might include modifying the profile data, maintaining policies for resource access, or updating access control instructions.

## 3. Disabling The Profile

When a user leaves, their profile needs to be flagged as such, and their access to system resources disabled.

# Sun ONE Identity Server

Sun ONE Identity Server is a package of integrated, standards-based middleware that provides web services to support access management, federation, and identity administration. This makes Identity Server a total identity management solution, integrating the ability to create and maintain user profiles with security processes, access management tools and a directory for data storage. These capabilities enable an organization to deploy a comprehensive system that protects their resources, and information as well as securely deliver their web-based applications.

## Access Management

Access management provides a common authentication and authorization infrastructure to replace ad hoc and application-specific authentication and authorization methods. From a central point of administration, organizations can provide policy-based control of access to multiple services. The collection of access management services in Identity Server provide the following functionalities.

### Single Sign-On (SSO)

Single sign-on functionality enables a user to authenticate once yet gain access to multiple resources. Identity Server supports SSO for web-based applications, and provides programmatic interfaces to integrate the SSO functionality into applications that are not web-based.

### Pluggable Authentication

The Java™ Authentication and Authorization Services-based (JAAS) authentication framework supports a variety of pluggable authentication modules including LDAP, Remote Authentication Dial-In User Service (RADIUS), X.509 digital certificates, SecureID®, SafeWord®, UNIX® (PAM-based), Windows® NT,



HTTP Basic Authentication, Anonymous, and Self-registration. The framework also allows for the development of custom authentication modules using the provided authentication service provider interfaces (SPI). Authentication can be configured to support the needs of a variety of organizations, roles, or users simultaneously in the same system, and supports multi-factor chained configurations. Multi-level authentication allows resources to be assigned a different level of required authentication based on the sensitivity of the data or service. The Authentication Service can be accessed via web-based, Java, C, and XML interfaces.

## Policy Evaluation

The Policy Service allows centralized configuration and evaluation of access management rules that can be mapped onto a variety of role and grouping mechanisms. Policy constraints such as IP address, day and time or custom conditions can be applied to a policy and evaluated at runtime.

# Federation Management

Because the Internet is fast becoming the prime vehicle for business, community, and personal interactions, it has become necessary to fashion a system for users to aggregate their various account identities, enabling them to have one network identity. This system is identity federation. Identity federation allows a user to associate, connect, or bind multiple Internet service providers' local identities. One network identity allows users to log in to one service provider's site and then move to an affiliated site, without having to reauthenticate or reestablish their identity. Identity Server provides full implementations of Liberty 1.1, and SAML 1.0. This includes complete profile implementations, as well as SDK support for custom integration. Multi-hosting of Liberty identity and service providers is provided.

## Liberty Alliance Project

Federation management provides a way to view, manage, and configure the metadata pertaining to authentication domains and providers. The Liberty Alliance Project, which was forged to make identity federation a reality, is comprised of more than two billion customers and 138 member companies representing a wide variety of industries. Its mission is to address the problem of fragmented identities by delivering and supporting a federated network identity solution that enables single sign-on for consumers and business users. Thus, a Liberty-enabled application can federate (or link) its user accounts with those of another Liberty-enabled application, and accomplish single sign-on between the two applications. Identity Server implements the Liberty Alliance Project's specifications.

## Security Assertion Markup Language (SAML)

SAML is a key enabler of business-to-business infrastructure. An application can use the SAML API integrated into Identity Server to exchange security information and execute business transactions with other trusted applications. An end user can employ a web browser to authenticate to Identity Server, then seamlessly access external URLs at trusted sites via an intersite transfer URL. Developers can use the SAML API in their applications to exchange authentication, authorization, and attribute information between trusted external applications.

## Identity Management

Identity management itself provides an extensible browser-based interface that allows for user provisioning, policy configuration, and service management. The Identity Server console allows centralized identity management with a single interface but, can also be delegated to other administrators, such as local group managers and external partners, or even to end users.

### User Profile Management

Simply, user profile management is the creation and deletion of identity profiles. But, it also entails delegating the management of those profiles to the administrators that know them as well as offering a self-service component where users can subscribe to a service or application, create a new user account and manage their own profiles (password changes, updating home addresses, et.al.).

### Policy Configuration

Policy configuration is the definition of the rules that are evaluated during access authorization. Delegation allows top-level administrators to distribute the configuration and management of policies to individuals at all levels of the organization ensuring that it is entrusted to people with authority over the resources.

### Service Management

Service management allows the configuration, registration and administration of web services and their corresponding attributes. Identity Server also provides an interface for the services that it uses for its own administration.

## Auditing

Administrators can use highly-configurable logging functions to generate detailed reports on user activity, traffic patterns, and authentication and authorization violations. These functions can also be used to perform security-level audits on resource access. Message Authentication Code (MAC) and digital signature-based log security detects any tampering with log or audit records. A debug function can also be enabled.

## Policy Agents

Access control in Identity Server is enforced using policy agents, which protect content on the designated web servers, application servers, and proxy servers from unauthorized intrusions. Identity Server supports both policy agents that protect web and proxy servers at the URL level as well as the Java™ 2 Platform, Enterprise Edition (J2EE) policy agents that enforce access on Java technology-enabled application servers.

## Identity Server Console

The Identity Server console is a browser-based interface for creating, managing, and monitoring the identities, services, and policies configured throughout an Identity Server deployment. It is built with Sun ONE Application Framework, a J2EE framework used to help developers build functional web applications. XML files, JavaServer Pages™ (JSP) and Cascading Style Sheets (CSS) are used to define the look of the HTML pages.

## Programmatic Interfaces

Non-graphical interfaces include the APIs, SPIs, and command line tools used to extend and customize Identity Server and allow other applications to access its functionality. More information on the APIs and SPIs can be found in “Identity Server SDK” on page 52 and in the Sun ONE Identity Server *Customization And API Guide*. Additional information on the command line tools can be found in the Sun ONE Identity Server *Administration Guide*.

## Sun ONE Directory Server

Sun ONE Directory Server acts as the integrated data repository for storing identity, policy, configuration and service information.

## Deploying Identity Server

Identity Server is designed with an open-standards platform that can be used to integrate its authentication, authorization, single sign-on, policy, identity and administration capabilities with existing infrastructures. Its functions are delivered as a collection of Java servlets, JavaBeans™ and JSP that run inside the Java Virtual Machine (JVM) of the web container and can access the API and various server frameworks. Integrating Identity Server into a corporate infrastructure can accomplish the following tasks:

- Eliminate ad hoc or proprietary utilities.
- Achieve secure authentication, access control and auditing across multiple web and application services.
- Implement centralized administration of identities with delegation capabilities [using configured levels of access control instructions (ACIs)] that allows an organization to:
  - Delegate internal identity management to non-IT employees.
  - Delegate external identity management to the partner or supplier employees.
- Configure the centralized policy framework to provide an authorization functionality to currently running applications and newly deployed services.
- Integrate federation management with support for both the Liberty Alliance specifications, v.1.1 and SAML.

## Integrating Identity Server

Identity Server is a complete identity management system but most likely organizations have already implemented aspects of an identity management system. For example, they might have already deployed a directory server or a web container. To allow Identity Server to interoperate with other systems, a policy agent must be downloaded and installed on the protected server. New agents are being developed and released concurrently with the development and release of

Identity Server. As of this release of Identity Server 6.1, the following operating systems have agents that can be downloaded from the Sun Microsystems' Web, Portal & Directory Servers Download Center. The Download Center page itself contains a more complete and up-to-date listing of released policy agents.

---

**CAUTION** Policy agents for Identity Server are scheduled for release outside the release of the product itself. This list of policy agents has most probably grown since the publication of this guide. See the the Sun Microsystems' Web, Portal & Directory Servers Download Center for a more up-to-date list of policy agents.

---

## Solaris™ Operating System

- Sun ONE Web Server 6.0
- Sun ONE Web Server 4.1
- Sun ONE Web Proxy 3.6
- Apache 1.3.27
- Lotus Domino 6.0
- Sun ONE Application Server 7.0 (J2EE)
- BEA WebLogic 6.1 SP2 (J2EE)
- BEA WebLogic 7.0 SP2 (J2EE)
- IBM WebSphere 5.0 (J2EE)
- PeopleSoft 8.3, 8.4 and 8.8 (J2EE)

## Windows 2000 Server

- Sun ONE Web Server 6.0
- Lotus Domino 6.0
- Microsoft IIS 5.0
- Sun ONE Application Server 7.0 (J2EE)
- BEA WebLogic 6.1 SP2 (J2EE)

## Linux

- Apache 1.3.27
- Sun ONE Application Server 7.0 (J2EE)

## HPUX 11

- Sun ONE Web Server 6.0
- PeopleSoft 8.3, 8.4 and 8.8 (J2EE)
- BEA WebLogic 6.1 SP2 (J2EE)

## Deployment Road Map

Mapping out your Identity Server integration is imperative to ensuring its success. This will include collecting information concerning hardware, currently deployed applications, identity data and access hierarchy. Identity Server deployment can be broken down into the following phases:

1. Identify business objectives such as:
  - Increase operational efficiency
  - Assure data security
  - Assure continued productivity by:
    - Understanding the scope and relationships within the organization.
    - Analyzing the behavioral changes needed to support the business objectives.
2. Develop a high-level technology analysis and map it to the business objectives by:
  - Listing technology services.
  - Listing tools needed to meet business objectives.
3. Define initiatives for each technology service such as:
  - Storing employee history and data accumulated through personalization.
  - Accomplishing password synchronization and identity administration through identity management.
  - Realizing enterprise security through the development of role strategies.
4. Prioritize initiatives based on:
  - Statistical accuracy
  - Predictability

- Scope
- Cost
- Impact
- Complexity
- Behavior
- Infrastructure
- Benefit
- Support
- Dependencies

## Deployment Guide Chapters

The following chapters in this guide follow the phases detailed in the Deployment Road Map:

- Chapter 2, “Planning The Deployment”—defines a methodology (including goals and challenges) for assessing the current state of your organization’s identity management solution and defining future requirements.
- Chapter 3, “Identity Server Architecture”—provides a high-level architectural overview of all components of the Identity Server product.
- Chapter 4, “Pre-Deployment Considerations” will help to analyze specific requirements, including hardware, sources of data and technical expertise.
- Chapter 5, “Deployment Scenarios” details simple scenarios for planning a topology and deploying the application.

Additional appendices have been added to the *Deployment Guide* for further edification. They include:

- Appendix A, “Installed Product Layout” details the directories and files created during the installation of Identity Server.
- Appendix B, “The User Session Life Cycle” details the session objects used to track user interaction with web applications across multiple HyperText Transfer Protocol (HTTP) requests.
- Appendix C, “Authenticate Against Active Directory” contains information on how authenticate users against a Microsoft Active Directory®.

- Appendix E, “Authenticate Against RADIUS Servers” contains information on how to authenticate users against a Remote Authentication Dial-In User Service (RADIUS) server.
- Appendix F, “Installing in a chroot Environment” contains information on how to install Identity Server in a chroot environment, preventing malicious programs from accessing the real root file system.

## Related Identity Server Documentation

Additional information on Identity Server can be found in the following manuals:

- Installation—For installation information, see the *Sun Java Enterprise System 2003Q4 Installation Guide*.
- Migration—For information on migrating existing data and updating previous versions of Identity Server, see the *Sun ONE Identity Server Migration Guide*.
- Administration—For information on how to use the console and administer an Identity Server deployment, see the *Sun ONE Identity Server Administration Guide*.
- Customization—For information on how to customize the application, see the *Sun ONE Identity Server Customization And API Guide*.



# Planning The Deployment

Sun™ ONE Identity Server is a complex, distributed identity management system that, when properly deployed, secures access to a wide variety of data and services spanning an enterprise's organizational boundaries. To ensure proper control over corporate resources, appropriate planning of the deployment process is required. This chapter offers information on how to plan the deployment. It contains the following sections:

- Defining Resources
- Setting Goals
- Gathering Information
- Evaluating Applications
- Categorizing Data
- Building Timelines

## Defining Resources

Since an identity management solution touches a broad variety of systems throughout an organization, proper Identity Server deployment requires a variety of resources. The following corporate resources will be involved/required in the deployment process.

## Human Resources

It is important to be savvy to the various business and political relationships within an organization. A team of individuals should be assembled with a direct and/or matrixed reporting structure. Typically, Identity Server deployments have small teams that might consist of a project manager and a couple of dedicated System Administrators. These report to the Team Lead and further up to an owner who has responsibility across a number of related projects and often reports directly to an executive sponsor. This is often augmented by virtual team members consisting of Sun Professional Services resources, and LOB Application Administrators which phase in and out as required. While this may or may not meet your exact needs, it does represent a fairly typical deployment team model. Although not necessarily distinct individuals, the following abstract technical roles representing various skill sets further define a typical Identity Server deployment team.

### Executive Sponsors

Successful identity management deployments traditionally cross organizational and political boundaries. This requires buy-in and support from those setting direction for the company. It is critical that executive sponsorship be in place. Planning meetings are an important process for gaining insight from those with a vested interest in the deployment. As the project plan is developed, ensure that its deliverables are inline with the goals of the company as a whole. For example, if cost reduction is a core business driver, collect statistics on current identity management costs, i.e.: what is the current cost of using the help desk for password resets? Having tangible statistics available will help define a concrete ROI as the deployment team attempts to shore up executive support. Other company issues that might be relevant include:

- Who benefits from the identity management deployment?
- What organizational problems does an identity management solution solve?
- How does the company address internal issues that may slow the deployment?

---

**NOTE**

It is often required that identity management concepts and the value of an Identity Server deployment be sold. A business and technology *evangelist* can passionately sell the new infrastructure to executives, helping to drive demand for integration and aid in the acceptance and ultimate success of the infrastructure changes.

---

## Team Lead

One person should be chosen as the party responsible for the project's success. This *team lead* must be clearly in charge and have the authority to make the project's goals happen. This may be a logically distributed role, perhaps between a technical lead, a project manager, and an executive head. However this role is defined, their goal is to show continued progress and demonstrated success throughout the deployment process to maintain executive sponsorship.

## Project Management

This individual is responsible for the coordination of timelines. They will maintain a schedule that correlates the availability of services, support provided by the core IT group, and the integration of the various line-of-business (LOB) applications. This person must be a strong communicator and politically savvy. They must balance the needs of the internal customers with the availability of resources in order to support new applications joining the environment.

---

**NOTE**

A line-of-business application is one vital to running an organization. They are generally large programs with capabilities that tie into databases and database management systems. They can include accounting, supply chain management, and resource planning applications. Increasingly, LOB applications are being connected with network applications that have user interfaces and with personal applications such as e-mail and address books.

---

## Systems Analyst

This individual is responsible for assessment and categorization of the various data and services to be integrated into the Identity Server deployment. The systems analyst interviews the LOB application owners and gathers details on technical requirements including platform, architecture, and its deployment schedule. With this information, the systems analyst formulates a plan on how the application will be integrated into the deployment in order to meet their customer's requirements. They must be an IT generalist, with broad knowledge of various application architectures and platforms. Detailed knowledge of Identity Server architecture, services, agents, and APIs is required.

## LOB Application Administrators

These individuals are responsible for integration of the Identity Server policy agents, or policy enforcement point, into their application. They must clearly communicate the LOB application's architecture, its integration points, and appropriate schedules. They are typically responsible for defining the access control model represented in Identity Server policies. This individual might

perform custom programming to enhance the integration between Identity Server and their application (for example, session coordination). Finally, they are generally responsible for QA and the regression testing of their application within the newly-deployed environment. This individual is a technical specialist with intimate knowledge of, and control over, the LOB application.

## System Administrators

It is critical that appropriate resources are in place to deploy and maintain the availability of Identity Server. System administrators are required at the following levels. Additional administrators might also include a web container administrator who is responsible for the deployment and performance of the software container in which Identity Server is deployed.

### *Identity Server Administrator*

This individual is responsible for the deployment and maintenance of Identity Server. Typically a dedicated resource, this individual assures the availability of the common services, provides necessary enhancements to the infrastructure in general and configures policies and roles in particular. They typically help support integration efforts by developing guidelines, and offer technical support to LOB Application Administrators. An understanding of Java, XML, LDAP, HTTP, and web application architectures is critical.

### *Directory Server Administrator*

Corporate directory services used for authentication and authorization are often already managed by a group within the organization before the Identity Server deployment is even considered. This individual is responsible for the availability of the directory services, as well as for accepting and integrating additions or modifications to the currently defined LDAP schema and identity data; changes that might be required to support the identity management infrastructure.

### *Hardware/Datacenter/Network Administrator*

Large organizations typically find economies of scale by separating hardware, operating system, datacenter and/or network administration from middleware administration. If this is the case in your company, it is essential that there is clear communication between these various administrators. It may be critical to the deployment's success to have access to certain machines, or to establish certain network configurations; keeping these individuals aware of project milestones and requirements will facilitate a smooth rollout.

## Independent Software Vendors

Sun Microsystems and other independent software vendors (ISV) are critical partners in the successful deployment of Identity Server. Purchasing packaged software allows an enterprise to diminish and distribute the cost and risk of software development across multiple organizations.

---

**NOTE** An independent software vendor makes and sells software products that can run on one or more types of computer hardware or operating system platforms. The companies that make the platforms (IBM, Hewlett-Packard, Apple, Microsoft, etc.) encourage and lend support to ISV. Sun Microsystems makes platforms and software products.

---

It is in the best interest of all parties involved for ISV to develop cooperative relationships and drive successful deployments. Engage Sun Professional Services and other ISV to help bootstrap the project, and impart knowledge they have gained from previous Identity Server deployments. Using Professional Services, as well as an open discussion with your account team (who can act as an intermediary between Identity Server engineers and your deployment team) will help insure your investment and a successful deployment.

## Third Party Affiliates

If you are planning on leveraging the Federation Management capabilities of Identity Server, you will be collaborating with external partners and third party affiliates. Consider an initial deployment of this functionality in conjunction with your own internal deployment. In this case, it is important to involve the LOB application that owns the business functionality which will be delivered and maintain communication with the technical resources of all parties. Your legal counsel can also help to establish a level playing field between involved parties.

## Funding

The core IT group is often responsible for the cost of the deployment project. In fact, it is common to have internal funds transferred from an LOB application to the core group in order to fund portions of the identity management project. But, even when a single LOB application group is providing initial funding, the needs of the larger organization should be balanced with the needs of the funding group.

## Setting Goals

By setting goals, an organization is defining where it wants to be after the deployment of Identity Server is finished. The deployment strategy is to plan a roadmap for reaching these objectives and move towards it. Goals are created by defining the expectations of all involved and getting approval early in the process.

In general, an identity management solution enhances security and improves infrastructure manageability while decreasing costs. More specifically, some common goals (and their benefits) that Identity Server allows an organization to meet include:

- Implementation of a scalable infrastructure to meet the expected increase in digital identities (employees, partners, and customers).
- Consolidation of the creation and management of identity profiles with each group controlling their own data.
- Cost reduction through vendor consolidation, user self-management and related administration costs.
- Improvement of security through immediate identity profile termination.
- Improved transparency of security models and access rights.
- Condensing timeframes required for access to critical systems.
- Removal of user rights to critical systems as roles or affiliation within the organization change.

Ultimately, these goals, combined with an understanding of the motivation of all groups involved and information gleaned from a site survey, can be used to design an infrastructure for the deployment. In addition, they can be used throughout the deployment process to keep interested parties engaged and encourage project endorsement.

## Gathering Information

A site survey can be used to gather information about the applications and data stores that will be integrated into the deployment. In addition, these departmental interviews help to forge an understanding of the motivation of the groups involved by defining their particular functions and goals. Once collected, the information can solidify buy-in from the executive sponsors as well as serve as a design blueprint. The following groups of individuals can help in a site survey.

- Users provide feedback on the applications with which they work on a daily basis.
- Human resources provides information on hiring and termination processes.
- Support personnel offer insight into problems that cross organizational boundaries.
- Application administrators and developers can provide technical information on the line-of-business (LOB) applications to be integrated into the deployment.
- Network administrators have knowledge of the organization's technical baseline for performance and standards.

An initial survey might include gathering information on Business Processes, the IT Infrastructure and Virtual Data.

## Business Processes

The business processes are the procedures that disparate groups in the organization define to do their job. Processes can include procedures for:

- Issuing payroll.
- Purchasing and accounts payable.
- Authorizing employee travel.
- Departmental budgeting.
- Terminating employees.

It is imperative to assess these processes as they are generally supported by the applications used by each business unit. Things to consider include:

- Do the current processes cause delays?
- Are there a number of different processes that perform the same function?
- Can processes be standardized across business unit boundaries?
- How complex are the processes? Can they be consolidated and/or simplified?
- Can the current processes handle organizational changes?

Any changes to be made to the processes should be initiated prior to the beginning of the deployment.

## IT Infrastructure

The IT infrastructure includes all the hardware servers, operating systems and integrated applications that will be integrated into the Identity Server deployment.

- What applications will leverage Identity Server?

Applications might include critical internal applications such as those for human resources and accounting, or less-critical employee portals. Also leveraging the functionality of Identity Server might be external business-to-business applications that deal with both confidential financial information and less confidential sales material, or business-to-consumer *shopping carts* that are concerned with credit card data and purchase histories.

- What systems will leverage Identity Server?

This includes the hardware on which the applications are being deployed as well as their operating systems. An Identity Server deployment, at the minimum, includes a web container to run the application, a Sun ONE Directory Server (or existing data store) and the Sun ONE Identity Server application itself. It will likely also include additional hardware servers which run their own web containers. They contain corporate resources and on which Identity Server policy agents may be installed for purposes of security.

- What Identity Server services will each department leverage?

This includes the default and custom services integrated within Identity Server. Role and policy strategies will have to be mapped and defined for each department. Authentication modules need to be assessed and custom services, if any, need to be developed.

Additionally, more technical considerations might include:

- Are there incompatibilities in the infrastructure?
- Does the current system experience slowdowns? Down time?
- Are the applications sufficiently secure?
- Are there virus control procedures?
- Can applications be customized based on user entitlements?

More information can be found in “Evaluating Applications” on page 42.



## Virtual Data

Virtual data is a catch-all phrase for the profiles that will access, the configurations that will be accessible from, and the data that will be secured by, Identity Server. This includes, but is not limited to, user profiles (employees, customers, etc.), data and service access rules, and other types of corporate data.

- What assets will Identity Server be protecting?

Identity Server secures access to all types of data and services. An administrator can regulate who can view and/or configure Identity Server data as well as control access to applications, portals and services.

- What users will leverage Identity Server?

Users might include employees, business partners, suppliers, and current or potential customers. Each user will have a profile which includes, at a minimum, their user ID and password. Employees will undoubtedly have larger and more confidential profiles than customers signing in for a look at external sales information.

- What data will be accessible?

Data might include public information, internal information, confidential information and restricted data. En toto, this might include sales information on an external website, confidential employee profiles, access rules which protect corporate resources, server configuration information and federated customer profiles. This data may or may not be stored on Directory Server.

- What is the authoritative source of the data?

There are often multiple schemas that define different types of data. These definitions need to be reconciled within your deployment. Be aware of data ownership issues, allowing the various LOB applications to maintain control over their data, where appropriate. It is imperative to balance the demands of the satellite groups in order to provide service that is representative of the overall enterprise as all services are critical to the larger organization.

Additionally, more technical considerations might include:

- Is the same information defined in multiple attributes?
- Do users have multiple cross-organizational profiles?
- Are the data stores located in front of the firewall?
- Is the data consistent across different data stores?
- How often is new data added, or existing data modified?

