

Not for Publication

Sun Java System Federation Manager 7.5 Guide to Authentication

Alpha



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

Part No: 820-1697
December 2007

Copyright 2007 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 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 U.S. patents or pending patent applications in the U.S. and in other countries.

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 other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, the Solaris logo, the Java Coffee Cup logo, docs.sun.com, Java, and Solaris 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 an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and SunTM 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 publication are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical or 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 2007 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Tous droits réservés.

Sun Microsystems, Inc. détient les droits de propriété intellectuelle 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 plusieurs brevets américains ou des applications de brevet en attente aux Etats-Unis et dans d'autres pays.

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

Certains composants de ce produit peuvent être dérivées du logiciel Berkeley BSD, licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays; elle est licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, le logo Solaris, le logo Java Coffee Cup, docs.sun.com, Java et Solaris 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.

L'interface d'utilisation graphique OPEN LOOK et Sun 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 cette publication et les informations qu'il contient sont régis 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 chimiques ou biologiques ou pour le 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 régis 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 CONTREFACON.

List of Remarks

REMARK 1-1	Reviewer	Haven't heard if this module is in or out yet 4/19	31
REMARK 1-2	Reviewer	Added org in the order below. Please confirm that this is correct? Also please explain number 4.	39
REMARK 1-3	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-user-success-url attribute of the user's profile (amUser.xml).	40
REMARK 1-4	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-success-url attribute of the user's role entry.	40
REMARK 1-5	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-success-url attribute of the user's realm entry.	40
REMARK 1-6	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-success-url attribute as a global default.	40
REMARK 1-7	Reviewer	Old: A URL set in the iplanet-am-user-success-url attribute of the user's profile (amUser.xml).	40
REMARK 1-8	Reviewer	Old: A URL set in the iplanet-am-auth-login-success-url attribute of the user's role entry.	40
REMARK 1-9	Reviewer	Old: A URL set in the iplanet-am-auth-login-success-url attribute of the user's realm entry.	40
REMARK 1-10	Reviewer	Old: A URL set in the iplanet-am-auth-login-success-url attribute as a global default.	41
REMARK 1-11	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-user-failure-url attribute of the user's entry (amUser.xml).	41
REMARK 1-12	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-failure-url attribute of the user's role entry.	41
REMARK 1-13	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-failure-url attribute of the user's realm entry.	41
REMARK 1-14	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-failure-url attribute as a global default.	41
REMARK 1-15	Reviewer	Old: A URL set for the iplanet-am-user-failure-url attribute in the user's entry (amUser.xml).	41
REMARK 1-16	Reviewer	Old: A URL set for the iplanet-am-auth-login-failure-url attribute of the user's role entry.	41

List of Remarks

REMARK 1-17	Reviewer	Old: A URL set for the <code>iplanet-am-auth-login-failure-url</code> attribute of the user's realm entry. 41
REMARK 1-18	Reviewer	Old: A URL set for the <code>iplanet-am-auth-login-failure-url</code> attribute as the global default. 41
REMARK 1-19	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-user-success-url</code> attribute of the user's profile (<code>amUser.xml</code>). 43
REMARK 1-20	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-success-url</code> attribute of the role to which the user has authenticated. 43
REMARK 1-21	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-success-url</code> attribute of another role entry of the authenticated user. (This option is a fallback if the previous redirection URL fails.) 43
REMARK 1-22	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-success-url</code> attribute of the user's realm entry. 43
REMARK 1-23	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-success-url</code> attribute as a global default. 44
REMARK 1-24	Reviewer	Old: A URL set in the <code>iplanet-am-user-success-url</code> attribute of the user's profile (<code>amUser.xml</code>). 44
REMARK 1-25	Reviewer	Old: A URL set in the <code>iplanet-am-auth-login-success-url</code> attribute of the role to which the user has authenticated. 44
REMARK 1-26	Reviewer	Old: A URL set in the <code>iplanet-am-auth-login-success-url</code> attribute of another role entry of the authenticated user. (This option is a fallback if the previous redirection URL fails.) 44
REMARK 1-27	Reviewer	Old: A URL set in the <code>iplanet-am-auth-login-success-url</code> attribute of the user's realm entry. 44
REMARK 1-28	Reviewer	Old: A URL set in the <code>iplanet-am-auth-login-success-url</code> attribute as a global default. 44
REMARK 1-29	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-user-failure-url</code> attribute of the user's profile (<code>amUser.xml</code>). 44
REMARK 1-30	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-failure-url</code> attribute of the role to which the user has authenticated. 44
REMARK 1-31	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-failure-url</code> attribute of another role entry of the authenticated user. (This option is a fallback if the previous redirection URL fails.) 45
REMARK 1-32	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-failure-url</code> attribute of the user's realm entry. 45
REMARK 1-33	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-failure-url</code> attribute as a global default. 45
REMARK 1-34	Reviewer	Old: A URL set in the <code>iplanet-am-user-failure-url</code> attribute of the user's profile