More information can be found in “Categorizing Data” on page 44.

## Evaluating Applications

Identity management services are generally provided as a centralized IT function with corporate and business unit applications forming the extended system. Upkeep of this system hierarchy involves a core IT group that manages and maintains the server infrastructure, and a satellite group of employees to maintain the LOB applications. As large organizations often have hundreds (or even thousands) of deployed internal applications, evaluating all of them would be time-intensive and cost-prohibitive. When conducting an application survey, focus on applications that:

- Are of particularly high value to the organization.
- Would naturally benefit from integration into a single sign-on infrastructure.
- Are indicative of standard programming and deployment platforms within your organization.
- Are generally receptive to the identity management infrastructure.
- Are currently in the early process of deployment or refactoring, and might logically have timelines that coincide with the Identity Server deployment.

You might develop a spreadsheet which can be used to organize the information from the most promising applications. An overall metric can be developed to compare the value of the application to the complexity of its integration. This might be considered an application's degree of fitness for deployment. An example of a highly fit application might be a web application that delegates authentication to an application server on which an Identity Server policy agent is installed for security. All user information would be stored in an LDAP directory. An example of an unfit application might be a *green screen* application (one with a text-based interface) driven off a mainframe. In this case, it would be advantageous to integrate other applications while waiting for a refactoring/architecture of the mainframe application. The following sections detail types of information that can be gathered when evaluating your organization's applications.

---

**NOTE** This step also helps in determining the resources that will be protected.

---

## Platform Information

General platform information, based on your existing technologies and hardware, can be used to assess the appropriateness of an application as a candidate for integration. Collected platform information might include:

- What operating system (including version) do the applications run on?
- Which web containers (including version) do they run in?
- With what programming model are the applications developed? (Java, ASP/.NET, C, etc.)
- Are there plans to upgrade the applications? If so, what is the timeline?

---

**NOTE** LOB applications may also be running third party applications (such as portals, content management databases, or human resource systems). These applications do not always deploy on platforms supported by Identity Server agents. Be prepared to assess the deployment criteria of these applications and schedule their inclusion based on agent availability. If custom work is required, Java-based applications are typically easier to integrate.

---

## Security Models

It is important to document the existing security models used within the LOB applications. Typically, applications that use external authentication and/or authorization will be good candidates for deployment as are applications which rely on external directory services. Security information might include:

- What authentication mechanisms are currently being used?
- Are there special authentication requirements (such as 2-factor authentication)?
- Is there a pluggable interface for external authentication mechanisms?
- What authorization mechanisms are currently being used?
- Can authorization be externalized? Does it make sense to do so?
- What user data repositories are being used? Can these be externalized?
- Who can access the application? Are there existing roles or groups in place? Under what special conditions are they granted access?

## Lifecycle Of A Session

An identity's session lifecycle is an important topic to assess when evaluating authentication applications. Make sure you have a clear picture of how a user session is created, managed and destroyed. Clearly document this process as it will be referred to when working on the application's integration. More information on this topic can be found in Appendix B, "The User Session Life Cycle."

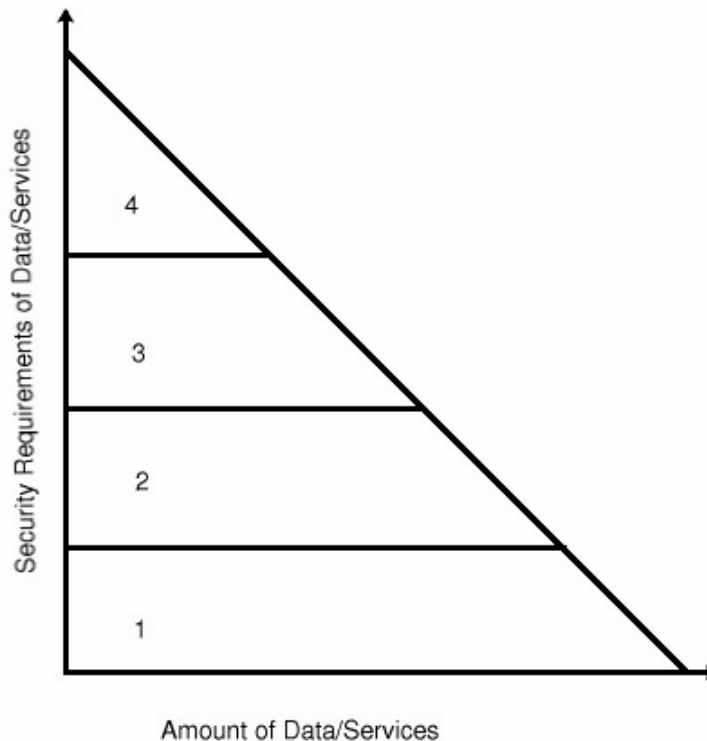
## Customization And Branding

Specific branding or look and feel requirements for the application needs to be considered. Often times, it is important to maintain an individual look and feel or to simply maintain consistency of user experience. Ensure that any customization and branding requirements are noted with your application assesment as time needs to be scheduled to account for this.

## Categorizing Data

Having analyzed your applications, and categorized them into fitness levels, you must now begin categorizing the data and services offered by those applications. This information will be used to build a security model. The categorization process itself is a procedure of data and service categorization, followed by cataloguing the existing authentication and authorization systems. The information collected in Evaluating Applications is used for the former portion of the process. A good methodology might be to organize the collected information into various tiers of security. These tiers would indicate the amount of risk associated with data loss, application compromise, abuse or other illicit types of access. Using well-defined categories will help to simplify the mapping of resources into a security model incorporating authentication and authorization requirements.

Figure 2-1 is a chart reflecting data within a typical organization. The data or service is separated into four levels of security. The X axis is the data or service and the Y axis is the security level associated with it. Tier 1 is illustrated with a minimal amount of security and might be data applicable to a public website. Tier 4, on the other hand, requires maximum security and might be financial or HR data. Your organization's categorization may have more or less tiers, but this chart shows how typical it is for large amounts of data to have low associated risk, and thus, low security requirements. As risk associated goes up, security requirements do also. (In reality, there is very little data with high security requirements, and a lot with no security requirements.)

**Figure 2-1** Security Requirements Of Data And Services

Keep in mind that you are aiming to build functional groupings of data and service types so that authentication and authorization functions can be mapped to them. Too many tiers will inject extra complexity into your process while too few might not offer enough flexibility. It is also important to note that there might be data with too much risk to place on the network at all. If relevant, make sure distinctions are made between internal and externally available data. Keep authentication and authorization requirements in mind as you build out these tiers, as well as conditional qualifiers such as such as access time of data and network location.

## Mapping To Authentication

With the data categorized according to security level, the next step is to inventory authentication and authorization mechanisms. Using a current list of available authentication mechanisms, associate those mechanisms with the security tiers defined. For example, the following might be appropriate for the data categorized in Figure 2-1.

- Tier 1 data might be appropriate for anonymous authentication with no access control.
- Tier 2 data might require password protection only.
- Tier 3 data might require hard token or certificate authentication.
- Tier 4 data might require multifactor authentication or might even be data that is not placed on the network.

You should ensure a clean mapping between authentication requirements and the data/services categorization. If there is none, look for common criteria between those items that do not match. Don't hesitate to make multiple charts if logical distinctions occur. For example, separate charts can be made for intranet and extranet applications. You might also categorize data based upon a functional security domain such as HR or finance. While not a universally applicable tool, categorizing your data in this manner will help you to understand your security requirements, and map them into logically manageable groups.

## Mapping To Authorization

Using the data available from your application assesment, examine each of the applications to determine a scalable authorization model. Typically, it is best to look for common groups/roles used across applications. These ideally will map to functional roles within the organization. You should also determine the source of those roles/groups (where does the membership data live and how is it modeled). Ideally, this will exist in Sun ONE Directory Server. If not, custom plugins may be required. If a robust grouping model is in place, begin associating each application with existing groups or roles. If not, begin planning a role or group machinism, finding common relationships between functional user types and access to specific applications. When completed you should have the following:

- A clear map of existing roles and groups.
- A clear understanding of where that data lives, and who is the authority over its quality and management.
- A clear understanding of new groups or roles that need to be created to facilitate your deployment, or to reduce cost/complexity.
- A mapping of existing and future grouping mechanisms to your categorized applications.
- Notes on additional conditions required by the applications to allow access to a certain group or role.

With this basic security model (categorization of data, with correlation to authentication and authorization mechanisms), you can now put together a timeline to drive your deployment.

## Building Timelines

From the information you have gathered, a preliminary timeline should be built. The following sections detail the steps for a generic schedule of deployment.

### Deployment Design

This phase of the timeline is where the concepts, business needs and user requirements gathered heretofore are put in their proper context. A total view of the deployment takes shape. Components are described, technological requirements are defined and a complete architecture is mapped out. Storyboarding login screens or creating data flow charts are two ways of initiating this design phase.

### Proof-of-Concept

A proof-of-concept enables the design to be tested in a business environment. Organizations often have a *test bed* database, a set of pre-configured test cases coupled with their expected results. The proof-of-concept can be applied to this test bed and, if all goes well, the documented results will be equivalent to the new results. A proof-of-concept aims to answer all question posed by the Deployment Design, proving that it meets all needs efficiently and with minimal risk. It is generally fast allowing for ample time to refine the design based on a limited set of data. There are certain to be several rounds of proof-of-concept followed by design refinement. The last round in the proof-of-concept should be integration of some internal applications. Integration of a corporation's shared services often adheres to a standard model of sign-on by early adopters, followed by general participation and, lastly, the stragglers. Demonstrated success with early adopters makes it easier to use those applications as references for general adoption.

## Early Adoption

Mission critical or revenue-building applications should not be chosen as your first application. A less risky strategy is to choose an important hub application which will not completely disrupt business operations if there are issues during roll-out. For example, a divisional portal serves as a natural staging ground for a single sign-on roll out, rather than an accounting system at the close of a fiscal period. Also, limit the number of applications roll-outs in the early phases so process flaws can be driven out, results demonstrated, and immediate success recognized. Minimizing organizational risk while maximizing visibility is the optimal roll-out strategy. This plan positions the deployment team with the appropriate product experience to take on critical applications.

## General Participation

Although the deployment project begins with a single application, the requirements of other internal customers should be assessed at the same time so that a general purpose system can be built. The central IT group should be able to accommodate the diverse criteria and schedules of the satellite groups in order to provide service that is representative of the larger organization. Schedules must have sufficiently large windows, allowing the satellite groups time to build changes and upgrades into their application's deployment and quality analysis (QA) cycle.

## Production Environment

Following the proof-of-concept, the refined design can be replicated into a production environment. The purpose of a production environment is to demonstrate the function of the design in a non-artificial environment, ensuring its proper behavior. It is compared to the behavior as observed in the proof-of-concept, and as defined in the deployment design. It is also tested for stability. An assessment is made and reports are generated. Early adoption applications go live in the production environment as they are ready. Incrementally phase new applications through the test bed phase and into production. Other applications are incrementally added to the production environment by working them through the proof-of-concept cycle as the early adoptors have been.

---

**NOTE** Sample timelines are not available, as they are variable based upon project complexity, but this process typically takes place in a span of 2-3 months.

---



# Identity Server Architecture

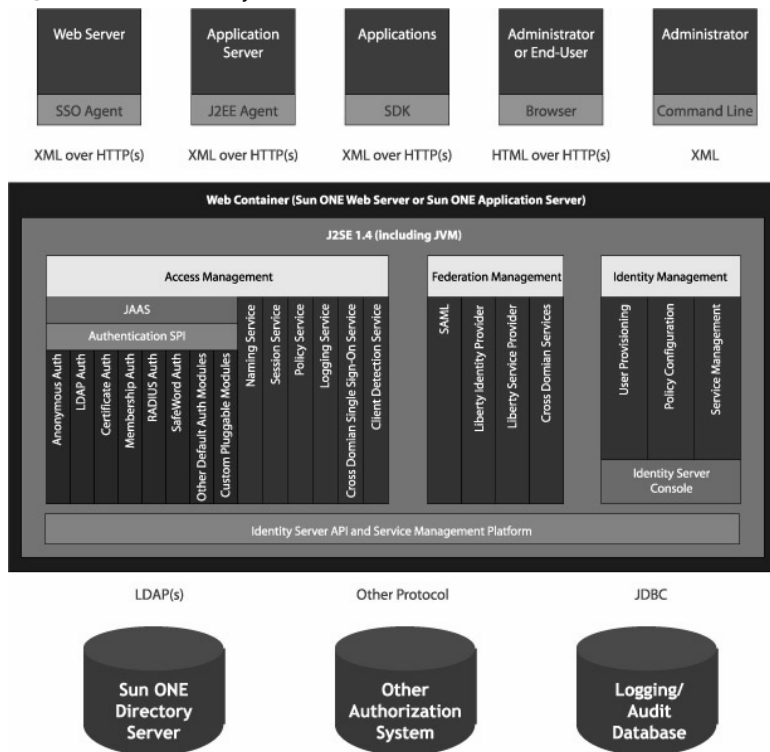
Sun™ ONE Identity Server provides interfaces for managing identity objects, policies and services. This chapter describes these interfaces as they relate to the core architecture of the product and specific architectural details of its services. It contains the following sections:

- Overview
- Integration Points
- Functional Processes
- Extending Identity Server

## Overview

Identity Server is built on the Java™ 2 Platform, Enterprise Edition (J2EE) platform and uses a *web container* as the infrastructure for operating the servlets, web service endpoints, and web applications which provide these services. Each of the provided services are implemented as web services which are built using a set of emerging standards and centrally provided on a network. In order to implement these services across the enterprise, Identity Server provides interfaces for managing identity-related objects, policies and services. The primary interfaces for this include policy agents that provide security for a wide variety of web and application servers, Java™ and C application programming interfaces (API), eXtensible Markup Language (XML) files, JavaServer Pages™, HyperText Markup Language (HTML) pages, a command line interface (CLI), and a graphical user interface (Identity Server console). With these interfaces it is possible to administer identity information in Sun ONE Directory Server, as well as extend SSO, authentication, policy-based authorization, and the logging functionality to applications across the enterprise. Figure 3-1 illustrates the functional architecture of Identity Server 6.1 and how the different components interact with each other.

**Figure 3-1** Identity Server 6.1 Functional Architecture



Specific details on the individual architecture of these components can be found in the *Sun ONE Identity Server Customization And API Guide*. The following section, Integration Points, details how some of these components interact with each other.

## Integration Points

Identity Server is a distributed system that relies on a variety of integration points to enable the delivery of its services. An *integration point* is a connector that allows an application to work smoothly with third-party systems or software. The primary integration points for Identity Server are policy agents and the software development kit (SDK).

## Policy Agents

Access control in the Identity Server is enforced using policy agents that protect content on designated web, application, or proxy servers from unauthorized intrusions. Policy agents are provided under separate cover from Identity Server. The most current Sun ONE Identity Server Policy Agents can be downloaded from the Sun Microsystems Download Center. They are divided into two categories: those that protect resources on web and proxy servers and those that protect J2EE applications in a variety of deployment containers.

### Web And Proxy Server Agents

Web and proxy server policy agents protect content deployed on said servers from unauthorized intrusions. The agents control access to a multitude of services and web resources based on policies configured by an administrator. When a user points a browser to a URL deployed on a protected web server, the agent intercepts the request and validates the user's session token, if any exists. If the token's authentication level is insufficient (or none exists), the appropriate Authentication Service is called for a login page, prompting the user for (further) authentication. The Authentication Service verifies that the user credentials are valid. (For example, the LDAP service verifies that the username and password are stored in Sun ONE Directory Server.) After the user's credentials are properly authenticated, the policy agent examines all the roles (which contain the policies) assigned to the user. Based on the aggregate of all policies assigned to the user, the individual is either allowed or denied access to the URL.

### J2EE Agents

Identity Server provides agents for protecting J2EE applications in a variety of deployment containers. These applications can include servers like IBM Lotus® Domino™ and applications from companies such as PeopleSoft. The J2EE agents are generally comprised of two components (although this is partially subject to the interfaces exposed and supported by the deployment container): an Agent Filter and an Agent Realm that handle Authentication and Authorization, respectively.

#### *Agent Filter*

The Agent Filter is a servlet filter that intercepts an inbound request to the server. It checks the request to see if it contains a session token. If one is available, the Agent Filter validates the token using the Identity Server Session Service. If no token is available, the user is re-directed to the Authentication Service as in a typical SSO exchange. Once the user is authenticated, they are directed back to the server where the Agent Filter once again intercepts the request, and then validates the

newly acquired token. After validation, the filter instantiates the principal and realm for the container, and the request is allowed to pass through the filter to the application. Through this mechanism, the Agent Filter ensures that only requests with a valid Identity Server token are allowed to access a protected application.

### *Agent Realm*

The Agent Realm provides the Authorization component. A *realm* is a means for a J2EE-compliant application server to provide information about users, groups, and access control to applications deployed on it. It is a scope over which security policy is defined and enforced. The server is configured to use a specific realm for validation of users and their roles, when attempts are made to access resources. By default, many application servers ship with a number of realm implementations, including the default File Based as well as LDAP, NT, Unix, and Relational Database Management System (RDBMS). The Agent Realm implements the server's Realm interface, and allows user and role information to be managed by the Identity Server deployment. The Agent Realm makes it possible to provide granular role-based Authorization of J2EE resources to users who have been authenticated by the Agent Filter.

---

**NOTE** For more detailed information, see the *Sun ONE Identity Server Web Policy Agents Guide* or the *Sun ONE Identity Server J2EE Policy Agents Guide*.

---

## Identity Server SDK

Another means to integrate an application within Identity Server is programmatically via the SDK. SDK access is provided primarily in Java, with C interfaces being provided for some functionality. It is also possible to interface directly with Identity Server services via XML over HTTP(s), although the document format is subject to change in subsequent releases. Future versions of Identity Server will formally support web service interfaces with Web Service Definition Language (WSDL) defined endpoints. The following SDKs are currently provided:

### SSO API

Identity Server provides interfaces for validating and managing session tokens, and for maintaining the user's authentication credentials. All applications wishing to participate in the SSO solution can use this API. The package name is `com.ipplanet.sso`.

## Authentication API And Authentication SPI

Identity Server provides a JAAS implementation which allows programmatic authentication against Identity Server. This provides remote access to the full capabilities of the Authentication Service. Additionally, an Authentication SPI following the JAAS PAM specification allows creation of new authentication types.

## Policy API

The Policy API can be used to evaluate and manage Identity Server policies as well as provide additional functionality for the Policy Service. A policy evaluator which directly loads and interprets policy from the Policy Store is provided, as well as a remote policy evaluator, which acts as a client of Identity Server using the Java API for XML-based Remote Procedure Calls (JAX-RPC). The API also provides the ability to determine which resources an identity has access to.

## Identity Management SDK

Combined with Identity Server templates that describe the data modeling used in the underlying Directory Server, the Identity Management SDK provides the framework to create and manage users, roles, groups, containers, organizations, organizational units, and sub-organizations. It provides an interface at a much higher abstraction level than the LDAP client APIs, and simplifies the management of directory-based identity information. The core packages here are `com.iplanet.am.sdk` which operates directly against Directory Server, and `com.sun.identity.um` which operates remotely against Identity Server via JAX-RPC which, in turn, acts against Directory Server on its behalf. (This is part of `am_services.jar`.) Finally, SPIs are provided to create plugins to the core SDK for tasks such as data validation, or password policy enforcement.

## Logging API and Logging SPI

The Logging Service records, among other things, access approvals, access denials and user activity. The Logging API can be used to enable logging for external Java applications, providing a common logging facility to any integrated application. The Logging SPI can be used to develop plug-ins for customized features.

## Service Management SDK

The service management interfaces can be used by developers to register service definitions that will be used to manage application configuration data. The API package name is `com.sun.identity.sm`.

## Client Detection API

Identity Server can detect the type of client browser that is attempting to access its resources and respond with the appropriately formatted pages. The API package used for this purpose is `com.ipplanet.services.cdm`. This package is part of `am_services.jar`.

## SAML SDK

Identity Server uses the SAML API to exchange acts of authentication, authorization decisions and attribute information. The API package names begin with `com.sun.identity.saml`. This package is part of `am_services.jar`.

## Federation Management API

Identity Server uses the Federation Management API to add functionality based on the Liberty Alliance Project specifications. The API package name is `com.sun.liberty`. This package is part of `am_services.jar`.

# Functional Processes

Identity Server contains a number of integrated functionalities. They include:

- Authentication and User Sessions
- Integrated Policy
- Integrated Client Detection
- CDSO, SAML and Federation

The integrated processes are discussed in the following sections. More specific details on many of the programmatic points can be found in the *Sun ONE Identity Server Customization And API Guide*.

## Authentication and User Sessions

The Authentication Service is the entry point of Identity Server. A user must pass an authentication process before accessing the Identity Server console and its corresponding management functions. A user attempting to access a service or application *protected* by Identity Server must also authenticate before access is

allowed. The Authentication Service invokes authentication modules to collect and validate the necessary credentials. Identity Server also provides APIs that allow applications to participate in a single sign-on functionality enabling a user to authenticate only once yet access multiple resources.

There are two ways to authenticate to the Identity Server: an administrator or end user attempts to access the Identity Server console itself for management purposes and is re-directed to the Authentication Service or, a user attempts to access an application protected by Identity Server and is re-directed to the Authentication Service. The former uses a HTML over HTTP(S) interface and the latter uses C-based or Java APIs.

---

**NOTE** A third option entails a user's attempt to access any protected resource on a remote server machine. This option includes the installation of Identity Server policy agents and will be discussed in "Integrated Policy" on page 57.

---

## HTML Over HTTP(S) Interface

In a typical web-based scenario, an administrator or end user desiring to open Identity Server for management purposes would first access the authentication user interface Java servlet by typing the Authentication Service uniform resource indicator (URI) into the location window of a web browser.

---

**NOTE** A *URI* is a generic term that encompasses the more well-known URL.

---

The URI is pre-configured to an authentication process that defines one or more authentication modules. The servlet parses the URI, contacts the Session Service which creates a session token (in an invalid state) and embeds the token in a cookie. The Authentication SPI (`com.sun.identity.authentication.spi`) invoke the specific authentication modules and pass the login pages and the token/cookie to the requesting browser. The user enters their credentials and the Authentication SPI returns the information to the Authentication Service which validates it against information in the corresponding data store. Upon successful completion of all required authentication processes, the user's policies (including URL access and deny lists), authentication level, and configuration parameters are passed to the Session Service which sets the token to a valid state. (Session information is not embedded in the token, but stored on the server. The token is only a pointer to the information.) Finally, the URL parameters are used to direct the user to the proper resource.

---

**NOTE** Additional information on the Authentication Service in general and URL parameters in particular can be found in the *Sun ONE Identity Server Customization And API Guide*.

---

## XML Over HTTP(S) Interface

Both Java and C applications can request user authentication from Identity Server with the Authentication API. The Java package, `com.sun.identity.authentication` provides interfaces to initiate the authentication process, and communicate the unvalidated authentication credentials from the application back to the Authentication Service.

---

**NOTE** `com.sun.identity.authentication` can be implemented locally or remotely. Remotely, developers incorporate the classes and methods from this package into their Java applications.

---

Java calls are converted into an XML message and passed to Identity Server over HTTP(S). Once received, the XML message is converted back into Java to be interpreted by the Authentication Service. C application developers would follow the same process but first must contact Identity Server.

---

**NOTE** `http://server_name.domain_name:port/service_deploy_uri/authservice` is a servlet which speaks using the XML over HTTP protocol. The relevant DTD, `remote-auth.dtd`, can be found in the `IdentityServer_base/SUNWam/dtd` directory.

---

After opening a connection, the C application is developed to send its authentication requests to the Identity Server as XML messages over HTTP(S). Again, when received, the XML messages are converted back into Java and is parsed by the Authentication Service which determines the authentication method, contacts the Session Service which creates a session token in an invalid state and embeds the token in a cookie. The Authentication SPI (`com.sun.identity.authentication.spi`) invokes the specific authentication modules and pass the login pages and the token/cookie to the requesting application. The user enters their credentials and the Authentication SPI returns the information to the Identity Server which validates it against the information in the corresponding authentication data store. Upon successful completion of all required



authentication processes, the user's policies (including URL access and deny lists), authentication level, and configuration parameters are passed to the Session Service which embeds the information in the session token and sets it to a valid state. Finally, the requestor is granted access to the application.

## Integrated Policy

Both authentication and single sign-on within an Identity Server deployment are interdependent; this means that there is not much a user can do after successful authentication if they do not also have a session token. The Policy Service has that same interdependence. Because a user's policies define their privileges regarding protected resources within the deployment, there is not much a user can do without Identity Server first regarding their defined policies. This function is taken care of through the use of *policy agents*, installed in web servers on the remote machines that store the resources protected by Identity Server.

---

**NOTE** The most current policy agents can be downloaded from the Web And Directory Servers page of the Sun Microsystems Download Center at [http://www.sun.com/software/download/inter\\_ecom.html](http://www.sun.com/software/download/inter_ecom.html).

---

When a user connects to a web server protected by a policy agent, the agent intercepts the request and checks for a session token embedded in a cookie. In the case where no token/cookie is found, the agent redirects the user to the login URL configured for this particular agent to begin the authentication process and create a session token. From this point, the process detailed in "Authentication and User Sessions" on page 54 is followed. After the user has received a valid session token, they are directed back to the originally requested resource where the agent once again intercepts the request and checks for the session token. Finding one, the agent must now validate the new token so a request is made to the Naming Service to decrypt the token and pass along the specific Session Service to contact for validation. The agent receives the response, extracts the URL for the Session Service, and contacts it for token validation. The Session Service sends a response back to the agent, validating the token. The agent then requests of the Session Service to receive notification if, for any reason, the session times out or becomes invalid. The Session Service responds in the affirmative. In order to get the user's defined policies, the agent can now contact the Policy Service which responds with a decision. The agent then uses this decision to allow or deny access to the requested resource. In addition, a log message is sent to the Logging Service.

## Integrated Client Detection

The first step in authenticating a user is to identify the client type making the request. The Authentication Service uses the URL information to retrieve the browser type's characteristics. Based on these characteristics, the correct authentication pages are sent back to the client browser; for example, HTML pages. Once the user is validated, the client type is added to the session token where it can be retrieved by other services. More information on client detection can be found in the Client Detection chapter of the *Sun ONE Identity Server Customization And API Guide*.

## CDSSO, SAML and Federation

Cross-Domain Single Sign-on (CDSSO), Security Assertion Markup Language (SAML) Service, and Federation Management are separate features of Identity Server that provide the same functionality in different ways. CDSSO is Identity Server's proprietary feature that allows single sign-on to occur across multiple DNS domains. The SAML Service can be used for the same purpose (cross domain single sign-on) in organizations that have implemented the SAML specifications in their existing deployments. Federation Management uses the Liberty Alliance Project's version 1.1 specifications for federated identity management. They enable simplified sign-on through a linked network of partners, and user control over the use and disclosure of personal identification.

---

**NOTE** The Liberty specifications are built integrating the SAML specifications although, in this case, the use of SAML is not a part of the Identity Server SAML service itself.

---

### CDSSO

CDSSO is a configurable part of the Identity Server authentication and authorization process. It includes the installation and configuration of policy agents which is discussed in "Authentication and User Sessions" on page 54. More indepth information can be found in the Single Sign-On chapter of the *Sun ONE Identity Server Customization And API Guide* as well as the *Sun ONE Identity Server Web Policy Agents Guide* or the *Sun ONE Identity Server J2EE Policy Agents Guide*.

## SAML

The SAML Service is based on an open-standard protocol that uses an XML framework to exchange security information between an authority and a trusted partner site. Information on its architecture within Identity Server can be found in the SAML Service chapter of the *Sun ONE Identity Server Customization And API Guide*.

## Federation

Federation Management enables the configuration and management of authentication domains, hosted providers and remote providers using the Identity Server console. This enables users to sign on once but gain access to any other federated resources. Information on the federation architecture can be found in the Federation Management chapter of the *Sun ONE Identity Server Customization And API Guide*.

# Extending Identity Server

Identity Server can be deployed with remote web or application servers, LDAP load-balancers such as Sun ONE Directory Proxy Server, and in multi-master replications. Before installation, consider how the following products might fit into a deployment. In some cases, they must be installed and configured before Identity Server.

## Web Containers

Some web containers are remote relative to the Sun ONE Web Server (or other web container) on which Identity Server itself is deployed. Remote web containers may already be deployed to serve content pages for your organization; additional ones might be needed. They become integrated within the Identity Server environment when a policy agent is installed to protect the content. For detailed web container installation and administration information, see the documentation that comes with the server, or access the Sun ONE Web Server documentation.

## Multiple Directory Server Instances

Multiple instances of Directory Server can be installed for upgrading, setting up failover directories, or deploying multi-masters. Directory Server must be installed, configured and deployed properly for the Identity Server deployment to be successful. Database replication agreements must be defined before installing Identity Server. For detailed Directory Server installation and deployment information, see the *Sun ONE Directory Server Installation And Tuning Guide* and the *Sun ONE Directory Server Deployment Guide*.

## LDAP Load Balancers

Identity Server can be configured to work with load balancers such as Sun ONE Directory Proxy Server. Load balancing across replicated servers and locating replicated servers closer to users are two ways to improve Identity Server performance and response time. Using load balancers adds a layer of high availability and directory failover protection beyond the basic level that comes with Identity Server. They manage request traffic as well as reject client queries when all back-end LDAP servers are unavailable. For detailed installation and administration information, see the Sun ONE Directory Proxy Server 5.2 documentation. For information on other load balancers, see the documentation that comes with the product.

# Pre-Deployment Considerations

Sun™ ONE Identity Server 6.1 enables large organizations with a heterogeneous hardware, software and application infrastructure to successfully deploy an identity management solution for their employees, contractors, customers and suppliers. This chapter provides a high-level technical overview of the issues related to this process. It contains the following sections:

- Deployment Options
- Hardware Requirements
- Software Requirements
- Understanding the Identity Server Schema

## Deployment Options

There are several key factors that an organization should consider when planning for an Identity Server deployment. These considerations generally deal with risk assessment and a growth strategy. For example,

- How many users is the environment expected to currently support, and what is the projected growth rate?

It is critical that user growth and system usage are monitored and this real data is compared with the projected data to ensure that the current capacity is capable of handling the projected growth.

- Are there future plans to add additional services to the environment which might impact the current design?

The architecture being developed now is optimized for the current service. Future plans should also be examined.

In addition, the architecture should provide a foundation for the objectives detailed in the following sections.

## Security

There are a number of options to consider when providing for a secure internal and external networking environment. They include:

- Server-based firewalls that provide an additional layer of security by locking down port-level access to the servers. As with standard firewalls, server-based firewalls lock down in-coming and out-going TCP/IP traffic.
- Minimization refers to removing all unnecessary software and services from the server in order to minimize the opportunity for exploitation of the vulnerabilities of a system.
- A Split-DNS infrastructure is one in which two zones are created in one domain. One zone is used by an organization's internal network clients and the other is used by external network clients. This approach is recommended to ensure a higher level of security. The DNS servers can also be load-balanced for performance.

## High Availability

IT deployments strive for no *single point of failure* (SPOF) as well as continuous availability to its users. Different products achieve availability in different ways; for example, clustering or multi-master replication. The desired *high availability* refers to a system or component that is continuously operational for a desirably long length of time. It is generally accomplished with multiple server machines that appear to the user as a single highly available system. In a deployment that meets the minimal requirements (all applications on one machine), the SPOFs might include:

- Web container
- Directory Server
- Java™ Virtual Machine (JVM)
- Directory Server hard disk
- Identity Server hard disk
- Policy agents

As most of these issues can be waylaid, much planning for high availability centers around backup and failover processing as well as data storage and access. For storage, a redundant array of independent disks (RAID) is one approach. For any system to be highly available, the parts of the system should be well-designed and thoroughly tested before they are used. For example, a new application program that has not been thoroughly tested is likely to become a frequent point-of-breakdown in a production system.

---

**NOTE** In a high availability scenario, the encryption key needs to be filled out identically for all installs of Identity Server.

---

## Clustering

Clustering is the use of multiple computers to form a single, highly available system. Clustering isn't used for Sun ONE Identity Server however, it is crucial within the underlying Sun ONE Directory Server data store. For example, a clustered MMR server pair can increase the availability of each master instance by ensuring availability.

## Scalability

*Horizontal scaling* is achieved by connecting multiple server machines so they work as one unit. A load balanced service is considered horizontally scaled as it increases the speed and availability of the service. *Vertical scaling*, on the other hand, is increasing the capacity of existing hardware by adding resources within one server machine. The types of resources that may be scaled include CPUs, memory, and storage. Horizontal scaling and vertical scaling are not mutually exclusive; they can work hand-in-hand. Typically, servers in an environment are not installed at full capacity, so vertical scaling is used to improve performance. When a machine approaches full capacity, horizontal scaling can be used to distribute the load among a larger number of servers.

# Hardware Requirements

The hardware on which Identity Server will be installed must meet certain requirements. At a minimum, Identity Server needs to be installed in tandem with Sun ONE Directory Server 5.2 (to be used as a data store) and a web container in which it will be deployed. It is recommended that Directory Server and Identity Server be installed on different machines.

The detailed requirements are high-level, based on a default configuration of the components: one instance of Identity Server (deployed by Sun ONE Web Server), and one instance of Directory Server. Please review the Directory Server *Installation And Tuning Guide* and the *Deployment Guide*, as well as the documentation from the web container of choice before installing Identity Server.

---

**NOTE** The recommended procedure is to consult Sun ONE Professional Services or a Sun ONE-certified system integrator before designing and deploying a Sun ONE Identity Server.

---

It is recommended that Identity Server be run on a 100 MB or greater Ethernet network. The minimum configuration for an Identity Server deployment is a machine with both Identity Server and Sun ONE Web Server installed on it. It should be a machine with one or more CPUs, with greatly diminishing returns on processor investment after four. Two to four CPUs per server is highly recommended. A minimum of 256 MB of RAM is necessary for basic testing of the software. For an actual deployment scenario, 1 GB of RAM is recommended for threads, the SDK, the HTTP server, and other internals; 2 GBs for basic operation, and object allocation space, and 100 MBs per 10,000 concurrent sessions. Each Identity Server is recommended to cap out at 100,000 concurrent sessions, after which horizontal load balancing should be applied (assuming the 4GB memory limitation of 32bit applications).

---

**NOTE** Typically, directory resource requirements are high although they may differ based on customer specific data and usage.

---

## Software Requirements

The systems on which Identity Server will be installed must also meet minimum software and operating system requirements.

### Operating System Requirements

Identity Server is supported on the following platforms:

- Sun Solaris™ 8 Operating System (SPARC® 32/64-bit processors)
- Sun Solaris 9 Operating System (SPARC 32/64-bit processors)
- Sun Solaris 9 Operating System (x86 processor)



## Patch Clusters for Solaris

When running Directory Server on a Solaris 8 operating system, the recommended patch cluster must be installed. Patch clusters are identified by two numbers, for example, 108827-15. The first number (108827) identifies the patch itself. The second number identifies the version of the patch (15). The latest patches should be installed in order to benefit from the latest fixes. Overall patch information and recommended patches can be downloaded from the SunSolve Patch Support Portal. (Patch clusters are required for Sun ONE Directory Server, Sun ONE Web Server and Sun ONE Application Server.)

---

**NOTE** The command `showrev -p` can be used to list the patches currently installed on a Solaris machine. .

---

Following are a list of patches that should be installed before installation of Identity Server. See the *Sun Java Enterprise System 2003Q4 Installation Guide* for more complete information on the required patches.

**Table 4-1** Recommended Patch List For Identity Server Installation

112396-02	111293-04	109888-20
108869-18	109326-10	108725-12
109147-21	112218-01	108981-10
108949-07	108434-10	109885-09
108435-10	110458-02	112237-07
111325-02	109898-05	109234-09
110075-01	110901-01	109134-27
108901-06	112325-01	108968-08
110662-11	110943-01	108975-06
110898-08	109324-05	108875-13
111111-03	110916-04	110460-26
109091-05	110387-03	108727-22
110283-06	109277-03	109318-31
110951-03	111234-01	108985-03
110903-05	112138-01	109238-02
109320-06	109470-02	109793-14

**Table 4-1** Recommended Patch List For Identity Server Installation (*Continued*)

109783-02	109951-01	111299-04
110453-04	110945-07	110934-11
111071-01	111232-01	108997-03
110700-01	110939-01	109007-09
111548-01	111570-02	108974-25
108652-65	110322-02	110386-02
110286-10	111504-01	108987-12
111606-02	109882-06	109805-15
111069-01	110615-08	108827-40
110668-03	110670-01	108993-13
111826-01	111874-06	110723-05
111659-07	111596-03	111327-05
109667-04	110957-02	108977-01
111626-03	111085-02	110380-04
108919-17	110838-06	108528-19
112254-01	112459-01	108989-02
111879-01	112611-01	111881-03
112425-01	112668-01	109657-09
111958-02	112796-01	111883-14
112846-01	112279-02	114152-01
108806-15	110842-10	111310-01
111321-03	109328-03	111098-01
113792-01	109862-03	113650-01
110896-02	108899-04	109223-02

## Java™ Requirement

Identity Server requires Java version 1.3.1. It is recommended though that Java version 1.4.1 is used.

## Resource Web Server Requirements

Supported web containers for Identity Server are Sun ONE Web Server 6.1 and Sun ONE Application Server 7.0. When policy agents are installed on these web containers, they use approximately 10MB of disk space. This must be taken into account when configuring the web container that serves content. For detailed information, see the *Sun ONE Identity Server Web Policy Agents Guide* or the *Sun ONE Identity Server J2EE Policy Agents Guide*.

## Web Browser Requirements

Administrators and end users use web browsers to perform user management tasks. Identity Server supports the following web browsers:

- Netscape Communicator 4.79.
- Microsoft Internet Explorer 5.5 SP 2.
- Microsoft Internet Explorer 6.0.

# Understanding the Identity Server Schema

In general, a schema is a set of rules imposed on data that serves to define how it is stored. Directory Server contains a Lightweight Directory Access Protocol (LDAP) schema that defines how its data is stored. Object classes define attributes in a LDAP schema. In Directory Server, each data entry must have one or more object class(es) to specify the type of object the entry describes and define the set of attributes it contains. Each entry then is basically a set of attributes and their corresponding values and the list of object classes to which these attributes correspond.

Identity Server uses Directory Server as the data repository for all its identity profiles, entitlement definitions, and deployment configuration information. Towards this end, Identity Server has its own schema which extends the Directory Server schema. When Identity Server is installed, the Identity Server schema, detailed in `ds_remote_schema.ldif` and `sunone_schema2.ldif`, is integrated with the Directory Server schema. `ds_remote_schema.ldif` details LDAP object classes and attributes that are specifically used by Identity Server; these object classes and attributes are generally holdovers from previous versions of Identity Server. `sunone_schema2.ldif` loads the Identity Server-specific LDAP schema

object classes and attributes defined by Sun Microsystems' new internal schema document. For reference, `ds_remote_schema.ldif` and `sunone_schema2.ldif` are reproduced in Code Example 4-1 on page 68 and Code Example 4-2 on page 71, respectively.

**Code Example 4-1**    `ds_remote_schema.ldif`

```

add: objectClasses

objectClasses: ( 2.16.840.1.113730.3.2.175 NAME 'iplanet-am-session-service'
DESC 'Session Service OC' SUP top AUXILIARY MAY (
iplanet-am-session-max-session-time $ iplanet-am-session-max-idle-time $
iplanet-am-session-max-caching-time $ iplanet-am-session-get-valid-sessions
$ iplanet-am-session-destroy-sessions $
iplanet-am-session-add-session-listener-on-all-sessions $
iplanet-am-session-service-status ) X-ORIGIN 'Sun Java System Identity
Management' )
objectClasses: ( 2.16.840.1.113730.3.2.176 NAME 'iplanet-am-user-service'
DESC 'User Service OC' SUP top AUXILIARY MAY ( iplanet-am-user-auth-modules
$ iplanet-am-user-login-status $ iplanet-am-user-admin-start-dn $
iplanet-am-user-auth-config $ iplanet-am-user-alias-list $
iplanet-am-user-success-url $ iplanet-am-user-failure-url $
iplanet-am-user-federation-info-key $ iplanet-am-user-federation-info $
iplanet-am-user-password-reset-options $
iplanet-am-user-password-reset-question-answer $
iplanet-am-user-password-reset-force-reset $ sunIdentityServerDiscoEntries
) X-ORIGIN 'Sun Java System Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.177 NAME
'iplanet-am-web-agent-service' DESC 'Web Agent Service OC' SUP top AUXILIARY
MAY ( iplanet-am-web-agent-access-allow-list $
iplanet-am-web-agent-access-deny-list $
iplanet-am-web-agent-access-not-enforced-list $
iplanet-am-web-agent-service-status ) X-ORIGIN 'Sun Java System Identity
Management' )
objectClasses: ( 2.16.840.1.113730.3.2.179 NAME 'iplanet-am-managed-role'
DESC 'Managed Role OC' SUP top AUXILIARY MAY ( iplanet-am-role-type $
iplanet-am-role-description $ iplanet-am-role-aci-description $
iplanet-am-role-aci-list $ iplanet-am-role-service-options $
iplanet-am-role-any-options $ iplanet-am-role-managed-container-dn $
iplanet-am-role-display-options) X-ORIGIN 'Sun Java System Identity
Management' )
objectClasses: ( 2.16.840.1.113730.3.2.180 NAME 'iplanet-am-managed-group'
DESC 'Managed Group OC' SUP top AUXILIARY MAY (
iplanet-am-group-subscribable $ inetgroupstatus ) X-ORIGIN 'Sun Java System
Identity Management' )

```

**Code Example 4-1** ds\_remote\_schema.ldif (Continued)

```

objectClasses: ( 2.16.840.1.113730.3.2.181 NAME
'iplanet-am-managed-filtered-group' DESC 'Managed Filter Group OC' SUP
iplanet-am-managed-group AUXILIARY X-ORIGIN 'Sun Java System Identity
Management' )
objectClasses: ( 2.16.840.1.113730.3.2.182 NAME
'iplanet-am-managed-assignable-group' DESC 'Managed Assignable Group OC' SUP
iplanet-am-managed-group AUXILIARY X-ORIGIN 'Sun Java System Identity
Management' )
objectClasses: ( 2.16.840.1.113730.3.2.183 NAME
'iplanet-am-managed-static-group' DESC 'Managed Static Group OC' SUP
iplanet-am-managed-group AUXILIARY X-ORIGIN 'Sun Java System Identity
Management' )
objectClasses: ( 2.16.840.1.113730.3.2.184 NAME 'iplanet-am-managed-person'
DESC 'Managed Person OC' SUP top AUXILIARY MAY ( iplanet-am-modifiable-by $
iplanet-am-static-group-dn $ iplanet-am-user-account-life ) X-ORIGIN 'Sun
Java System Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.186 NAME
'iplanet-am-managed-org-unit' DESC 'Managed OrganizationalUnit OC' SUP top
AUXILIARY MAY ( sunPreferredDomain $ associatedDomain $
sunPreferredOrganization $ sunAdditionalTemplates $ sunOverrideTemplates $
iplanet-am-service-status ) X-ORIGIN 'Sun Java System Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.187 NAME
'iplanet-am-managed-people-container' DESC 'Managed People Container OC' SUP
top AUXILIARY X-ORIGIN 'Sun Java System Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.189 NAME
'iplanet-am-managed-group-container' DESC 'Managed Group Container OC' SUP
top AUXILIARY X-ORIGIN 'Sun Java System Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.166 NAME 'iplanet-am-managed-policy'
DESC 'Managed Name Policy OC' SUP top AUXILIARY MAY
iplanet-am-named-policy-dn X-ORIGIN 'Sun Java System Identity Management' )
objectClasses: ( 2.16.840.1.113730.3.2.167 NAME
'iplanet-am-domain-url-access-service' DESC 'Domain URL Access Service OC'
SUP top AUXILIARY MAY iplanet-am-domain-url-access-allow X-ORIGIN 'Sun Java
System Identity Management' )
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.22 NAME 'iplanet-am-saml-service'
DESC 'SAML Service OC' SUP top AUXILIARY MAY ( iplanet-am-saml-user $
iplanet-am-saml-password ) X-ORIGIN 'Sun Java System Identity Management' )
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.23 NAME
'iplanet-am-auth-configuration-service' DESC 'Authentication Configuration
Service OC' SUP top AUXILIARY MAY ( iplanet-am-auth-configuration $
iplanet-am-auth-login-success-url $ iplanet-am-auth-login-failure-url $
iplanet-am-auth-post-login-process-class ) X-ORIGIN 'Sun Java System
Identity Management' )
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.25 NAME 'sunservice' DESC 'object
containing service information' SUP top MUST ou MAY ( labeleduri $
sunserviceschema $ sunkeyvalue $ sunxmlkeyvalue $ sunpluginschema $
description ) X-ORIGIN 'Sun Java System Identity Management' )

```

**Code Example 4-1** ds\_remote\_schema.ldif (Continued)

```

objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.26 NAME 'sunorgservice' DESC
'Service information specific to organizations' SUP top MUST ou MAY (
sunkeyvalue $ sunxmlkeyvalue $ description ) X-ORIGIN 'Sun Java System
Identity Management' )
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.27 NAME 'sunservicecomponent' DESC
'Sub-components of the service' SUP top MUST ou MAY ( sunserviceid $
sunsmspriority $ sunkeyvalue $ sunxmlkeyvalue $ description ) X-ORIGIN 'Sun
Java System Identity Management' )
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.28 NAME 'sunserviceplugin' DESC
'Object that stores information specific to plugins' SUP top MUST ou MAY (
sunpluginid $ sunkeyvalue $ sunxmlkeyvalue $ sunsmspriority ) X-ORIGIN 'Sun
Java System Identity Management' )
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.74 NAME
'iplanet-am-managed-filtered-role' DESC 'Managed Filtered Role OC' SUP
iplanet-am-managed-role AUXILIARY X-ORIGIN 'Sun Java System Identity
Management' )
objectClasses: ( sunISManagedOrganization-oid NAME
'sunISManagedOrganization' DESC 'Sun Java System objectclass to identify
organizations' SUP top AUXILIARY MAY ( sunOrganizationAlias ) X-ORIGIN 'Sun
Java System Identity Management' )
objectClasses: ( sunIdentityServerDiscoveryService-OID NAME
'sunIdentityServerDiscoveryService' DESC 'Discovery Service OC' SUP top
AUXILIARY MAY ( sunIdentityServerDynamicDiscoEntries ) X-ORIGIN 'Sun Java
System Identity Management' )

```

**Code Example 4-1** ds\_remote\_schema.ldif (Continued)

```

objectClasses: ( sunIdentityServerLibertyPPService-oid NAME
'sunIdentityServerLibertyPPService' DESC 'sunIdentityServerLibertyPPService
OC' SUP top AUXILIARY MAY ( sunIdentityServerPPCommonNameCN $
sunIdentityServerPPCommonNameALTCN $ sunIdentityServerPPCommonNameFN $
sunIdentityServerPPCommonNameSN $ sunIdentityServerPPCommonNamePT $
sunIdentityServerPPCommonNameMN $ sunIdentityServerPPInformalName $
sunIdentityServerPPLegalIdentityLegalName $
sunIdentityServerPPLegalIdentityDOB $
sunIdentityServerPPLegalIdentityMaritalStatus $
sunIdentityServerPPLegalIdentityGender $
sunIdentityServerPPLegalIdentityAltIDType $
sunIdentityServerPPLegalIdentityAltIDValue $
sunIdentityServerPPLegalIdentityVATIDType $
sunIdentityServerPPLegalIdentityVATIDValue
$sunIdentityServerPPEmploymentIdentityJobTitle
$sunIdentityServerPPEmploymentIdentityOrg $
sunIdentityServerPPEmploymentIdentityAltO ) X-ORIGIN 'Sun Java System
Identity Management' )
objectClasses: ( sunIdentityServerDevice-OID NAME 'sunIdentityServerDevice'
DESC 'Device OC' SUP top AUXILIARY MAY ( cn $ uid $
sunIdentityServerDeviceVersion $ sunIdentityServerDeviceType $ userpassword
$ sunIdentityServerDeviceKeyValue $ sunxmlkeyvalue $ description $
sunIdentityServerDeviceStatus ) X-ORIGIN 'Sun Java System Identity
Management' )

```

**Code Example 4-2** sunone\_schema2.ldif

```

add: objectClasses

objectClasses: ( 2.16.840.1.113730.3.2.185 NAME 'sunManagedOrganization'
DESC 'Auxiliary class which must be present in an organization entry' SUP
top AUXILIARY MAY ( inetDomainStatus $ sunPreferredDomain $ associatedDomain
$ sunPreferredOrganization $ sunAdditionalTemplates $ sunOverrideTemplates $
sunRegisteredServiceName $ organizationName ) X-ORIGIN 'Sun Java System
Identity Management' )
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.75 NAME 'sunManagedSubOrganization'
DESC 'Auxiliary class which must be present in an sub organization entry'
SUP top AUXILIARY MAY ( inetDomainStatus $ parentOrganization ) X-ORIGIN
'Sun Java System Identity Management' )

```

**Code Example 4-2** sunone\_schema2.ldif

```

objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.29 NAME 'sunNameSpace' DESC
'Auxiliary class which must be present at the root of a subtree representing
a namespace' AUXILIARY MAY sunNameSpaceUniqueAttrs X-ORIGIN 'Sun Java System
Identity Management' )
objectClasses: ( 1.3.6.1.4.1.42.2.27.9.2.27 NAME 'sunservicecomponent' DESC
'Sub-components of the service' SUP top MUST ou MAY ( sunserviceid $
sunsmspriority $ sunkeyvalue $ sunxmlkeyvalue $ description ) X-ORIGIN 'Sun
Java System Identity Management' )

```

## Marker Object Classes

Identity entries created using the Identity Server console and stored in Directory Server are appended with *marker object classes*. Marker object classes define the designated entries as those which Identity Server will manage. The object classes will not interfere with other aspects of the directory tree, such as servers or hardware. As well, existing identity entries can not be managed using Identity Server until they are modified to include these object classes. More detailed information on the marker object classes can be found in Chapter 5, “Identity Management” in the Sun ONE Identity Server *Customization And API Guide*. Information on how to migrate existing Directory Server data for use with Identity Server can be found in the Sun ONE Identity Server *Migration Guide* .

## Administrative Roles

Delegated administration of the LDAP entries (mapped to each identity-related object in Identity Server) are implemented through the use of pre-defined roles and access control instructions (ACIs). Default administrative roles and their defined ACIs are created during Identity Server installation, and can be viewed and managed using the Identity Server console. When an Identity Server identity-related object is created, the appropriate administrative roles (and, thus, corresponding ACIs) are also created and assigned to the LDAP entry for that object. The role can then be assigned to an individual user who maintains control of that object’s node. For example, when Identity Server creates a new organization, several roles are automatically created for it and stored in Directory Server:

- Organization Administrator has read and write access to all entries in the configured organization.



- Organization Help Desk Administrator has read access to all entries in the configured organization and write access to the `userPassword` attribute in those entries.
- Organization Policy Administrator has read and write access to all policies in the organization.

The assignment of any of these roles to a user gives that user all the permissions accorded that role. Table 4-2 summarizes the Identity Server administrator roles and the scope of write permissions that correspond to eachone.

**Table 4-2** Default And Dynamic Roles And Their Permissions

<b>Role</b>	<b>Administrative Suffix</b>	<b>Permissions</b>
Top-level Organization Admin (amadmin)	Root level	Read and write access to all entries (roles, policy, groups, etc.) under top-level organization.
Top-level Organization Help Desk Admin	Root level	Read and write access to all passwords under top-level organization.
Top-level Organization Policy Admin	Root level	Read and write access to policies created under top-level organization only. Used by sub-organizations to delegate referral policy creation.
Organization Admin	Organization only	Read and write access to all entries (roles, policy, groups, etc.) under the created sub-organization only.
Organization Help Desk Admin	Organization only	Read and write access to all passwords under the created sub-organization only.
Organization Policy Admin	Organization only	Read and write access to all policies under the created sub-organization only.
Container Admin	Container only	Read and write access to all entries (roles, policy, groups, etc.) under the created container only.
Container Help Desk Admin	Container only	Read and write access to all passwords under the created container only.
Group Admin	Group only	Read and write access to all entries (roles, policy, groups, etc.) under the created group only.
People Container Admin	People Container only	Read and write access to all entries (roles, policy, groups, etc.) under the created people container only.

**Table 4-2** Default And Dynamic Roles And Their Permissions (*Continued*)

Role	Administrative Suffix	Permissions
User (self-administrator)	User only	Read and write access to all attributes in the user entry only.

Using roles instead of group-based ACIs is more efficient and requires less maintenance. Filtered roles are simpler for LDAP clients, since they can just ask for the `nsRole` attribute of a user. Roles do suffer though from scope limitations, where a role must be a peer of a parent of a member of that role. More detailed information on the default ACIs can be found in Chapter 14, “Administration Attributes” in the Sun ONE Identity Server *Administration Guide*.

## Schema Limitations

The following sections detail some limitations of the Identity Server schema.

### People Containers

In Identity Server, the *people container* is a parent entry for users only. An organizational unit is typically used as a people container, but any entry may be this type as long as it has the `iplanet-am-managed-people-container` object class and an Identity Server manageable parent type object, *organization* or *container*. When an LDAP entry is marked with `iplanet-am-managed-people-container`, Identity Server assumes it will only contain sub-people containers or users. Therefore, the people container can only contain sub-people containers or users.

### Only One Identity Server Organization Allowed

Only one LDAP object type can be marked as an Identity Server organization. In other words, the `iplanet-am-managed-org` marker object class can only be added to one LDAP type entry. (This object class is added so Identity Server will manage this entry as if it were an organization.)

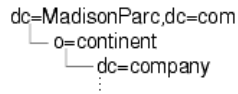
---

**CAUTION** Be aware that the root suffix organization created during installation does not count towards this one organization restriction.

---

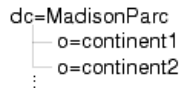
Thus, in Figure 4-1, both the `dc` and `o` entries can not be managed as organizations using the Identity Server console.

**Figure 4-1** Unmanageable Directory Tree



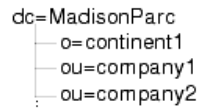
In Figure 4-2, this tree can be managed by Identity Server.

**Figure 4-2** One Organization Rule Exception



If you were to add `dc=company1` below `o=continent1`, this tree would be manageable only if `dc` is marked as a container, not an organization. The `iplanet-am-managed-container` object class is usually added to all organizational units, however, it can be added to any entry.

**Figure 4-3** Two Organizational Units Allowed

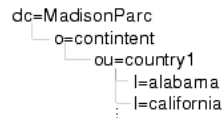


In this example, since you cannot mark both `o=` and `ou=` entries as organizations you could mark the `o=` entry as an organization and the `ou=` entries as containers. Management of both organizations and containers using Identity Server have the same options. People containers, sub-organizations, groups, roles, and users can be created under both of them.

## Unsupported Directory Trees

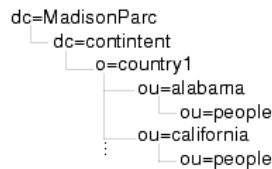
Most existing directory trees can be reconfigured to work with Identity Server although, in some cases, reconfiguration is not recommended. In general, if an existing tree uses more than one type of LDAP entry (examples: `dc`, `o`, and `ou`) to define organizations, the user data will be recognized by Identity Server only under certain conditions. In the following example, you would need three types of organization makers: `o`, `ou`, and `l`. Assuming that `l=california` and `l=alabama` are not people containers, this DIT would not work with Identity Server.

**Figure 4-4** Scenario 1: Three LDAP Organization Attributes not Allowed



In the next example, you would need three types of Identity Server markers (`dc`, `o`, `ou`) plus the people container (`ou=people`). Under these assumptions, the DIT would not work with Identity Server:

**Figure 4-5** Scenario 2: Three LDAP Organization Attributes not Allowed



# Deployment Scenarios

This chapter details simple deployment scenarios for Identity Server. It contains the following sections:

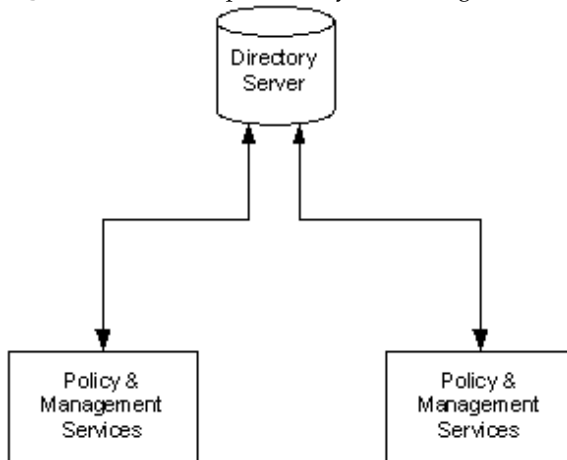
- Multiple Servers Scenario
- Web Deployment
- Java Application Deployment
- Multiple JVM Environment
- Replication Considerations
- Implementing Federation Management

## Multiple Servers Scenario

The recommended deployment scenario contains two physical servers with both Identity Server and Directory Server installed on each. The instances of Directory Server will be set up in a multiple master configuration. The proper procedure for this deployment would begin with the installation of the first instance of Directory Server on the first machine. The second instance of Directory Server would then be installed on the second machine and the multiple master relationship configured. Identity Server can then be installed on the first machine pointing to either instance of Directory Server. The installation method would be to choose an existing directory with or without an existing DIT, depending on the current setup. The second instance of Identity Server would then be installed on the second machine pointing to an existing Directory Server with an existing DIT. Identity Server does not write any information to Directory Server that it recognizes as already existing therefore, there is no danger of overwriting existing data. It will add two attributes which allow the second instance to function.

More than one instance of Identity Server can be installed against Directory Server for enhanced performance, for directory replication support, or for agent failover purposes. In order to have these multiple instances of Identity Server on different hardware against the same Directory Server, the instructions detailed in “To Install Multiple Identity Servers With `ammultiserverinstall`” on page 78 can be followed. A second method would be to install the second Identity Server, and point it against the already configured Directory Server. Figure 5-1 illustrates two Identity Server instances installed against a single Directory Server.

**Figure 5-1** Multiple Identity Servers Against One Directory Server



## To Install Multiple Identity Servers With `ammultiserverinstall`

To install multiple installations of Identity Server against one master Directory Server, the `ammultiserverinstall` script must be run. The administrator must have root permissions to create and install multiple Identity Server instances.

1. Go to the following directory:

```
cd Identity_Server_root/SUNWam/bin
```

2. At the command line, type the following command:

```
./ammultiserverinstall instance_name port_number
```

where *instance\_name* is the new Identity Server instance to be created and *port\_number* is the port number of the new Identity Server instance.

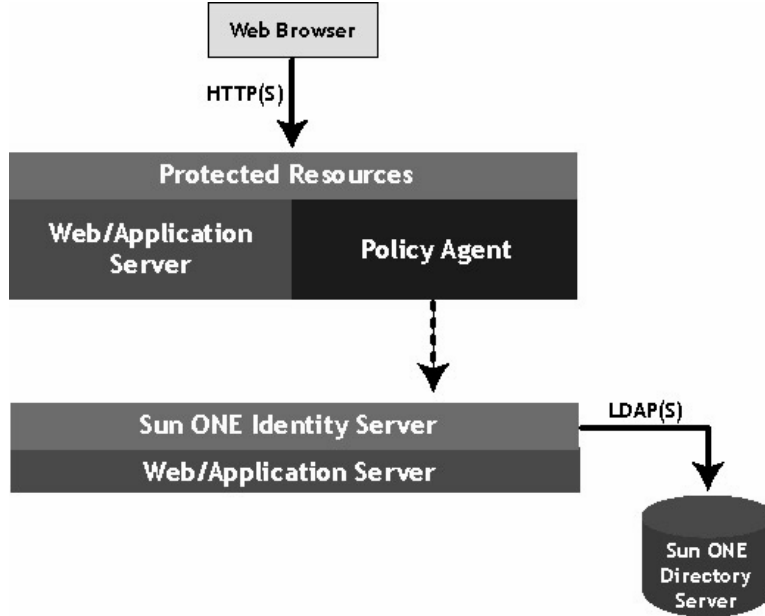
When a new instance is created, the following are created:

- A new `amserver` script file at  
`/IdentityServer_base/SUNWam/bin/amserver.instance_name`
- A new `AMConfig.properties` file at  
`/IdentityServer_base/SUNWam/lib/AMConfig-instance_name.properties`
- A new web server instance directory at  
`/IdentityServer_base/SUNWam/servers/https-instance_name`

More information on the `ammultiserverinstall` script can be found in the *Sun ONE Identity Server Administration Guide*.

## Web Deployment

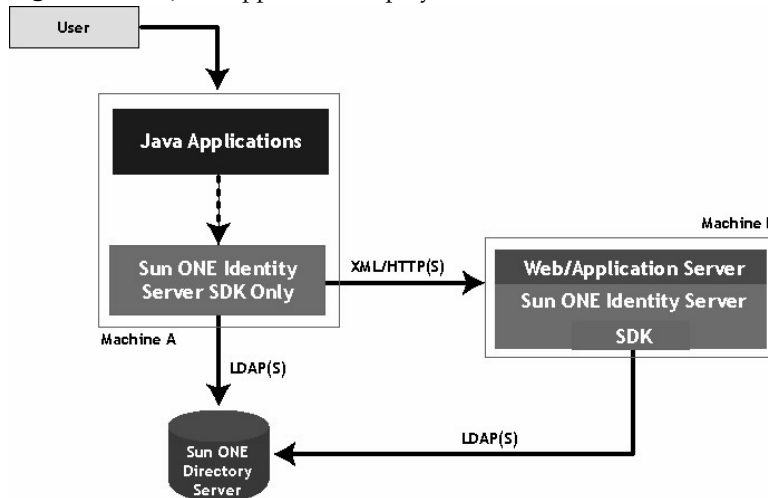
The most common use of an Identity Server deployment is when a web browser accesses an application or resource deployed on a web server. The application/resource is protected by Identity Server and communicates with it using a policy agent installed on the web server. Additionally, the web server might have the Identity Server SDK deployed. This architecture does not impose restrictions with regards to the number of web servers within a machine, or the instances of Identity Server across multiple machines. For example, a machine might have multiple web servers, each deploying Identity Server. Similarly, there might be multiple web servers running on different machines, each deploying Identity Server. Figure 5-2 on page 80 illustrates this simple deployment scenario.

**Figure 5-2** Simple Web Deployment Scenario

## Java Application Deployment

Another common scenario for Identity Server is one in which Java™ applications access an Identity Server SDK installed directly on the machine where they are deployed. In this scenario, there must be an additional machine with an instance of Sun ONE Web Server or Sun ONE Application Server running, at least, one instance of Identity Server; this machine maintains the state information to provide single sign-on. Figure 5-3 on page 81 illustrates this Java application deployment scenario.



**Figure 5-3** Java Application Deployment

## Multiple JVM Environment

Identity Server services are supported in multiple Java Virtual Machine (JVM) environments. In other words, an instance of Sun ONE Application Server can be configured to have multiple JVMs with Identity Server services running in all of them. The Identity Server architecture imposes no restrictions on the deployment with regards to the number of Application Server instances within a machine, the number of Identity Server services across multiple machines, or the number of JVMs that a single Application Server can have. See the Sun ONE Application Server documentation for more information on the multiple JVM environment.

## Replication Considerations

Load balancing across replicated Directory Servers and locating replicated servers closer to users are two ways to improve Identity Server performance and response time. Directory Servers can be set up in single-supplier or multi-supplier configurations. Load-balancing applications such as Sun ONE Directory Proxy Server can also be used. Directory Proxy Server dynamically performs proportional load balancing of LDAP operations across a set of configured Directory Servers. If one or more Directory Server(s) become unavailable, the load is proportionally redistributed among the remaining servers. When the server comes back on line, the load is proportionally and dynamically reallocated.

---

**NOTE** Directory replication agreements must be configured before installing Identity Server. This ensures that the supplier and consumer databases are synchronized correctly, allowing time to verify that referrals and updates are synchronized properly.

---

When Identity Server is installed for replication purposes, each instance of Directory Server and each instance of Identity Server, must be configured with the same values for the following:

- Directory Manager
- Directory Manager Password
- Directory Server Administrator ID
- Server Administrator Password
- Base suffix
- Default organization

## Configuring For Replication

Identity Server can be configured to work with single-supplier or multi-supplier replication. Figure 5-4 illustrates a single-supplier configuration where the consumer is a read-only database. Requests for write operations are referred to the supplier database. This configuration provides some measure of enhanced server performance by distributing the workload to more than one directory.

**Figure 5-4** Single Supplier Replication

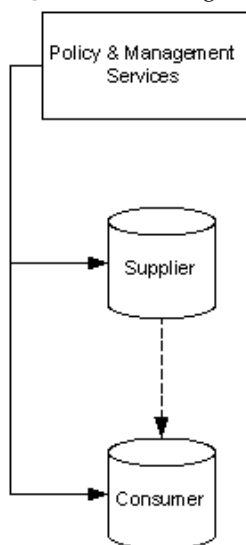
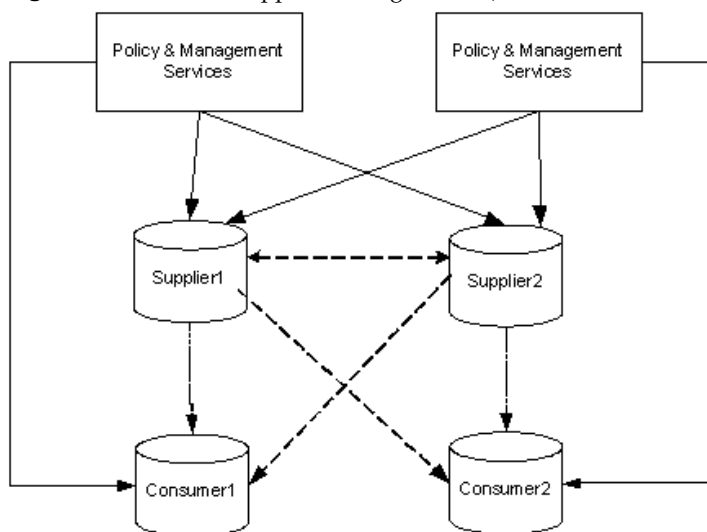


Figure 5-5 illustrates a multi-supplier configuration using multiple instances of Identity Server. This configuration provides failover protection as well as high availability, resulting in further enhanced server performance.

**Figure 5-5** Multi Supplier Configuration (also known as Multi-Master Replication)



The following steps can be used to configure replication at the root or top level of the Identity Server directory tree when Identity Server has not yet been installed. They can also be used to configure replication at the default organization level.

1. Install supplier and consumer Directory Servers.

See the Sun ONE Directory Server Installation Guide for detailed instructions.

2. Set up replication agreements between the supplier and consumer, and verify that the directory referrals and updates are working properly.

See the Sun ONE Directory Server Administrator's Guide for detailed instructions.

---

**NOTE** It might be necessary to migrate existing Directory Server data to work with this version of Identity Server. For instructions on how to do this see the Sun ONE Identity Server *Migration Guide*.

---

3. If deploying Identity Server and Directory Server for the first time, or there is no plan to use existing user data, run the Sun Java Enterprise System 2003Q4 installation program to install Identity Server.

During installation, answer yes when asked if there is an existing Directory Server, and specify the host name and port number for a supplier Directory Server you installed in Step 1.

4. In the server where Identity Server Management and Policy services are installed, modify the following file:

`/IdentityServer_base/SUNWam/lib/AMConfig.properties`

- a. Modify the following properties to reflect the host and port number of a consumer Directory Server installed in Step 1.
  - `com.ipplanet.am.directory.host`
  - `com.ipplanet.am.directory.port`
- b. Modify the following properties to reflect the number of times Identity Server should continue to make the same request when the requested entry is not found.  
`com.ipplanet.am.replica.retries`
- c. Modify the following properties to reflect the number of milliseconds Identity Server should allow to elapse between retries.

`com.ipplanet.am.replica.delay.between.retries`

5. In each Identity Server Authentication module enabled, specify the consumer directory installed in Step 1. In the following substeps, the LDAP Authentication module is used as an example:
  - a. In the Identity Server console, choose Service Management.
  - b. Locate the authentication module to be reconfigured, clicking the Properties arrow.
  - c. In the right pane:
    - In the first field named LDAP Server and Port, type the host name and port number for the primary (consumer) Directory Server. For example, `consumer1.example.com:389`.
    - In the second field named LDAP Server and Port, type the host name and port number for the secondary (or supplier) Directory Server. For example, `supplier1.example.com:389`.
  - d. Click Submit.
6. In the following file:
 

```
/Identity_Server_root/SUNWam/config/ums/serverconfig.xml
```

, specify the host name and port number of the consumer directory installed in Step 1 as in Code Example 5-1.

**Code Example 5-1**    `serverconfig.xml` Replication Modification

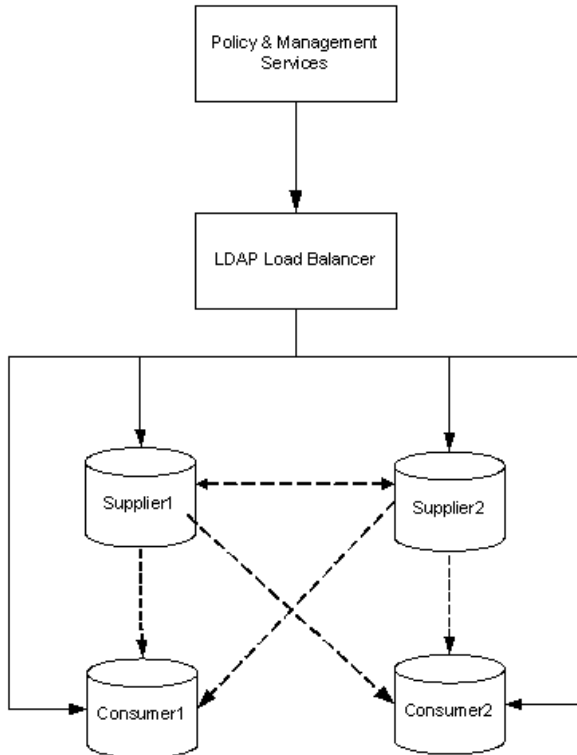
```
<iPlanetDataAccessLayer>
<ServerGroup name="default" minConnPool="1"
maxConnPool="10">
<Server name="Server1"
host="consumer1.madisonparc.com" port="389"
type="SIMPLE" />
```

7. Restart Identity Server with the following command:

```
/Identity_Server_root/SUNWam/bin/amserver start
```

## Configuring With a Load Balancer

Figure 5-6 illustrates a multi-supplier configuration that includes Directory Proxy Server. This configuration takes full advantage of Identity Server support for failover, high availability, and managed load-balancing.

**Figure 5-6** Multi-supplier Replication With Load Balancer

Using LDAP load balancers adds a layer of high availability and directory failover protection beyond the basic level that comes with Identity Server. For example, Directory Proxy Server can specify what percentage of the load gets redistributed to each of server. And, when all back-end LDAP servers become unavailable, it continues to manage request traffic and begins rejecting client queries. If you choose to install a load-balancer, Identity Server must be configured to recognize the application.

1. Before configuring:
  - a. Set up the Directory Servers for replication. For comprehensive information about directory replication and for detailed setup instructions, see "Managing Replication" in the Sun ONE Directory Server Administrator's Guide.
  - b. Install and configure the LDAP load balancer. Follow the instructions in the documentation that comes with the product.

2. In the file `/IdentityServer_base/SUNWam/lib/AMconfig.properties`, modify the following properties to reflect the host and port number of a consumer Directory Server installed in Step a.

```
com.iplanet.am.directory.host
```

```
com.iplanet.am.directory.port
```

3. For each Identity Server Authentication module enabled, specify the consumer Directory Server installed in Step a. In the following substeps, the LDAP Authentication module is used as an example:
  - a. In the Identity Server console, choose Service Management.
  - b. Locate the authentication module to be reconfigured, clicking the Properties arrow.
  - c. In the right pane:
    - In the first field named LDAP Server and Port, type the host name and port number for the primary (consumer) Directory Server using the form `proxyhostname:port`.
    - In the second field named LDAP Server and Port, enter nothing.
  - d. Click Submit.
4. In the `/IdentityServer_base/SUNWam/config/ums/serverconfig.xml` file, specify the host name and port number of the consumer directory installed in Step a as in Code Example 5-2.

**Code Example 5-2**    `serverconfig.xml` Load Balancer Modification

```
<iPlanetDataAccessLayer>
  <ServerGroup name="default" minConnPool="1"
  maxConnPool="10">
    <Server name="Server1"
    host="idar.example.com" port="389"
    type="SIMPLE" />
  </ServerGroup>
</iPlanetDataAccessLayer>
```

5. Restart Identity Server with the following command:

```
/IdentityServer_base/SUNWam/bin/amserver start
```

---

**NOTE**    See Appendix D, "Load Balancer Configuration" for more specifics about configuring Identity Server with a load balancer.

---

## Replication Caveats

There may be situations in which directory replication cannot be implemented in an Identity Server deployment. For example, authentication server host names or IP addresses must be the same. This precludes using geographically separated, replicated Identity Server servers. The remote servers would not be able to perform authentication against servers that are only local to their respective LANs. For comprehensive information on planning and implementing Directory Server replication, see the *Sun ONE Directory Server Deployment Guide*.

## Implementing Federation Management

This section details how to configure a deployment using the Federation Management functionality of Identity Server. This scenario assumes that Domain A, Domain B and Domain C each contain two instances of Identity Server and one instance of Directory Server. Domain A and Domain B have Service Provider A and Service Provider B, respectively. Domain C has Identity Provider C.

---

**NOTE** Identity Server has the capability to host multiple providers in the same server instance.

---

In order to manage this federation scenario correctly there are two concepts to understand.

- A *Hosted Provider* defines a specific instance of Identity Server as one acting as a service provider.
- A *Remote Provider* contains data related to any type of provider hosted on machine different from the one on which the instance of Identity Server is currently being configured. This may or may not be an instance of Identity Server; it could be any compliant service provider or identity provider.

So, in the above defined scenario, Service Provider A will configure itself as a hosted provider and Service Provider B and Service Provider C as remote providers. Service Provider B will configure itself as a hosted provider and Service Provider A and Service Provider C as remote providers. Service Provider C will configure itself as a hosted provider and Service Provider A and Service Provider B as remote providers. Once these are configured, an Authentication Domain can be created in each of the domains.



# Installed Product Layout

This chapter describes the product software layout after a typical installation. It contains the following sections:

- The Sun Java Enterprise System 2003Q4 Base Directory
- The SUNWam Directory

## The Sun Java Enterprise System 2003Q4 Base Directory

During Identity Server installation, if no directory is specified, the default `/opt` is used. Because Identity Server installation is now accomplished using the Sun Java Enterprise System 2003Q4 installer, any or all products chosen for installation might be installed in this directory. When Identity Server is chosen, the default directory will contain a sub-directory named `SUNWam`. `SUNWam` contains all Identity Server shared binary files and command line tools. For a complete listing of sub-directories installed, see the Sun Java Enterprise System 2003Q4 Installation Guide.

# The SUNWam Directory

Identity Server packages are installed in in this directory. Brief descriptions of each package are provided following the listing itself.

---

**NOTE** Directories shown here reflect a product installation for the Solaris Operating Environment. File names and extensions may differ for installations on other platforms. After installing Identity Server, a full list of installed path names for a particular package can be obtained by using the `pkgchk(1M)` utility, `pkgchk -v package-name`.

---

The SUNWam directory contains the following directories and files:

- agents/
- bin/
- config -> /etc/opt/SUNWam/config/
- console.war
- docs/
- dtd/
- ldaplib/
- ldif/
- lib/
- locale/
- migration/
- password.war\*
- public\_html/
- samples/
- services.war
- share/
- web-apps/

## /opt/SUNWam/agents/

This directory contains tools, header files and configuration files specific to Identity Server policy agents. For more information on what is included here, see the *Web Policy Agents Guide* or the *J2EE Policy Agents Guide*.

## /opt/SUNWam/bin/

This directory contains the command line tools shipped with Identity Server. They are detailed in Table A-1. More information can be found in the Command Line Reference Guide section of the Sun ONE Identity Server *Administration Guide*.

**Table A-1** Identity Server Command Line Utilities

Utility	What it does
VerifyArchive*	This is the deprecated version of amverifyarchive installed in 6.1 for backward compatibility.
am2bak*	This utility performs a backup of Identity Server components.
amadmin*	amadmin is used to load XML service files into Directory Server and to perform batch administrative tasks on the DIT.
ampassword*	This utility is used to change the passwords set for the Identity Server administrator or users.
amserver*	The amserver command line executable is used to create, start, stop, and delete Identity Server instances.
amverifyarchive*	This utility verifies the log archives. Verification of an archive detects possible tampering and/or deletion of any file in the archive.
bak2am*	This utility performs a restore of the Identity Server components backed-up by the am2back utility.
ldapmodify*	ldapmodify edits the contents of an LDAP directory, either by adding new entries or by modifying existing ones.
ldapsearch*	ldapsearch issues search requests to an LDAP directory and displays the result as LDIF text.

## /config ---> /etc/opt/SUNWam/config/

The `config` directory is a symbolic link to `/etc/opt/SUNWam/config`. This directory contains the XML service and configuration files detailed in Table A-2. The XML files are not generally used for configuration. If they are modified, they must be manually reloaded into the Directory Server data store. (Any changes in the server are not synched with these files.) Information on all the XML files in this directory can be found in the *Sun ONE Identity Server Customization And API Guide*.

**Table A-2** XML Service and Configuration Files

Sub-directory	What it contains
ums	<ul style="list-style-type: none"><li>• <code>serverconfig.xml</code> provides configuration information for the Sun™ One Identity Server regarding the Sun ONE Directory Server that is used as its data store.</li><li>• <code>ums.xml</code> provides a set of templates that contain LDAP configuration information for identity-related objects managed using Identity Server.</li></ul>

**Table A-2** XML Service and Configuration Files (*Continued*)

Sub-directory	What it contains
xml	<ul style="list-style-type: none"> <li>• amAdminConsole.xml</li> <li>• amAuth.xml</li> <li>• amAuthAnonymous.xml</li> <li>• amAuthCert.xml</li> <li>• amAuthConfig.xml</li> <li>• amAuthHTTPBasic.xml</li> <li>• amAuthLDAP.xml</li> <li>• amAuthMembership.xml</li> <li>• amAuthNT.xml</li> <li>• amAuthRadius.xml</li> <li>• amAuthSafeWord.xml</li> <li>• amAuthSecurID.xml</li> <li>• amAuthUnix.xml</li> <li>• amAuthenticationDomainConfig.xml</li> <li>• amClientData.xml</li> <li>• amClientDetection.xml</li> <li>• amEntrySpecific.xml</li> <li>• amDSS.xml</li> <li>• amG11NSettings.xml</li> <li>• amLogging.xml</li> <li>• amNaming.xml</li> <li>• amPasswordReset.xml</li> <li>• amPlatform.xml</li> <li>• amPolicy.xml</li> <li>• amPolicyConfig.xml</li> <li>• amProviderConfig.xml</li> <li>• amSAML.xml</li> <li>• amSession.xml</li> <li>• amUser.xml</li> <li>• amWebAgent.xml</li> </ul>

## /opt/SUNWam/console.war

console.war is the *web application archive* (WAR) that contains files pertaining to the Identity Server console application. Information on all WAR files in this directory can be found in the *Sun ONE Identity Server Customization And API Guide*.

## /opt/SUNWam/docs

The docs directory contains the HTML and related files used for the API Javadocs. Following is a list of the included files.

- META-INF/
- allclasses-frame.html
- com/
- deprecated-list.html
- en\_US/
- help-doc.html
- index-all.html
- index.html
- overview-frame.html
- overview-summary.html
- overview-tree.html
- package-list
- packages.html
- serialized-form.html
- stylesheet.css

## /opt/SUNWam/dtd

The dtd directory contains all the Document Type Definition (DTD) files used by Identity Server. A DTD defines the structure for XML files accessed by Identity Server. More information on the DTD files can be found in the chapter on Service Management in the *Sun ONE Identity Server Customization And API Guide*. Table A-3 contains a listing of the DTD files.

**Table A-3** DTD Files in Identity Server

File	Purpose
Auth_Module_Properties.dtd	Defines the structure for XML files used by the authentication modules to specify their properties.
amAdmin.dtd	Defines the structure for XML files used to perform batch LDAP operations on the directory tree using the amAdmin command line tool.
amWebAgent.dtd	Defines the structure for XML files used to handle requests from, and send responses to, web agents. This file is deprecated and remains for purposes of backward compatibility.
policy.dtd	Defines the structure for XML files used to store policies in Directory Server.
remote-auth.dtd	Defines the structure for XML files used by the Authentication Service's remote Authentication API.
server-config.dtd	defines the structure for serverconfig.xml which details ID, host and port information for all server and user types.
sms.dtd	Defines the structure for XML service files.
web-app_2_2.dtd	Defines the structure for XML files used by the Identity Server deployment container to deploy J2EE applications.

## /opt/SUNWam/ldaplib

ldaplib contains the shared object (.so) files needed to run the LDAP utilities included with Identity Server. Table A-4 details a listing of the files.

**Table A-4** Shared Object Files

Sub-directory	Included files
ldapsdk	<ul style="list-style-type: none"> <li>• libicudata.so.2</li> <li>• libicui18n.so.2</li> <li>• libicuuc.so.2</li> <li>• libldap50.so</li> <li>• libprldap50.so</li> <li>• libsasl.so</li> <li>• libssldap50.so</li> </ul>

## /opt/SUNWam/ldif

The ldif directory contains LDIF files needed for populating the Directory Server data store when installing Identity Server. Table A-5 contains a list of the included LDIF files.

**Table A-5** LDIF Files

File	Purpose
ds_remote_schema.ldif	During installation, this file loads the Identity Server-specific LDAP schema object classes and attributes (iplanet-am-managed-people-container, etc.) needed to store Identity Server data in Directory Server.
ds_remote_schema_uninstall.ldif	This file is used to remove the Identity Server-specific LDAP schema object classes and attributes from Directory Server.
sunone_schema2.ldif	During installation, this file loads the Identity Server-specific LDAP schema object classes and attributes defined by Sun Microsystems' internal Schema 2 document.

## /opt/SUNWam/lib

The lib directory contains JAR files and additional shared object (.so) files. Following is a list of the files and sub-directories.

- AMConfig.properties
- AMConfig.properties.template
- LogConfig.properties
- SSOConfig.properties
- acmecrypt.jar\*
- am\_logging.jar
- am\_sdk.jar
- am\_services.jar
- am\_sso\_provider.jar
- commons-logging.jar
- crimsondeb.jar\*
- dom.jar



- dom4j.jar
- endorsed sub-directory includes:
  - dom.jar
  - sax.jar
  - xalan.jar
  - xercesImpl.jar
  - xslyc.jar
- iaik\_ssl.jar\*
- jakarta-log4j-1.2.6.jar
- jax-qname.jar
- jaxm-api.jar
- jaxm-runtime.jar
- jaxp-api.jar
- jaxrpc-api.jar
- jaxrpc-ri.jar
- jdk\_logging.jar
- jss subdirectory
- libaceclnt.so
- libamsdk.so
- libamsdk.so.2
- libamutils.so
- libxml2.so
- libxml2.so.2
- mail.jar
- saaj-api.jar
- saaj-ri.jar
- sax.jar
- servlet.jar
- swec.jar\*
- xalan.jar

- xercesImpl.jar
- xmlsec.jar
- xsltc.jar
- AMConfig.properties
- AMConfig.properties.template

## /opt/SUNWam/locale

The locale directory contains the localization properties files. Following is a list of the included files. Each properties file listed also includes a corresponding English localization file as, for example, amAdminCLI\_en.properties.

- amAdminCLI.properties
- amAdminConsole.properties
- amAdminModuleMsgs.properties
- amAuth.properties
- amAuthAnonymous.properties
- amAuthApplication.properties
- amAuthCert.properties
- amAuthConfig.properties
- amAuthContext.properties
- amAuthContextLocal.properties
- amAuthenticationDomainConfig.properties
- amAuthHTTPBasic.properties
- amAuthLDAP.properties
- amAuthMembership.properties
- amAuthNT.properties
- amAuthRadius.properties
- amAuthSafeWord.properties
- amAuthSecurID.properties
- amAuthUI.properties
- amAuthUnix.properties

- `amClientData.properties`
- `amClientDetection.properties`
- `amEntrySpecific.properties`
- `amFederation.properties`
- `amGllnSettings.properties`
- `amLogging.properties`
- `amNaming.properties`
- `amPasswordReset.properties`
- `amPasswordResetModuleMsgs.properties`
- `amPlatform.properties`
- `amPll.properties`
- `amPolicy.properties`
- `amPolicyConfig.properties`
- `amProfile.properties`
- `amProviderConfig.properties`
- `amSAML.properties`
- `amSDK.properties`
- `amSession.properties`
- `amSSOProvider.properties`
- `amUser.properties`
- `amUtilMsgs.properties`
- `amWebAgent.properties`
- `getEncoding.class`
- `LC_MESSAGES/`
- `netscape/`

## /opt/SUNWam/migration

The migration directory contains PERL scripts used to migrate data from earlier versions of Identity Server. Following is a list of the migration scripts included with Identity Server. More information on migrating data can be found in the *Sun ONE Identity Server Migration Guide*.

- 60to61/
- update-assignable-dynamic-groups.pl\*
- update-filtered-groups.pl\*
- update-groups.pl\*
- update-o.pl\*
- update-ou.pl\*
- update-people.pl\*
- update-static-groups.pl\*
- update-users.pl\*

## /opt/SUNWam/password.war

password.war is the *web application archive* (WAR) that contains files pertaining to the Identity Server password reset application. Information on all WAR files in this directory can be found in the *Sun ONE Identity Server Customization And API Guide*.

## /opt/SUNWam/public\_html

The public\_html directory contains the HTML and related files used for the online help.

## /opt/SUNWam/samples

The samples directory contains the sub-directories listed in Table A-6. These sub-directories hold the samples for the respective functionality. More specific information on the samples themselves can be found in the respective chapter of the *Sun ONE Identity Server Customization And API Guide*.

**Table A-6** Directories For Functional Samples

Sub-directory Name
Readme.html
admin/
appserver/
authentication/
console/
liberty/
policy/
saml/
sample.css
sso/
sunLogo.gif
um/

## /opt/SUNWam/services.war

`services.war` is the *web application archive* (WAR) that contains files pertaining to all Identity Server services not included in the previously mentioned WAR files. Information on all WAR files in this directory can be found in the *Sun ONE Identity Server Customization And API Guide*.

## /opt/SUNWam/share

The share directory contains a sub `bin/` directory that contains additional utilities used internally by Identity Server.

- `/bin/ammultiserverinstall*`
- `/bin/amsecuridd*`
- `/bin/amserver.instance_template*`
- `/bin/amunixd*/bin/checkport*`
- `/bin/configds*`

- /bin/unconfigds\*
- /bin/wsutils.ksh\*

## /opt/SUNWam/web-apps

The web-apps directory contains the directories in which Identity Server J2EE web application are deployed on a web container. It contains the following directories and files.

- applications/ is the directory where Identity Server Console is deployed. It contains:
  - META-INF/
  - WEB-INF/
  - console
    - auth/
    - base/
    - css/
    - federation/
    - html/
    - images/
    - index.html
    - js/
    - policy/
    - service/
    - session/
    - user/
  - index.html
- introduction/ is the directory where Identity Server Liberty Common Domain component is deployed. It contains:
  - META-INF
  - WEB-INF
- introduction.war

- `password/` is the directory where the Identity Server Password Synchronization component is deployed.
  - `META-INF/`
  - `WEB-INF/`
  - `index.html`
  - `password/`
- `services/` is the directory where Identity Server Core Services are deployed. It contains:
  - `META-INF/`
  - `WEB-INF/`
  - `admin/`
  - `config/`
  - `css/`
  - `docs/`
  - `fed_css/`
  - `fed_images/`
  - `images/`
  - `index.html`
  - `js/`
  - `login_images/`





# The User Session Life Cycle

In providing access management services, Sun™ ONE Identity Server allows for the creation of session objects which are used to track user interaction with web applications across multiple HyperText Transfer Protocol (HTTP) requests. This chapter describes the life cycle of a session by tracing the protocol interactions of the Identity Server components. It contains the following sections:

- Overview
- The Request
- The Authentication
- The Session Token
- The Policy
- The Requested Page
- Single Sign-On Requests
- Terminating a Session

## Overview

The following sections trace the protocol interaction of Identity Server components as they provide authentication and authorization services to a user requesting access to protected resources via a web browser. In so doing, it also demonstrates the session life cycle.

# The Request

To begin the scenario, an unauthenticated user makes a request for a protected resource which is sent to the server. The resource is protected by a policy agent. Code Example 5-3 depicts the GET request sent by the browser.

## Code Example 5-3 GET Request Header

```
GET / HTTP/1.1
Host:application.sun.com:8089
```

With Identity Server, all access requests are implicitly denied unless explicitly allowed by the presence of a valid session token (programmatically, `SSOToken`) and the existence of an applicable policy allowing access.

---

**NOTE** This behavior can be inverted based upon deployment criteria.

---

Since, in this case, a session token is not provided, the policy agent redirects the request to the Authentication Service. Code Example 5-4 depicts the redirect information sent back to the requesting browser. It includes the URI to the Authentication Service and a `goto` parameter that contains the URL of the original request.

## Code Example 5-4 GET Response With Redirect Information

```
HTTP/1.1 302 Moved Temporarily
Location:
http://identityserver.sun.com:58081/amserver/UI/Login?goto=http%3A%2F%2Fapp
lication.sun.com%3A8089%2Findex.html
```

Biding by the HTTP, the user's browser allows the redirect and makes a request to the Authentication Service URI. Code Example 5-5 depicts the GET Request sent to the Authentication Service.

**Code Example 5-5** GET Request Redirected To Authentication Service

```

GET
/amserver/UI/Login?goto=http%3A%2F%2Fapplication.sun.com%3A8089%2Findex.htm
1 HTTP/1.1
Host: identityserver.sun.com:58081

```

## The Authentication

Upon receiving the request for authentication, the Authentication Service determines which authentication module to present the user with based upon Identity Server configuration and the request parameters. As in all new authentication requests, an invalid session token is created by the Session Service to track the user interaction. (This session token also contains an encrypted session identifier which is a randomly-generated string that represents the user.) The session token is set in a cookie (`iPlanetDirectoryPro` by default) which, along with a form asking the user for the appropriate credentials, is sent by the Authentication Service in response to the request for authentication.

---

**NOTE** The protocol of the form (HTML, WML, etc.) is determined by the Client Detection Service based on the client requesting the authentication. More information on this service can be found in the *Sun ONE Identity Server Customization And API Guide*.

---

Code Example 5-6 depicts the header of the HTML authentication form requesting authentication credentials from the user. (The HTML itself has been deleted for the sake of brevity.)

**Code Example 5-6** Authentication Form Returned To User

```

HTTP/1.1 200 OK
Server: Sun-ONE-Web-Server/6.1
X-dsnameversion: 6.1
Set-cookie: JSESSIONID=DE271E3F2D52473B409DD8A7C58C24A5;Path=/amserver
Set-cookie:
iPlanetDirectoryPro=AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4%3D;Doma
in=.sun.com;Path=/
<html>

```

**Code Example 5-6** Authentication Form Returned To User

```
Login Form...  
</html>
```

After receiving the form, the user enters their authentication credentials and posts them to Identity Server. Normally, this process might have occurred over SSL to protect the password attribute. For demonstration purposes, the credentials posted in Code Example 5-7 was made over clear HTTP.

**Code Example 5-7** POST Credentials Returned To Identity Server

```
POST /amsserver/UI/Login HTTP/1.1  
Host: identityserver.sun.com:58081  
Cookie: JSESSIONID=DE271E3F2D52473B409DD8A7C58C24A5;  
iPlanetDirectoryPro=AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4%3D  
Content-Type: application/x-www-form-urlencoded  
  
IDToken1=user1&IDToken2=password
```

Once the credentials are received by the Authentication Service, they are validated by the appropriate authentication module. Assuming they pass muster, the state of the session token is changed to valid, and session information(Login time, Authentication Scheme, Authentication Level, etc.) is stored. The iPlanetDirectoryPro cookie now contains a valid session token, and the server replies to the browser with a redirect to the originally requested resource. Code Example 5-8 contains this redirect response.

**Code Example 5-8** Redirect Back To Originally Requested Resource

```
HTTP/1.1 302 Moved Temporarily  
Server: Sun-ONE-Web-Server/6.1  
X-autherrorcode: 0  
X-dsnameversion: 6.1  
Location: http://application.sun.com:8089/index.html
```

Biding by the HTTP, the user's browser allows the redirect back to the original resource and, once again, requests access. This time, the session token created during the authentication process is included with the request.

**Code Example 5-9**    Redirected GET Request Header With Token

```
GET /index.html HTTP/1.1
Host: application.sun.com:8089
Referer:
http://identityserver.sun.com:58081/amserver/UI/Login?goto=http%3A%2F%2Fapp
lication.sun.com%3A8089%2Findex.html
Cookie:
iPlanetDirectoryPro=AQIC5wM2LY4SfcywF1TefDRmqlthG54qrg27LiyS8LnHAj4%3D
```

## The Session Token

The policy agent, once again, intercepts the request. Since the request now contains a session token in the same DNS domain as Identity Server, the agent attempts to determine the validity of said token and its associated session. First, it must learn where the session originated. To do this, it contacts the Naming Service. The Naming Service allows clients to find the service URL for the internal services used by Identity Server. This information can then be used for communication regarding a session. The Naming Service decrypts the session and returns the corresponding URLs which will then be used to obtain information about the session from the applicable services. Code Example 5-10 contains the POST request for naming information.

**Code Example 5-10**    POST Request For Naming Information

```
POST /amserver/namingservice HTTP/1.0
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RequestSet vers="1.0" svcid="com.iplanet.am.naming" reqid="9">
<Request><![CDATA[
<NamingRequest vers="1.0" reqid="2"
sessid="AQIC5wM2LY4SfcywF1TefDRmqlthG54qrg27LiyS8LnHAj4=">
<GetNamingProfile>
</GetNamingProfile>
</NamingRequest>]]>
</Request>
</RequestSet>
```

Code Example 5-11 contains the response to the POST request for naming information. Note the attribute names and their corresponding URL values.

### Code Example 5-11 Response With Naming Information

```

HTTP/1.1 200 OK
Content-type: text/html

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ResponseSet vers="1.0" svcid="com.ipplanet.am.naming" reqid="9">
<Response><![CDATA[<NamingResponse vers="1.0" reqid="2">
<GetNamingProfile>
<Attribute name="ipplanet-am-naming-policy-url"
value="http://identityserver.sun.com:58081/amserver/policyservice"></Attrib
ute>
<Attribute name="ipplanet-am-naming-session-class"
value="com.ipplanet.dpro.session.service.SessionRequestHandler"></Attribute>
<Attribute name="ipplanet-am-naming-session-url"
value="http://identityserver.sun.com:58081/amserver/sessionsevice"></Attri
bute>
<Attribute name="ipplanet-am-naming-samlawareservlet-url"
value="http://identityserver.sun.com:58081/amserver/SAMLAwareServlet"></Att
ribute>
<Attribute name="serviceObjectClasses"
value="ipplanet-am-naming-service"></Attribute>
<Attribute name="ipplanet-am-naming-auth-url"
value="http://identityserver.sun.com:58081/amserver/authservice"></Attribut
e>
<Attribute name="ipplanet-am-naming-profile-class"
value="com.ipplanet.dpro.profile.agent.ProfileService"></Attribute>
<Attribute name="ipplanet-am-naming-samlassertionmanager-url"
value="http://identityserver.sun.com:58081/amserver/AssertionManagerServlet
/AssertionManagerIF"></Attribute>
<Attribute name="ipplanet-am-naming-umservice-url"
value="http://identityserver.sun.com:58081/amserver/UserManagementServlet/"
></Attribute>
<Attribute name="01"
value="http://identityserver.sun.com:58081"></Attribute>
<Attribute name="ipplanet-am-naming-policy-class"
value="com.sun.identity.policy.remote.PolicyRequestHandler"></Attribute>
<Attribute name="ipplanet-am-naming-logging-class"
value="com.sun.identity.log.service.LogService"></Attribute>
<Attribute name="ipplanet-am-naming-profile-url"
value="http://identityserver.sun.com:58081/amserver/profileservice"></Attri
bute>
<Attribute name="ipplanet-am-naming-samlsoapreceiver-url"
value="http://identityserver.sun.com:58081/amserver/SAMLSOAPReceiver"></Att
ribute>

```

**Code Example 5-11** Response With Naming Information (*Continued*)

```

<Attribute name="iplanet-am-naming-logging-url"
value="http://identityserver.sun.com:58081/amserver/loggingservice"></Attribute>
<Attribute name="iplanet-am-naming-fsassertionmanager-url"
value="http://identityserver.sun.com:58081/amserver/FSAssertionManagerServlet/FSAssertionManagerIF"></Attribute>
<Attribute name="iplanet-am-platform-server-list"
value="http://identityserver.sun.com:58081"></Attribute>
<Attribute name="iplanet-am-naming-samlpostervlet-url"
value="http://identityserver.sun.com:58081/amserver/SAMLPOSTProfileServlet"
></Attribute>
<Attribute name="iplanet-am-naming-auth-class"
value="com.sun.identity.authentication.server.AuthXMLHandler"></Attribute>
</GetNamingProfile>
</NamingResponse>]]></Response>
</ResponseSet>

```

Using the information provided by the Naming Service, the policy agent makes a POST request to the Session Service to validate the included session token. Code Example 5-12 is the POST request to the Session Service.

**Code Example 5-12** POST Request To Session Service For Session Validation

```

POST /amserver/sessionsservice HTTP/1.0
Host: identityserver.sun.com
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RequestSet vers="1.0" svcid="Session" reqid="10">
<Request><![CDATA[
<SessionRequest vers="1.0" reqid="4">
<GetSession reset="true">
<SessionID>AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4=</SessionID>
</GetSession>
</SessionRequest>]]>
</Request>
<Request><![CDATA[
<SessionRequest vers="1.0" reqid="5">
<AddSessionListener>
<URL>http://application.sun.com:8089/amagent/UpdateAgentCacheServlet?shortcircuit=false</URL>
<SessionID>AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4=</SessionID>
</AddSessionListener>
</SessionRequest>]]>
</Request>
</RequestSet>

```

The Session Service receives the request and checks the session token for validity. Assuming the session has not timed out or been invalidated for other reasons, the Session Service responds that the session is valid. This assertion is coupled with supporting information about the session itself.

---

**NOTE** A Session Listener is also registered for the session. This allows notification of the policy agent should there be a change in the session's state or validity.

---

Code Example 5-13 details the Session Service's response.

**Code Example 5-13** Session Service Response Asserting A Session's Validity

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ResponseSet vers="1.0" svcid="session" reqid="10">
<Response><![CDATA[<SessionResponse vers="1.0" reqid="4">
<GetSession>
<Session sid="AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4="
stype="user" cid="uid=user1,ou=people,dc=sun,dc=org"
cdomain="dc=sun,dc=org" maxtime="300" maxidle="120" maxcaching="3"
timeidle="0" timeleft="17993" state="valid">
<Property name="authInstant" value="2003-10-02T01:38:23Z"></Property>
<Property name="clientType" value="genericHTML"></Property>
<Property name="CharSet" value="UTF-8"></Property>
<Property name="Locale" value="en_US"></Property>
<Property name="UserToken" value="user1"></Property>
<Property name="loginURL"
value="http://identityserver.sun.com:58081/amserver/UI/Login"></Property>
<Property name="SessionHandle"
value="shandle:AQIC5wM2LY4Sfcxlv0iI7LJwWcusZAdkM3CHmIoeehu6urc="></Property
>
<Property name="Host" value="192.168.1.100"></Property>
<Property name="AuthType" value="LDAP"></Property>
<Property name="Principals"
value="uid=user1,ou=People,dc=sun,dc=org|AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg
27LiyS8LnHAj4="></Property>
<Property name="cookieSupport" value="true"></Property>
<Property name="Organization" value="dc=sun,dc=org"></Property>
<Property name="USERS_DN"
value="uid=user1,ou=people,dc=sun,dc=org"></Property>
<Property name="AuthLevel" value="0"></Property>
<Property name="UserId" value="user1"></Property>
<Property name="HostName" value="192.168.1.100"></Property>
<Property name="Principal"
value="uid=user1,ou=people,dc=sun,dc=org"></Property>
</Session></GetSession>
</SessionResponse]]></Response>
<Response><![CDATA[<SessionResponse vers="1.0" reqid="5">
```



**Code Example 5-13** Session Service Response Asserting A Session's Validity (*Continued*)

```

<AddSessionListener>
<OK></OK>
</AddSessionListener>
</SessionResponse>]]></Response>
</ResponseSet>

```

## The Policy

Having received a reply indicating that the session token presented by the user was valid, the policy agent must now determine if the user can be granted access to the resource being requested. It formulates a request to the Policy Service, requesting decisions regarding resources in its portion of the HTTP namespace. Included with this request is additional environmental information which might impact conditions set on a configured policy, such as IP address or DNS name. Code Example 5-14 is the POST request sent to the Policy Service URI for information.

**Code Example 5-14** POST Request For Policy Information

```

POST /amservice/policyservice HTTP/1.0
Host: identityserver.sun.com
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RequestSet vers="1.0" svcid="Policy" reqid="11">
<Request><![CDATA[
<PolicyService version="1.0">
<PolicyRequest requestId="3"
appSSOToken="AQIC5wM2LY4SfczlhkwdQHfKgzxYu0qWY+DFCB9VGmknzvm=">
<GetResourceResults
userSSOToken="AQIC5wM2LY4SfcywF1TefDRmqlthG54qrg27LiyS8LnHAj4="
serviceName="iPlanetAMWebAgentService"
resourceName="http://application.sun.com:8089/" resourceScope="subtree">
<EnvParameters>
<AttributeValuePair>
<Attribute name="requestDnsName"/>
<Value>drnick</Value>
<Value>drnick.sun.com</Value>
</AttributeValuePair>
<AttributeValuePair>
<Attribute name="requestIp"/>
<Value>192.168.1.100</Value>
</AttributeValuePair>
</EnvParameters>
</GetResourceResults>
</PolicyRequest>
</PolicyService>]]>

```

**Code Example 5-14** POST Request For Policy Information (*Continued*)

```
</Request>
</RequestSet>
```

After receiving the request, the Policy Service checks for policies that contain resource definitions which apply to the request.

---

**NOTE** Policies are cached in Identity Server. If it has not yet warmed its cache, the policies will first be loaded out of Directory Server.

---

Once policies are found that apply to the request, the Policy Service checks to see if the user identified by the session token is a member of any of the Policy Subjects. If policies are found that match the resource and the user is a valid subject, additional conditions of the policy are evaluated. (For example, *Is it the right time of day?*, *Are they coming from the correct network?*, etc.) If all the conditions are met, the Policy Service determines that the user can be granted access and responds to the policy agent with the allow decision. Code Example 5-15 is the response from the Policy Service verifying that the user can have access to the protected resource.

**Code Example 5-15** Allow Policy Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ResponseSet vers="1.0" svcid="policy" reqid="11">
<Response><![CDATA[<PolicyService version="1.0">
<PolicyResponse requestId="3">
<ResourceResult name="http://application.sun.com:8089/">
<PolicyDecision>
<ActionDecision timeToLive="1065121515571">
<AttributeValuePair>
<Attribute name="POST"/>
<Value>allow</Value>
</AttributeValuePair>
<Advices>
</Advices>
</ActionDecision>
<ActionDecision timeToLive="1065121515571">
<AttributeValuePair>
<Attribute name="GET"/>
<Value>allow</Value>
</AttributeValuePair>
<Advices>
</Advices>
</ActionDecision>
```

**Code Example 5-15** Allow Policy Response (*Continued*)

```

</PolicyDecision>
</ResourceResult>
</PolicyResponse>
</PolicyService[]>
</Response>

```

## The Requested Page

Having received a decision from the Policy Service, the policy agent must act on this access information. In this case, an *allow* decision was issued for GET and POST operations. This matches the operation requested by the user, thus the policy agent will allow access. The decision is cached, along with the session token, so that subsequent requests can be checked using the cache and need not contact Identity Server. The cache will expire after an administrator-defined interval has passed or upon an explicit notification of change in policy or session status. However, before the user is granted access, the action must be logged so an audit trail is maintained. The policy agent issues a logging request to the Logging Service. Code Example 5-16 is that request.

**Code Example 5-16** Logging Request From Policy Agent

```

POST /amservice/loggingservice HTTP/1.0
Host: identityserver.sun.com
Content-Length: 416
Accept: text/xml
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RequestSet vers="1.0" svcid="Logging" reqid="12">
<Request><![CDATA[
<logRecWrite reqid="2"><log logName="amAuthLog"
sid="AQIC5wM2LY4SfczlhkwdQHfKgzxYu0qWY+DFCB9VGmknzVM="></log><logRecord><recType>Agent</recType><recMsg>User user1 was allowed access to
http://application.sun.com:8089/index.html.</recMsg></logRecord></logRecWrite>]]></Request>
</RequestSet>

```

The Logging Service receives the request and, based upon configuration, logs the request to an optionally signed file, or JDBC store. A response is then returned to the policy agent notifying it of the log. Code Example 5-17 is that response.

**Code Example 5-17** Return Response Notifying Agent Of Log

```
HTTP/1.1 200 OK
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ResponseSet vers="1.0" svcid="iplanet.webtop.service.logging" reqid="12">
<Response><![CDATA[OK]]></Response>
</ResponseSet>
```

Now, the policy agent allows access to the requested resource. Code Example 5-18 is the HTML page for the requested resource. (The HTML itself has been deleted in the interest of brevity.)

**Code Example 5-18** Agent Allows Access To Requested Page

```
HTTP/1.1 200 Ok
Content-type: text/html

<HTML>
The requested page....
</HTML>
```

## Single Sign-On Requests

This section follows two threads. The first occurs when the authenticated user requests a protected resource on a different server in the same DNS domain for which a session token has already been validated; this first thread uses the single sign-on functionality. The second thread occurs when the authenticated user requests a protected resource on a different server in a different DNS domain; this second thread uses the cross domain single sign-on functionality.

### Thread One: Single Sign-On

Having received the requested page, the user makes a request for a protected resource on a different server. Code Example 5-19 is this second request. Note that the session token is included with the request. This makes single sign-on possible.

**Code Example 5-19** Second Request With A Valid Session Token

```
GET / HTTP/1.1
Host: webservice.sun.com:8090
Cookie:
iPlanetDirectoryPro=AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4%3D
```

Since a session token is present, the policy agent does not need to get user authentication. Therefore, in this case, the procedure detailed in “The Authentication” on page 107 is skipped. The agent, though, has not seen the session token before (i.e.: it has no cached entry), so it will follow the procedures as detailed in “The Session Token” on page 109, “The Policy” on page 113 and “The Requested Page” on page 115.

---

**NOTE** The following is heavily edited from the procedures detailed in “The Session Token” on page 109, “The Policy” on page 113 and “The Requested Page” on page 115.

---

1. The Naming Service is contacted. Code Example 5-20 is the request to the Naming Service for URLs of the internal services used by Identity Server. The Naming Service decrypts the session and returns the corresponding URLs which will then be used to obtain session data.

**Code Example 5-20** POST Request For Naming Service

```
POST /amservice/namingservice HTTP/1.0
Host: identityserver.sun.com

...
<NamingRequest vers="1.0" reqid="2"
sessid="AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4=">
....

HTTP/1.1 200 OK

<ResponseSet vers="1.0" svcid="com.iplanet.am.naming" reqid="9">
....
</ResponseSet>
```

- Based upon the reply from the Naming Service, the appropriate Session Service is contacted. Code Example 5-21 is the request sent to the Session Service containing the session identifier.

**Code Example 5-21** POST Request To Session Service

```
POST /amserver/sessionsservice HTTP/1.0

...
<SessionID>AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4=</SessionID>
....

HTTP/1.1 200 OK

....
<Session sid="AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4="
stype="user" cid="uid=user1,ou=people,dc=sun,dc=org"
cdomain="dc=sun,dc=org" maxtime="300" maxidle="120" maxcaching="3"
timeidle="0" timeleft="17991" state="valid">
....
```

- Assuming a valid session reply from the Session Service, a POST Request is made to the Policy Service. Code Example 5-22 is the request to the Policy Service for policy information pertaining to the authenticated user.

**Code Example 5-22** POST Request To Policy Service

```
POST /amserver/policyservice HTTP/1.0

...
<GetResourceResults
userSSOToken="AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4="
serviceName="iPlanetAMWebAgentService"
resourceName="http://webservice.sun.com:8090/" resourceScope="subtree">
...

```

- In this case, a policy allowing access to the protected resource is not found and the Policy Service responds appropriately. Code Example 5-23 is the negative response.

**Code Example 5-23** Response From Policy Service Negating Access

```

HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ResponseSet vers="1.0" svcid="policy" reqid="11">
<Response><![CDATA[<PolicyService version="1.0">
<PolicyResponse requestId="3">
<ResourceResult name="http://webservice.sun.com:8090/">
<PolicyDecision>
</PolicyDecision>
</ResourceResult>
</PolicyResponse>
</PolicyService>]]>
</Response>
</ResponseSet>

```

- As the policy agent has been set up for logging, it logs this denial of access. Code Example 5-24 is the request to the Logging Service to write a log indicating the user is not allowed access to the requested resource.

**Code Example 5-24** POST Request To Logging Service

```

POST /amserver/loggingservice HTTP/1.0

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RequestSet vers="1.0" svcid="Logging" reqid="12">
<Request><![CDATA[
<logRecWrite reqid="2"><log logName="amAuthLog"
sid="AQIC5wM2LY4SfcyueSeNMEFDFRRLub8BfjaxeyDvTPXlVnA="></log><logRecord><re
cType>Agent</recType><recMsg>User user1 was denied access to
http://webservice.sun.com:8090/index.html.</recMsg></logRecord></logRecWrit
e]]></Request>
</RequestSet>

HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ResponseSet vers="1.0" svcid="iplanet.webtop.service.logging" reqid="12">
<Response><![CDATA[OK]]></Response>
</ResponseSet>

```

- The policy agent then issues a *Forbidden* message to the user. Code Example 5-25 is the HTML page for this message. Optionally, the user could then be redirected to an administrator-specified page indicating they were denied access.

**Code Example 5-25** HTML Page With Access Denied Message

```
HTTP/1.1 403 Forbidden

<HTML><HEAD><TITLE>Forbidden</TITLE></HEAD>
<BODY><H1>Forbidden</H1>
Your client is not allowed to access the requested object.
</BODY></HTML>
```

## Thread Two: Cross Domain Single Sign-On

Having been denied access to the second requested page, the user makes a request for a protected resource on a server in a *different* DNS domain. Since Identity Server uses HTTP Domain cookies (<http://www.ietf.org/rfc/rfc2965.txt>) to transfer the session token between applications, the token is not automatically handed up to the policy agent with the request as it was in Thread One: Single Sign-On.

- It is the job of Cross Domain Single Sign-On (CDSSO) protocol within Identity Server to transfer the token into the new DNS domain so it is usable for applications in that domain. Code Example 5-26 is the request for access to a protected application in a different DNS domain without the session token.

**Code Example 5-26** GET Request For Protected Application In Different DNS Domain

```
GET /index.html?sunwMethod=GET HTTP/1.1
Host: webservice.java.com:8088
```

- The policy agent sees that no session token is present. However, in this case, the agent is configured for CDSSO so, rather than redirect to the Authentication Service, it redirects to the CDSSO Controller Service which uses Liberty Protocols to transfer sessions, and hence, the redirect includes the relevant Liberty parameters. Code Example 5-27 is the redirect with the Liberty parameters (by way of the browser).



**Code Example 5-27** Redirect To The CDSSO Controller Service Via The Browser

```

HTTP/1.1 302 Moved Temporarily
Location:
http://identityserver.sun.com:58081/amserver/cdcservlet?goto=http%3A%2F%2Fwebservice.java.com%3A8088%2Findex.html%3FsunMethod%3DGET&refererservlet=http%3A%2F%2Fwebservice.java.com%3A8088%2Findex.html%3FsunMethod%3DGET&RequestID=29980&MajorVersion=1&MinorVersion=0&ProviderID=http%3A%2F%2Fwebservice.java.com%3A8088%2Famagent&IssueInstant=2003-10-02T04%3A25%3A06Z&ForceAuthn=false&IsPassive=false&Federate=false

```

3. Bidding by the HTTP, the user's browser allows the redirect. This time the request contains the session token, as it is a cookie in the primary domain. Code Example 5-28 is the HTTP redirect from the browser to the CDSSO Controller Service with the session token.

**Code Example 5-28** HTTP Redirect From Browser With Session Token

```

GET
/amserver/cdcservlet?goto=http%3A%2F%2Fwebservice.java.com%3A8088%2Findex.html%3FsunMethod%3DGET&refererservlet=http%3A%2F%2Fwebservice.java.com%3A8088%2Findex.html%3FsunMethod%3DGET&RequestID=29980&MajorVersion=1&MinorVersion=0&ProviderID=http%3A%2F%2Fwebservice.java.com%3A8088%2Famagent&IssueInstant=2003-10-02T04%3A25%3A06Z&ForceAuthn=false&IsPassive=false&Federate=false HTTP/1.1
Host: identityserver.sun.com:58081
Cookie:
iPlanetDirectoryPro=AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4%3D

```

4. The CDC Servlet (in the CDSSO Controller Service) receives the session token, formulates a Liberty Post Profile response detailing session information, and replies to the browser. Code Example 5-29 is the reply.

**Code Example 5-29** POST Reply To The Browser

```

HTTP/1.1 200 OK
Server: Sun-ONE-Web-Server/6.1
Date: Thu, 02 Oct 2003 01:38:28 GMT
Content-type: text/html
Pragma: no-cache

```

**Code Example 5-29** POST Reply To The Browser

```

Transfer-encoding: chunked

<HTML>
<BODY Onload="document.Response.submit()">
<FORM NAME="Response" METHOD="POST"
ACTION="http://webservice.java.com:8088/index.html?sunwMethod=GET">
<INPUT TYPE="HIDDEN" NAME="LARES" VALUE="<ENCODED_LIBERTY_DOC>" />
</FORM>
</BODY></HTML>

```

5. The user's browser automatically submits the form containing the Liberty document to the policy agent, based upon the Action and the Javascript included in the Body tags onLoad. Code Example 5-30 is the Browser POST to the policy agent.

**Code Example 5-30** Browser POST To Policy Agent

```

POST /index.html?sunwMethod=GET HTTP/1.1
Host: webservice.java.com:8088
Referer:
http://identityserver.sun.com:58081/amserver/cdcservlet?goto=http%3A%2F%2Fwebservice.java.com%3A8088%2Findex.html%3FsunwMethod%3DGET&refererservlet=http%3A%2F%2Fwebservice.java.com%3A8088%2Findex.html%3FsunwMethod%3DGET&RequestID=29980&MajorVersion=1&MinorVersion=0&ProviderID=http%3A%2F%2Fwebservice.java.com%3A8088%2Famagent&IssueInstant=2003-10-02T04%3A25%3A06Z&ForceAuthn=false&IsPassive=false&Federate=false
Content-Type: application/x-www-form-urlencoded

LARES=<ENCODED_LIBERTY_DOC>

```

6. The policy agent receives the Liberty document and extracts the user's session information. It must now validate the session as a regular agent would so it follows the procedures as detailed in "The Session Token" on page 109, "The Policy" on page 113 and "The Requested Page" on page 115 before allowing or denying access to the resource.

---

**NOTE** “The Session Token” on page 109, “The Policy” on page 113 and “The Requested Page” on page 115 are not reiterated here in the interest of brevity.

---

In this case, the session token was determined to be valid, and the user is allowed access. The policy agent responds to the user with the requested document, and sets the session token in a cookie for the new DNS domain. The cookie can now be used by all agents in the new domain. Code Example 5-31 is the returned HTML page with the new DNS domain cookie set. (The HTML itself has been deleted in the interest of brevity.)

**Code Example 5-31** User Allowed Access To HTML Page With New Cookie

```
TTP/1.1 200 Ok
Set-cookie:
iPlanetDirectoryPro=AQIC5wM2LY4Sfcwwte3peDKZILScZ40uK%2FsU0I0WQQP6xXA%3D;Do
main=.java.com;Path=/

<HTML>
...
</HTML>
```

## Terminating a Session

1. Having authenticated, performed SSO and CDSSO, the user now wishes to end their session. Sessions can be terminated by administrators, due to idle or maximum timeouts, or via an explicit user logout. In this case, the user logs out by clicking on a link to the Logout Service. Code Example 5-32 is the request to access the Logout Service.

**Code Example 5-32** GET Request For Logout Service

```
GET /amsrver/UI/Logout HTTP/1.1
Host: identityserver.sun.com:58081
Cookie:
iPlanetDirectoryPro=AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4%3D
```

2. The Logout Service receives the Logout request, marks the user's session as destroyed, sets a new invalid value for the session token, and returns a successful logout page to the user. Code Example 5-33 details the HTML page sent to the user after a successful logout. (The HTML itself has been deleted in the interest of brevity.)

**Code Example 5-33** Successful Logout HTML Page Returned To User

```
HTTP/1.1 200 OK
Server: Sun-ONE-Web-Server/6.1
Set-cookie:
iPlanetDirectoryPro=AQICv6c3Z1VfvgCgpLlaEqkqM70TLV240TB1XkPa%2BLtA8bXvC7c5X
ABU95Ta8UJi6dQZhXEUSgTRBWHXQ6kcx8qgw%3D%3D;Domain=.sun.com;Path=/

<title>Sun ONE Identity Server (Logout)</title>
...
```

3. Since the user's session status has changed, it is the responsibility of the Session Service to notify applications which have expressed interest in the session. In this case, each of the policy agents was configured for Session Notification, and each is sent a document instructing the agent that the session is invalid. This causes it to be flushed from cache.

**Code Example 5-34** POST For Session Notification

```
POST /amagent/UpdateAgentCacheServlet?shortcircuit=false HTTP/1.1
Host: application.sun.com:8089

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NotificationSet vers="1.0" svcid="session" notid="8">
<Notification><![CDATA[<SessionNotification vers="1.0" notid="8">
<Session sid="AQIC5wM2LY4SfcywFlTefDRmqlthG54qrg27LiyS8LnHAj4="
stype="user" cid="uid=user1,ou=people,dc=sun,dc=org"
cdomain="dc=sun,dc=org" maxtime="300" maxidle="120" maxcaching="3"
timeidle="3" timeleft="17983" state="destroyed">
<Property name="authInstant" value="2003-10-02T01:38:23Z"></Property>
<Property name="clientType" value="genericHTML"></Property>
<Property name="CharSet" value="UTF-8"></Property>
<Property name="Locale" value="en_US"></Property>
<Property name="UserToken" value="user1"></Property>
<Property name="loginURL"
value="http://identityserver.sun.com:58081/amserver/UI/Login"></Property>
<Property name="SessionHandle"
value="shandle:AQIC5wM2LY4SfcxlvOiI7LjWwCusZAdkM3CHmIoeehu6urc="></Property
>
<Property name="Host" value="192.168.1.100"></Property>
```

**Code Example 5-34** POST For Session Notification (*Continued*)

```

<Property name="AuthType" value="LDAP"></Property>
<Property name="Principals"
value="uid=user1,ou=People,dc=sun,dc=org|AQIC5wM2LY4SfcywF1TefDRmqlthG54qrg
27LiyS8LnHAj4="></Property>
<Property name="cookieSupport" value="true"></Property>
<Property name="Organization" value="dc=sun,dc=org"></Property>
<Property name="USERS_DN"
value="uid=user1,ou=people,dc=sun,dc=org"></Property>
<Property name="AuthLevel" value="0"></Property>
<Property name="UserId" value="user1"></Property>
<Property name="HostName" value="192.168.1.100"></Property>
<Property name="Principal"
value="uid=user1,ou=people,dc=sun,dc=org"></Property>
</Session>
<Type>5</Type>
<Time>1065058714469</Time>
</SessionNotification>]]></Notification>
</NotificationSet>

```

```

POST /amagent/UpdateAgentCacheServlet?shortcircuit=false HTTP/1.1
Host: webservice.sun.com:8090

```

```

....
<Session sid="AQIC5wM2LY4SfcywF1TefDRmqlthG54qrg27LiyS8LnHAj4="
stype="user" cid="uid=user1,ou=people,dc=sun,dc=org"
cdomain="dc=sun,dc=org" maxtime="300" maxidle="120" maxcaching="3"
timeidle="3" timeleft="17983" state="destroyed">
...

```

```

POST /amagent/UpdateAgentCacheServlet?shortcircuit=false HTTP/1.1
Host: webservice.java.com:8088

```

```

...
<Session sid="AQIC5wM2LY4SfcywF1TefDRmqlthG54qrg27LiyS8LnHAj4="
stype="user" cid="uid=user1,ou=people,dc=sun,dc=org"
cdomain="dc=sun,dc=org" maxtime="300" maxidle="120" maxcaching="3"
timeidle="3" timeleft="17983" state="destroyed">
...

```

4. Following the receipt of the session notification, the policy agents flush the session from cache, and the session's life cycle is complete.

## Terminating a Session

# Authenticate Against Active Directory

Sun™ ONE Identity Server provides the ability to authenticate against a variety of backend sources. Primarily, Sun ONE Directory Server and LDAP are used, but the LDAP Authentication module can be configured to use Microsoft® Active Directory® as its authentication source. This appendix contains the following sections:

- Overview
- Setting Up Active Directory Authentication
- Troubleshooting

## Overview

Since Active Directory is an LDAPv3-compliant directory server, it is just a matter of configuring the Identity Server to recognize Active Directory as an authentication source. There are two options to consider when defining this configuration. The administrator can either point all LDAP authentication to the Active Directory or a new authentication module can be created and registered for this purpose.

---

**NOTE** The LDAP Authentication module can be used in the manner described in this chapter against any LDAP v3 compliant server.

---

## Point to Existing LDAP Authentication Module

Using this method, all users attempting to authenticate with the LDAP Authentication module would be verified against the Active Directory. This is a change from the default LDAP authentication which is verified against Sun ONE Directory Server. This appendix focuses on this scenario.

## Create New Active Directory Authentication Module

The other option is to create and register an authentication module which specifically performs LDAP authentication against the Active Directory. Identity Server does not allow multiple instances of the same authentication module to be used with different server configurations so a new class would be needed for this approach. This option is used if the Active Directory suffix and Directory Server suffix are not the same. It is not covered in this appendix, but information on how to create a custom authentication module can be found in the *Sun ONE Identity Server Customization And API Guide*.

## Multiple LDAP Sub-Configurations

An administrator can define multiple LDAP authentication module configurations under one organization. Although these additional configurations are not visible from the console, they work in conjunction with the primary configuration if an initial search for the requesting user's authorization is not found. For example, one organization can define a search through LDAP servers for authentication in two different domains or it can configure multiple user naming attributes in one domain. For the latter, which has only one text field in the console, if a user is not found using the primary search criteria, the LDAP module will then search using the second scope. More information on this option can be found in the *Sun ONE Identity Server Customization And API Guide*.

# Setting Up Active Directory Authentication

The following steps detail the procedure for the LDAP Authentication Module to verify credentials against Active Directory.



1. Install and configure Identity Server.

The following steps will be modifying the default configuration of the LDAP Authentication module. This is potentially hazardous, and can cause a lockout. See Troubleshooting if access to Identity Server is denied.

2. Select Dynamically Created in the User Profile attribute of the Core Authentication Service for the organization.

Identity Server requires that an account exist within Directory Server for authorization despite authentication being delegated to an external source. The options for this are:

- Use a meta directory to synchronize accounts.
- Enable dynamic profile creation which allows Identity Server to look for an account for the user in question. If none exists, the account is automatically created with the same account name used in Active Directory.

---

**NOTE** More information on this attribute and how to enable it can be found in *Sun ONE Identity Server Administration Guide*.

---

3. Change the Primary LDAP Server and Port attribute in the LDAP Authentication module to the host name and port number of the Active Directory in the form: `hostname.domain.com:389`.

4. Change the DN to Start User Search attribute to the proper base for the Active Directory.

Simply specifying the base suffix will not work in this attribute. Generally, it would be the `cn=Users` plus suffixes as in: `cn=Users,dc=domain,dc=com`.

5. Change the DN for Root User bind attribute to a user with the right to read the Active Directory and update the password.

Anonymous access to the Active Directory is not allowed, so a bind account is needed. It is simply an account for Active Directory that has read ability on the attribute to which the user will authenticate. An example might be `cn=administrator,cn=Users,dc=domain,dc=com`. The password should be updated for this entry.

## 6. Change the User Naming Attribute.

This attribute is the one that the LDAP Authentication module will search for in Active Directory, and pass back to match the UID in Directory Server. For example, if a meta-directory synchs the two systems and entries in Directory Server were stored with mail as the RDN, `userPrincipalName` might be specified here as it stores a mail address in Active Directory. In this case, it might be best to specify `sAMAccountName`, as it is the attribute in Active Directory most like UID.

## 7. Change the User Entry Search Attributes.

This attribute is used to locate the account in Active Directory. It should correspond to the attribute with which people use to log in. For example, if users login using their Common Name, the value of this attribute would be `cn`. In this case, it might be best to specify `sAMAccountName`, as it is the attribute in Active Directory most like UID.

---

**NOTE** This attribute may contain multiple values, each of which will be tried.

---

## 8. Deselect the Return User DN to Auth attribute.

This attribute forces the LDAP Authentication module to return the DN it authenticated as to Identity Server, saving the need to search for it again for authorization. The DN though will not be the same in Active Directory as in the Directory Server so it needs to be turned off. This enables Identity Server to take the value returned from Active Directory as the attribute specified in the User Naming Attribute in Step 6, and search Directory Server using the attribute specified in Core Authentication (`uid` by default). So, in this example, LDAP Authentication will search Active Directory for `sAMAccountName=UID_entered_by_user`, and after authentication, search the Directory Server for `uid=sAMAccountName` from Active Directory.

## 9. Add the `amAdmin` account to Active Directory.

An account needs to be created with the `sAMAccountName` of `amAdmin`, so that `amAdmin` may continue to login to Identity Server. This user is created in Active Directory by:

- a. Selecting Active Directory Users and Computers from Start -> Programs -> Administrative Tools.
- b. Right click on the Users node in the left pane and select New -> User.
- c. Fill in appropriate information and specify `amAdmin` as the account name.

10. Verify that `amAdmin` is able to log in to Identity Server.
  11. Verify that a user defined in Active Directory is able to log in to Identity Server.
- Identity Server is now configured for authentication against Active Directory.

## Troubleshooting

Modifying the LDAP Authentication information makes it possible for a lock out to occur. The following procedures detail how to overcome this.

### Quick Access To Identity Server

The DN found in the `com.iplanet.authentication.super.user` property of `AMConfig.properties` allows `amAdmin` to login to Identity Server via the LDAP Authentication module. `amAdmin` is the Identity Server manager for Directory Server. The value of this property is the full DN of the `amAdmin` account, `uid=amAdmin,ou=People,root-suffix`. The full DN is entered in the User Name field. In this case, the instance of Directory Server configured in `AMConfig.properties` is used and the LDAP Authentication module parameters are not.

### Reconfigure Using Directory Server

Authentication configuration information is stored in Directory Server and, thus, the entry is easily modified.

`ou=default,ou=OrganizationConfig,ou=1.0,ou=iPlanetAMAuthLDAPService,ou=service,root-suffix` is the object that defines the Authentication Configuration Service. Modify the `iplanetKeyValue` attribute to correct any errors made. The relevant values are:

- `iplanet-am-auth-ldap-server`
- `iplanet-am-auth-ldap-base-dn`
- `iplanet-am-auth-ldap-user-naming-attribute`
- `iplanet-am-auth-ldap-user-search-attributes`
- `iplanet-am-auth-ldap-return-user-dn`
- `iplanet-am-auth-ldap-bind-dn`

- `iplanet-am-auth-ldap-bind-passwd`

---

**NOTE** The value of this property is encrypted using the `ampassword` utility.

---

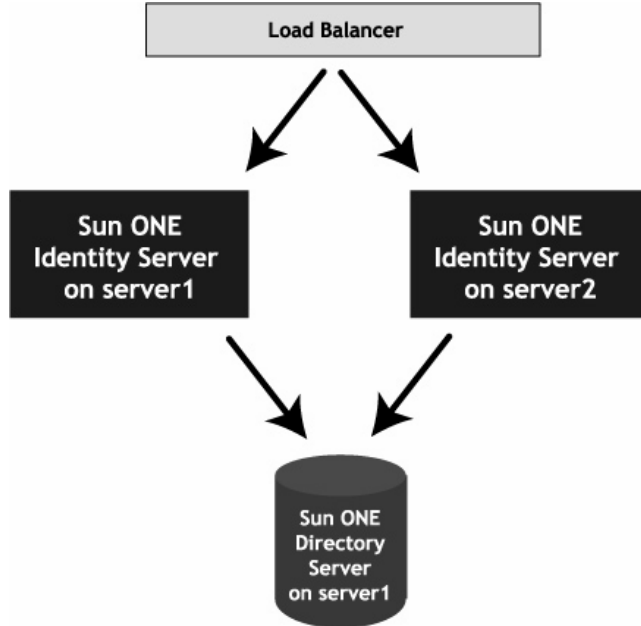
# Load Balancer Configuration

Sun ONE Identity Server can be configured to work with a load balancer. This chapter details the features of load balancing and how it can be accomplished. The chapter contains the following sections:

- Load Balancer Overview
- Configuring The Load Balancer
- Confirming The Configuration

## Load Balancer Overview

Load balancing is used to divvy up between two or more servers the amount of work usually done by one, allowing more work to get done in the same amount of time. In general, this means that all users are served faster. Load balancing can be implemented with hardware, software, or a combination of both. Figure 5-7 illustrates how an Identity Server deployment can be configured with a load balancer. It is important in this configuration that all instances of Identity Server share the same Directory Server. Once configured, the load balancer (and thus, all Identity Server services) is accessed via the URL  
`http://loadbalancer_host.domain:port/amconsole.`

**Figure 5-7** Identity Server Configuration With Load Balancer

## Sticky Sessions

A load balancer deployed with Identity Server must support sticky sessions. A *sticky session* specifies that once a session is created by a given server, subsequent requests from the user will continue to be routed to that same server in order to preserve session information. Since Identity Server uses cookies to relay session information, the load balancer needs to redirect to the server that created the session. Without sticky sessions, all servers would have to be trusted and performance might be impaired.

## Resonate Central Dispatch Installation

Resonate Central Dispatch is a software-based load balancer. The first step in configuring Identity Server to work with a load balancer is installation. Assuming two physical servers, ensure that the machines are in the same subnet. On machine1, install Sun ONE Web Server, Sun ONE Directory Server, and Identity Server (in that order), pointing the instance of Identity Server to the installed

instance of Directory Server. On machine2, install Sun ONE Web Server and Identity Server, pointing the instance of Identity Server to the instance of Directory Server installed on server1. The Central Dispatch software should be installed as follows:

- machine1 contains CDNode, CDMaster, and CDAction.
- machine2 contains CDNode, CDAdaptor, and CDAction.

A Reporter Agent will be automatically installed on both machines during the installation process itself. The terms defined in Table 5-1 are specific to Central Dispatch and might be used in the configuration procedures.

**Table 5-1** Resonate Central Dispatch Terms Defined

Central Dispatch Term	Defintion
CDMaster	The Central Dispatch Master is the graphical user interface used to manage and monitor a single (or multiple) Central Dispatch site(s). All Central Dispatch configurations will be applied using this console.
Node	A Node is an instance of Identity Server configured as such via the CDMaster console. A Node can be configured as either a scheduler or server.
CDAdapter	The Central Dispatch Adapter is a proxy that provides a link between a single Central Dispatch site and the CDMaster.
CDAction	CDAction is a command line utility used to configure, monitor, and administer a Central Dispatch site.

For more information on installing Central Dispatch and the product in general, see the documentation set that is provided with the software.

## Configuring The Load Balancer

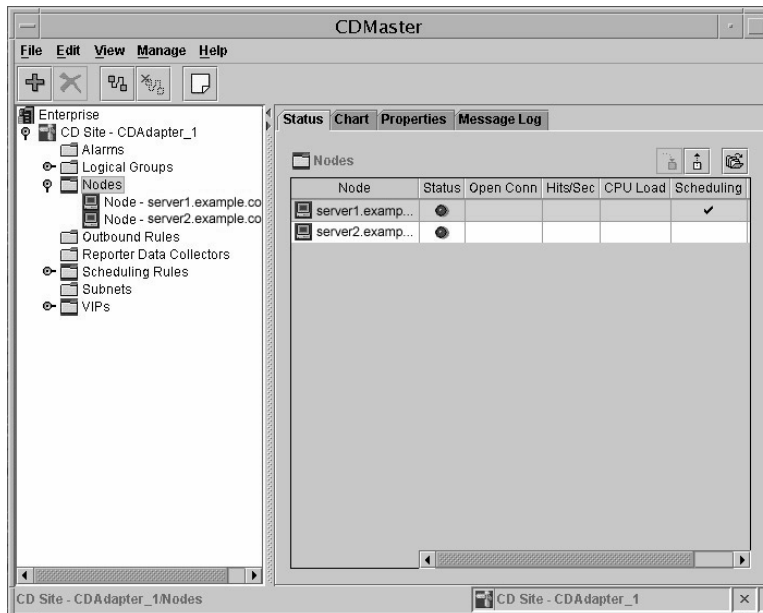
“Sticky Sessions” can be implemented using either the `setcookie` function or load balancer cookies. The procedures detailed in the next sections illustrate how to configure the load balancer for both of these options. The steps involved relate to the Resonate Central Dispatch load balancer although they can be modified to work with any load balancer software.

## To Configure Central Dispatch for setcookie

1. Create two Solaris users (cdadmin and cdmon) using admintool.
2. Launch the CDMaster console on server1.  
  
Change to the default directory (/usr/local/resonate/cd/cdmaster/bin) and run ./cdmaster. When instructed, connect to the CDAdapter installed on machine2.
3. Click on Nodes in the left frame of CDMaster and create one node for each of the two installed instances of Identity Server.

Figure 5-8 is a screenshot of the CDMaster console illustrating this step.

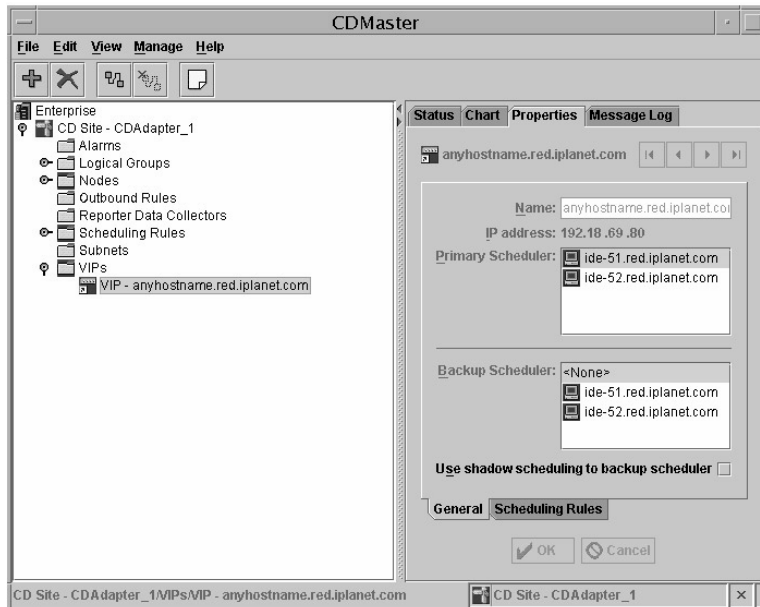
**Figure 5-8** Creating Nodes With Resonate



4. Click on VIPs in the left frame of CDMaster and create a new virtual IP address for the host on which the load balancer is installed.

Figure 5-9 is a screenshot of the CDMaster console illustrating this and the following steps. Ensure that the Primary Scheduler and Backup Scheduler are correctly configured.



**Figure 5-9** Creating a new Virtual IP address

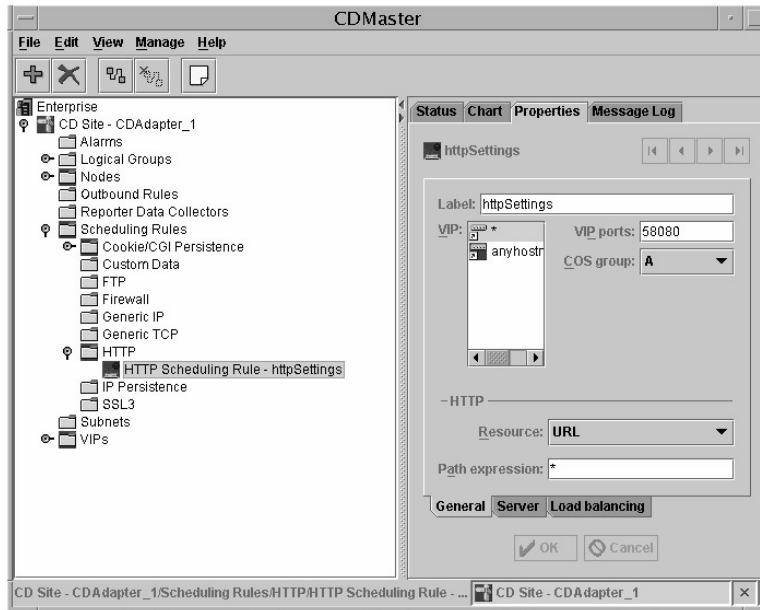
5. Click on the Scheduling Rules tab in the right frame of VIPs and select HTTP to configure a HTTP scheduling rule as follows:
  - a. Under the Properties tab, ensure that the host on which the load balancer is installed is listed as a virtual IP.
  - b. Check that the VIP port is the same one defined for the virtual IP.
  - c. Select URL for Resource.
6. Click on HTTP under Scheduling Rules in the left frame of CDMaster.

Figure 5-10 on page 138 is a screenshot of the CDMaster console illustrating this and the following steps.

7. Select the Server tab at the bottom of the right frame.
 

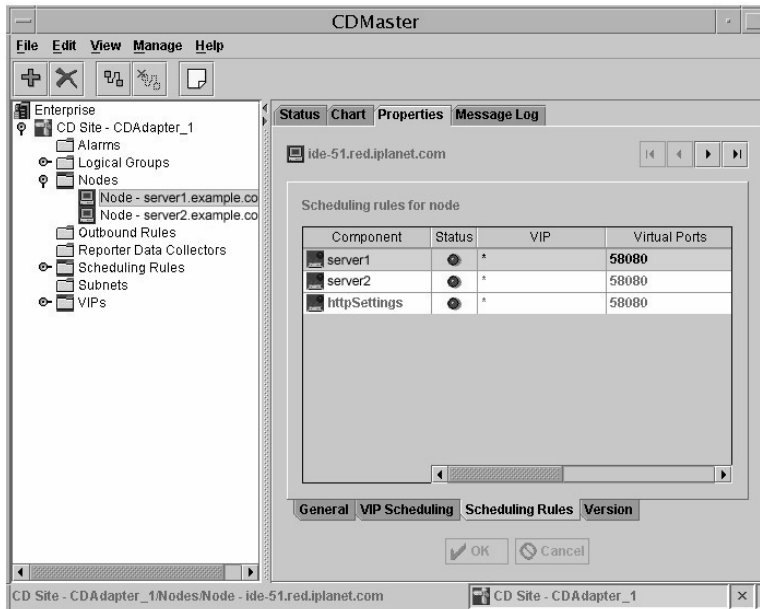
Ensure that the servers are selected.
8. Click on the Load Balancing tab at the bottom of the right frame and select Round Robin (Basic).

**Figure 5-10** Configuring HTTP Scheduling Rules



9. Click on Nodes in the left frame of CDMaster and select the configured node for the second instance of Identity Server, server2.

Figure 5-11 is a screenshot of the CDMaster console illustrating this and the following steps.

**Figure 5-11** Configuring Nodes With CDMaster

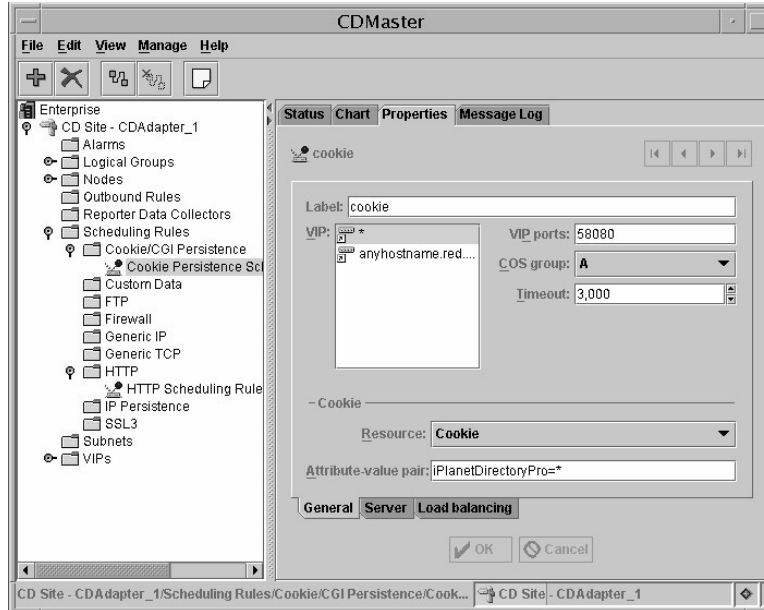
10. Click on Properties at the top of the right frame and make sure that the Alias is server2.example.com and that Server Enabled and Server auto enabled are selected.
11. Click the VIP Scheduling tab at the bottom of the right frame and check that the Primary virtual IP is configured for the host on which the load balancer is installed.
12. Select the Scheduling Rules tab at the bottom of the right frame and make sure that all the servers and ports are listed under Component.
13. Repeat Step 9 on page 138 through Step 12 above for server1.

Configure the first instance of Identity Server (server1) as detailed, deleting Step 11 which configures server2 as the Primary under the Scheduling Rules tab.

14. Click on Scheduling Rules in the left frame of CDMaster.

Figure 5-12 is a screenshot of the CDMaster console illustrating this and the following steps.

**Figure 5-12** Configuring a Cookie Persistence Scheduling Rule



15. Select Cookie/CGI Persistence and create a Cookie Persistence Scheduling Rule.

The Attribute-value pair is defined as `iPlanetDirectoryPro=*`.

16. Label the rule and make sure that the correct port for the virtual IP is defined.

The VIP list should contain the configured host on which the load balancer is installed.

17. Select the Server tab at the bottom of the right frame and make sure that both servers are checked.

18. Select the Load Balancing tab at the bottom of the right frame and select Round Robin (Basic).

This completes the configuration of Central Dispatch for `setcookie`. Continue to the next section, "To Configure Identity Server for `setcookie`," to complete the deployment.

## To Configure Identity Server for setcookie

The Identity Server configuration for both server1 and server2 needs to be updated to recognize the load balancer when using setcookie.

1. Login as amadmin to the Identity Server instance installed on server1.
2. Add the value of the host machine on which the load balancer is installed to the Organization Aliases attribute.  
  
View the top-level organization under the Identity Management tab to find the Organization Aliases attribute.
3. Add server2 to the Platform List attribute in the Platform Service under the Service Management tab.
4. Set the fqdnMap property in `AMConfig.properties`.

---

**CAUTION** This step can not be done using the Identity Server console.

---

By default, the fqdnMap property is commented out. Remove the # and configure the property as:

```
com.sun.identity.server.fqdnMap[loadbalancer_host.domain]=loadbalancer_hos
t.domain
```

5. Restart server1 and server2.

See “Confirming The Configuration” on page 144 for instructions on how to verify the configuration procedure.

## To Configure Central Dispatch with Load Balancer Cookies

1. Create two Solaris users (cdadmin and cdmon) using admintool.
2. Launch the CDMaster console on server1.

Change to the default directory (`/usr/local/resonate/cd/cdmaster/bin`) and run `./cdmaster`. When instructed, connect to the CDAdapter installed on server2.

3. Click on Nodes in the left frame of CDMaster and create one node for each of the two installed instances of Identity Server.

4. Click on VIPs in the left frame of CDMaster and create a new virtual IP address for the host on which the load balancer is installed.

Ensure that the Primary Scheduler and Backup Scheduler are configured.

5. Click on the Scheduling Rules tab in the right frame of VIPs and select HTTP to configure a HTTP scheduling rule as follows:
  - a. Under the Properties tab, ensure that the host on which the load balancer is installed is listed as a virtual IP.
  - b. Check that the VIP port is the same one as defined for virtual IP.
  - c. Select URL for Resource.

6. Click on HTTP under Scheduling Rules in the left frame of CDMaster.

7. Select the Server tab at the bottom of the right frame.

Ensure that the servers are selected.

8. Click on the Load Balancing tab at the bottom of the right frame and select Round Robin (Basic).

9. Click on Nodes in the left frame of CDMaster and select the configured node for the second instance of Identity Server, server2.

10. Click on Properties at the top of the right frame and make sure that the Alias is server2.example.com and that Server Enabled and Server auto enabled are selected.

11. Click the VIP Scheduling tab at the bottom of the right frame and check that the Primary virtual IP is configured for the host on which the load balancer is installed.

12. Select the Scheduling Rules tab at the bottom of the right frame and make sure that all the servers and ports are listed under Component.

13. Repeat Step 9 on page 138 through Step 12 above for server1.

Configure the first instance of Identity Server (server1) as detailed, deleting Step 11 which configures server2 as the Primary under the Scheduling Rules tab.

14. Click on Scheduling Rules in the left frame of CDMaster.

15. Select Cookie/CGI Persistence and create two Cookie Persistence Scheduling Rules: one for server1 and one for server2.

16. Label server1 and make sure that the correct port for the virtual IP is defined.  
The VIP list must also contain the configured host on which the load balancer is installed.
17. Choose cookie as the Resource and define the Attribute-value pair as server1=server1.
18. Select the Server tab at the bottom of the right frame and make sure that both servers are selected.
19. Select the Load Balancing tab at the bottom of the right frame and select Round Robin (Basic).
20. Repeat Step 14 through Step 19 for server2.

This completes the configuration of Central Dispatch for load balancer cookies. Continue to the next section, "To Configure Identity Server with Load Balancer Cookies," to complete the deployment.

## To Configure Identity Server with Load Balancer Cookies

The Identity Server configuration for both server1 and server2 needs to be updated to recognize the load balancer.

1. Login as amadmin to the Identity Server instance installed on server1.
2. Add the value of the host machine on which the load balancer is installed to the Organization Aliases attribute.  
View the top-level organization under the Identity Management tab to find the Organization Aliases attribute.
3. Add server2 to the Platform List attribute in the Platform Service under the Service Management tab.
4. Set the fqdnMap property in `AMConfig.properties`.

---

**CAUTION** This step can not be done using the Identity Server console.

---

By default, the `fqdnMap` property is commented out. Remove the `#` and configure the property as:

```
com.sun.identity.server.fqdnMap[loadbalancer_host.domain]=loadbalancer_host.domain
```

5. Add the following properties to the `AMConfig.properties` files on `server1` and `server2`, respectively.
  - a. Set the cookie name and value on `server1` as:

```
com.iplanet.am.lbcookie.name=server1  
com.iplanet.am.lbcookie.value=server1
```
  - b. Set the cookie name and value on `server2` as:

```
com.iplanet.am.lbcookie.name=server2  
com.iplanet.am.lbcookie.value=server2
```
6. Restart `server1` and `server2`.

## Confirming The Configuration

The following steps will confirm that the configuration is correct.

---

**CAUTION** Before starting these procedures, disable the `keepAliveTimeout` option in the Sun ONE Web Server web container.

---

1. Start the CDMaster by selecting Start under the Manage tab of the console.
2. Create several new users and login as those users.
3. Type `http://loadbalancer_host.domain:port/amconsole` into the Location bar of a web browser.
4. Login to Identity Server as `amadmin` and select the Current Sessions tab.

As `amadmin`, the created users and their corresponding servers will be visible. The users should all redirect back to the server on which their sessions were initiated. The web server access logs can also confirm this.



# Authenticate Against RADIUS Servers

Sun™ One Identity Server is able to authenticate users against a Remote Authentication Dial-In User Service (RADIUS) server. This appendix contains instructions to setup this deployment. It contains the following sections:

- Overview
- RADIUS Server Configuration
- Identity Server Configuration

## Overview

RADIUS is an industry standard protocol used to provide authentication and authorization services. In this type of authentication, Identity Server, the client, sends RADIUS-formatted messages to a RADIUS server which authenticates and authorizes the request and sends back a RADIUS-formatted response.

## RADIUS Server Configuration

The following procedures will allow an administrator to test Identity Server authentication against a RADIUS server.

1. Add a user entry to the RADIUS server which will be used to test authentication.

The following user information should be added to `RADIUS_install/etc/raddb/users` where *Login-Host* is the host and domain of the machine where Identity Server is running.

**Code Example 5-35** RADIUS User Entry

```
"Sample_User1" Password == "Password"
User-Service-Type = Login-User,
Login-Host = identity_server_host.domain_name,
Login-Service = PortMaster
```

2. Add the Identity Server Fully Qualified Domain Name (FQDN) or IP address to the RADIUS server.

This client information is added to *RADIUS\_install/etc/raddb/clients*. Ensure that the defined shared 'secret' is also added.

**Code Example 5-36** RADIUS Client Entry

```
191.18.18.111          <secret>
ms.red.example.com    <secret>
```

3. Change to the *RADIUS\_install/sbin* directory and restart the RADIUS server using the command:

```
./radiusd &
```

## Identity Server Configuration

1. Login to Identity Server as `amAdmin`.
2. Go to the top-level organization.
3. Select Services from the View drop down in the Navigation frame.
4. If RADIUS is not a registered authentication service, then click `Register . . .`.  
If RADIUS is already registered, go to Step 6.
5. Select "RADIUS" from the Data frame and click `Register`.
6. Click on the RADIUS properties arrow in the Navigation frame.  
If the template is not created, create it.

7. Add the FQDN or IP address of the RADIUS Server in the RADIUS Server 1 field.
8. Enter the shared secret used in Step 2 of “RADIUS Server Configuration” on page 145.
9. Enter the RADIUS server’s port number and save the template’s changes.  
The default is 1645.
10. Click on the Core properties arrow in the Navigation frame.
11. Select RADIUS in the Organization Authentication Modules list and save the change.

---

**CAUTION** In Step 11, be sure not to deselect LDAP when selecting RADIUS.

---

12. Logout from the Identity Server console.
13. Login as `Sample_User1` with the URL  
`http://identity_server_host.domain_name:port/service_deploy_uri/UI/Login?module=RADIUS`.



# Installing in a chroot Environment

This appendix provides instructions for installing and running Sun™ ONE Identity Server in a chroot environment. This appendix contains the following sections:

- Overview
- Before Creating chroot
- Creating a chroot Environment
- Installing Identity Server Under chroot
- Starting Identity Server In chroot
- Identity Server Log Files In chroot

## Overview

Chroot comes from the `change root` function in Unix®. When Identity Server is installed in a chroot environment, a named directory becomes the root directory. The new chroot directory then becomes the starting point in the path for all Identity Server searches. The chroot prevents malicious programs from accessing the real root file system, providing an added measure of security for the Web Server and Directory Server processes that power Identity Server.

## Before Creating chroot

Before beginning the chroot creation process, the following points should be noted:

- You must have root privileges in UNIX or Administrator privileges in Windows® to install Identity Server in a chroot environment.

- Patches cannot be automatically installed in the `chroot` environment. If any patches have been installed to your system, they must be manually copied to the appropriate locations:
  - Sun ONE Directory Server
  - Sun ONE Web Server
  - Sun ONE Identity Server
  - Solaris

## Creating a chroot Environment

Identity Server processes that serve requests cannot see or access files outside the `chroot` directory so the appropriate Identity Server files must be copied into the `chroot` directory structure. The first step is to create the *user root* environment for Identity Server. This new root must contain everything that Identity Server expects from the natural root (/).

---

**NOTE**      `$CHROOT` is the variable name for the `chroot` directory.

---

1. Create the following system directories on the computer system that will host Identity Server.

```
/dev
/etc
/sbin
/usr
/var
/proc
/opt
/etc/lib
/usr/platform
/usr/bin
/usr/sbin
/usr/lib
```

```
/usr/openwin/lib
```

```
/var/opt
```

```
/var/tmp
```

```
/usr/share
```

```
/usr/share/lib
```

**2.** Create system soft links using the command `ln -s`.

The following soft links should be created:

- o `/usr/bin -> bin`
- o `/usr/lib -> lib`
- o `/var/tmp -> tmp`

**3.** Create devices under `$CHROOT` using the command `mknod`.

---

**TIP** For example, to create the device `/dev/tcp` under the `$CHROOT` directory, enter:

```
mknod dev/tcp c 11 42
```

---

The following devices should be created under the `$CHROOT` directory:

- o `/dev/null`
  - o `/dev/tcp`
  - o `/dev/ticots`
  - o `/dev/ticlts`
  - o `/dev/ticotsord`
  - o `/dev/tty`
  - o `/dev/udp`
  - o `/dev/zero`
  - o `/dev/conslog`
- 4.** Copy files from directory `/etc` to corresponding locations under the `$CHROOT` directory. (For example, copy `/etc/hosts` to `$CHROOT/etc/hosts`.)

The following files should be copied:

- o `/etc/hosts`

- /etc/nsswitch.conf
  - /etc/vfstab
  - /etc/group
  - /etc/passwd
  - /etc/shadow
  - /etc/hosts
  - /etc/resolv.conf
  - /etc/nsswitch.conf
5. Copy binaries to the corresponding locations under the `$CHROOT` directory.  
(For example, copy `/usr/bin/sh` to `$CHROOT/usr/bin/sh`.)

The following files should be copied:

- /usr/bin/sh
- /usr/bin/ls
- /usr/bin/cat
- /usr/bin/date
- /usr/bin/dirname
- /usr/bin/mv
- /usr/bin/expr
- /usr/bin/file
- /usr/bin/awk
- /usr/bin/grep
- /usr/bin/gettext
- /usr/bin/echo
- /usr/bin/rm
- /usr/bin/sed
- /usr/bin/sleep
- /usr/bin/hostname
- /usr/bin/domainname



- /usr/bin/cut
  - /usr/bin/uname
  - /usr/bin/ksh
  - /usr/bin/basename
  - /usr/bin/id
  - /usr/bin/chmod
  - /usr/bin/nohup
  - /usr/bin/pkginfo
  - /usr/bin/su
  - /usr/bin/chown
  - /usr/bin/ftp
  - /usr/bin/isainfo
  - /usr/bin/ldd
  - /usr/bin/truss
  - /usr/bin/rm
  - /usr/xpg4/bin/id
6. Copy libraries to corresponding locations under the `$CHROOT` directory. (For example, copy `/usr/lib/libc.so*` to `$CHROOT/usr/lib/.`)

---

**NOTE** If the computer system in which Identity Server is installed is a 64-bit machine, the libraries from `/usr/lib/64` and `/usr/lib/lwp/64` need to be copied to `$CHROOT/usr/lib/64` and `$CHROOT/usr/lib/lwp/64`, respectively.

---

The following files should be copied:

- /usr/lib/libc.so\*
- /usr/lib/ld.so\*
- /usr/lib/libdl.so\*
- /usr/lib/libintl.so\*
- /usr/lib/libnsl.so\*

- /usr/lib/libsocket.so\*
- /usr/lib/librpcsvc.so\*
- /usr/lib/libw.so\*
- /usr/lib/libproject\*
- /usr/lib/libproc\*
- /usr/lib/libsecdb\*
- /usr/lib/libcmd\*
- /usr/lib/libmp.so\*
- /usr/lib/libm.so\*
- /usr/lib/libresolv.so\*
- /usr/lib/nss\_dns.so\*
- /usr/lib/nss\_files.so\*
- /usr/lib/nss\_nis.so\*
- /usr/lib/nss\_nisplus.so\*
- /usr/lib/straddr.so\*
- /usr/lib/libthread.so\*
- /usr/lib/libposix4.so\*
- /usr/lib/libC.so\*
- /usr/lib/librt.so\*
- /usr/lib/libaio.so\*
- /usr/lib/libelf.so\*
- /usr/lib/libgen.so\*
- /usr/lib/libpthread.so\*
- /usr/lib/libadm.so\*
- /usr/lib/libX\*
- /usr/lib/libCrun.so\*
- /usr/lib/libatomic.so\*

- /usr/lib/libdhcagent.so\*
  - /usr/lib/libproject.so\*
  - /usr/lib/libpam.so\*
  - /usr/lib/libbsm.so\*
  - /usr/lib/libldap.so\*
  - /usr/lib/libldap.so\*
  - /usr/lib/libdoor.so\*
  - /usr/lib/libz.so\*
  - /usr/lib/libkstat\*
  - /usr/lib/libld.so\*
  - /usr/lib/liblddbg.so\*
  - /usr/lib/librtld.so\*
  - /usr/lib/libsh.so\*
  - /usr/lib/lddstub\*
  - /usr/lib/librtld\_db.so\*
  - /usr/lib/libmd5.so\*
  - /usr/lib/locale
  - /usr/lib/libCstd.so\*
7. Copy the zoneinfo files to corresponding locations under the `$CHROOT` directory. (For example, copy `/usr/share/lib/zoneinfo/*` to `$CHROOT/usr/share/lib/.`)
  8. Copy the locale files to corresponding locations under the `$CHROOT` directory. (For example, copy `/usr/lib/locale/*` to `$CHROOT/usr/lib/locale.`)

## Installing Identity Server Under chroot

1. Install Identity Server 6.1 in a directory other than `$CHROOT`.

If an installation directory is not specified, Identity Server is installed, by default, in `/opt/IdentityServer_base`.

2. Stop the instance of Sun ONE Web Server.  
`./WebServer_base/https-instanceName/https-stop`
3. Stop the instance of Sun ONE Directory Server.  
`./DirectoryServer_base/slapd-instanceName/stop-slapd`
4. Copy the entire Identity Server directly from *IdentityServer\_base* to the *\$CHROOT* directory using the same structure.
5. Copy files from the following locations to the corresponding directories under *\$CHROOT*:
  - o `/etc/opt/SUNWam`
  - o `/var/opt/SUNWam`
  - o `/etc/init.d/amserver`

6. Create a loopback mount for Java.

The following command makes Identity Server ready to run under chroot environment:

```
mount -F lofs /usr/j2se $CHROOT/usr/j2se
```

7. Create a loopback mount for `/usr/share/lib` and `/usr/lib/mps`.

Examples:

```
mount -F lofs /usr/share/lib $CHROOT/usr/share/lib
```

```
mount -F lofs /usr/lib/mps $CHROOT/usr/lib/mps
```

## Starting Identity Server In chroot

To start Identity Server in the `chroot` environment, both Directory Server and Web Server must be started.

1. Start Directory Server using the `chroot` command:

```
chroot $ROOT $IS_ROOT/DSServers/slapd-hostName/start-slapd
```

2. Start Web Server using the `chroot` command:

```
chroot $ROOT $IS_ROOT/SUNWam/servers/https-instanceName/https-start
```

# Identity Server Log Files In chroot

Identity Server normally creates log files (Error Log, Debug, etc.) in the following directory:

`/var/opt/SUNWam/debug` and `/var/opt/SUNWam/logs`

Under chroot these log files will be created at the following locations:

`$CHROOT/var/opt/SUNWam/log`

`$CHROOT/var/opt/SUNWam/debug`

Identity Server Log Files In chroot

# Index

## A

- Active Directory
  - authenticate 127
- administrative roles 72
- architecture
  - authentication and user sessions 54
  - CDSSO, SAML and Federation 58
  - functional processes 54
  - Identity Server SDK 52
  - integrated client detection 58
  - integrated policy 57
  - integration points 50
  - overview 49
  - policy agents 51
- Authentication
  - RADIUS 145
- authentication
  - Active Directory 127
  - RADIUS servers 145
- authentication and user sessions 54

## B

- base directory 89

## C

- CDSSO, SAML and Federation 58

- chroot environment 149
- Configuration
  - RADIUS Server 145

## D

- Deployment Guide
  - chapter overview 31
- deployment planning
  - build timelines 47
  - categorize data 44
  - define resources 33
  - evaluate applications 42
  - gather information 38
  - set goals 38
- deployment road map 30
- deployment scenarios
  - Directory Servers 81
  - Identity Servers 78
  - Java application 80
  - multiple JVM environment 81
  - web deployment 77
- Directory Server
  - multiple instance deployment 81
  - multiple instances 60
  - overview 145
- documentation
  - overview 16
  - terminology 18
  - typographic conventions 18
- ds\_remote\_schema.ldif 68

## F

- federation
  - implementing 88
- functional processes 54
  - authentication and user sessions 54
  - CDSSO, SAML and Federation 58
  - integrated client detection 58
  - integrated policy 57

## H

- hardware requirements 63
- high availability 62

## I

- identity management
  - defined 21
  - identity profile 23
  - infrastructure 22
- identity profile 23
- Identity Server
  - access management overview 24
  - auditing overview 27
  - console overview 27
  - deploy 28
    - chapter overview 31
    - integrating 28
    - road map 30
  - extending 59
  - federation management overview 25
  - hardware requirements 63
  - high availability 62
  - identity management overview 26
  - multiple instance deployment 78
  - policy agents overview 27
  - product overview 24
  - programmatic interfaces overview 27
  - related product information 19
  - scalability 63
  - schema overview 67

- administrative roles 72
- limitations 74
- marker object classes 72
- SDK 52
- security 62
- software requirements 64
- technical considerations 61
- integrated client detection 58
- integrated policy 57
- integrating Identity Server 28
- integration points 50
  - Identity Server SDK 52
  - policy agents 51

## J

- Java application deployment 80
- JVM deployment 81

## L

- LDAP load balancers 60
- life cycle
  - user session 105
- load balancer replication 85
- load balancers 60

## M

- marker object classes 72

## O

- overview
  - access management 24
  - auditing 27
  - console 27



- Directory Server 145
- federation management 25
- identity management 26
- policy agents 27
- programmatically interfaces 27
- schema 67
  - administrative roles 72
  - limitations 74
  - marker object classes 72

## R

- RADIUS Server
  - Configuration 145
- RADIUS servers
  - authenticate 145
- replication 81
  - configuring for 82
  - with load balancer 85

## S

- scalability 63
- schema overview 67
  - administrative roles 72
  - limitations 74
  - marker object classes 72
- security 62
- session life cycle 105
- software requirements 64
- Solaris
  - patches 19
  - support 19
- sunone\_schema2.ldif 71
- SUNWam directory 90
- support
  - Solaris 19

## T

- technical considerations 61
  - high availability 62
  - scalability 63
  - security 62

## W

- web containers 59
- web deployment 77