		(amUser.xml).	45
REMARK 1-35	Reviewer	Old: A URL set in the iplanet-am-auth-login-failure-url attribute of the role to which the user has authenticated.	45
REMARK 1-36	Reviewer	Old: A URL set in the iplanet-am-auth-login-failure-url attribute of another role entry of the authenticated user. (This option is a fallback if the previous redirection URL fails.)	45
REMARK 1-37	Reviewer	Old: A URL set in the iplanet-am-auth-login-failure-url attribute of the user's realm entry.	45
REMARK 1-38	Reviewer	Old: A URL set in the iplanet-am-auth-login-failure-url attribute as a global default.	45
REMARK 1-39	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-user-success-url attribute of the user's profile (amUser.xml).	47
REMARK 1-40	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-success-url attribute of the service to which the user has authenticated.	47
REMARK 1-41	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-success-url attribute of the user's role entry.	47
REMARK 1-42	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-success-url attribute of the user's realm entry.	48
REMARK 1-43	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-success-url attribute as a global default.	48
REMARK 1-44	Reviewer	Old: A URL set in the iplanet-am-user-success-url attribute of the user's profile (amUser.xml).	48
REMARK 1-45	Reviewer	Old: A URL set in the iplanet-am-auth-login-success-url attribute of the service to which the user has authenticated.	48
REMARK 1-46	Reviewer	Old: A URL set in the iplanet-am-auth-login-success-url attribute of the user's role entry.	48
REMARK 1-47	Reviewer	Old: A URL set in the iplanet-am-auth-login-success-url attribute of the user's realm entry.	48
REMARK 1-48	Reviewer	Old: A URL set in the iplanet-am-auth-login-success-url attribute as a global default.	48
REMARK 1-49	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-user-failure-url attribute of the user's profile (amUser.xml).	48
REMARK 1-50	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-failure-url attribute of the service to which the user has authenticated.	48
REMARK 1-51	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-failure-url attribute of the user's role entry.	49
REMARK 1-52	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-failure-url attribute of the user's realm entry.	49

List of Remarks

REMARK 1-53	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-failure-url</code> attribute as a global default. 49
REMARK 1-54	Reviewer	Old: A URL set in the <code>iplanet-am-user-failure-url</code> attribute of the user's profile (<code>amUser.xml</code>). 49
REMARK 1-55	Reviewer	Old: A URL set in the <code>iplanet-am-auth-login-failure-url</code> attribute of the service to which the user has authenticated. 49
REMARK 1-56	Reviewer	Old: A URL set in the <code>iplanet-am-auth-login-failure-url</code> attribute of the user's role entry. 49
REMARK 1-57	Reviewer	Old: A URL set in the <code>iplanet-am-auth-login-failure-url</code> attribute of the user's realm entry. 49
REMARK 1-58	Reviewer	Old: A URL set in the <code>iplanet-am-auth-login-failure-url</code> attribute as a global default. 49
REMARK 1-59	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-user-success-url</code> attribute of the user's profile (<code>amUser.xml</code>). 51
REMARK 1-60	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-success-url</code> attribute of the user's role entry. 51
REMARK 1-61	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-success-url</code> attribute of the user's realm entry. 51
REMARK 1-62	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-success-url</code> attribute as a global default. 51
REMARK 1-63	Reviewer	Old: A URL set in the <code>iplanet-am-user-success-url</code> attribute of the user's profile (<code>amUser.xml</code>). 51
REMARK 1-64	Reviewer	Old: A URL set in the <code>iplanet-am-auth-login-success-url</code> attribute of the user's role entry. 51
REMARK 1-65	Reviewer	Old: A URL set in the <code>iplanet-am-auth-login-success-url</code> attribute of the user's realm entry. 51
REMARK 1-66	Reviewer	Old: A URL set in the <code>iplanet-am-auth-login-success-url</code> attribute as a global default. 52
REMARK 1-67	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-user-failure-url</code> attribute of the user's entry (<code>amUser.xml</code>). 52
REMARK 1-68	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-failure-url</code> attribute of the user's role entry. 52
REMARK 1-69	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-failure-url</code> attribute of the user's realm entry. 52
REMARK 1-70	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-failure-url</code> attribute as a global default. 52
REMARK 1-71	Reviewer	Old: A URL set for the <code>iplanet-am-user-failure-url</code> attribute in the user's entry (<code>amUser.xml</code>). 52
REMARK 1-72	Reviewer	Old: A URL set for the <code>iplanet-am-auth-login-failure-url</code> attribute of the user's role entry. 52

REMARK 1-73	Reviewer	Old: A URL set for the <code>iplanet-am-auth-login-failure-url</code> attribute of the user's realm entry. 52
REMARK 1-74	Reviewer	Old: A URL set for the <code>iplanet-am-auth-login-failure-url</code> attribute as the global default. 52
REMARK 1-75	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-user-success-url</code> attribute of the user's profile (<code>amUser.xml</code>). 55
REMARK 1-76	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-success-url</code> attribute of the user's role entry. 55
REMARK 1-77	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-success-url</code> attribute of the user's realm entry. 55
REMARK 1-78	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-success-url</code> attribute as a global default. 55
REMARK 1-79	Reviewer	Old: A URL set in the <code>iplanet-am-user-success-url</code> attribute of the user's profile (<code>amUser.xml</code>). 55
REMARK 1-80	Reviewer	Old: A URL set in the <code>iplanet-am-auth-login-success-url</code> attribute of the user's role entry. 55
REMARK 1-81	Reviewer	Old: A URL set in the <code>iplanet-am-auth-login-success-url</code> attribute of the user's realm entry. 55
REMARK 1-82	Reviewer	Old: A URL set in the <code>iplanet-am-auth-login-success-url</code> attribute as a global default. 55
REMARK 1-83	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-user-failure-url</code> attribute of the user's entry (<code>amUser.xml</code>). 55
REMARK 1-84	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-failure-url</code> attribute of the user's role entry. 56
REMARK 1-85	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-failure-url</code> attribute of the user's realm entry. 56
REMARK 1-86	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-failure-url</code> attribute as a global default. 56
REMARK 1-87	Reviewer	Old: A URL set for the <code>iplanet-am-user-failure-url</code> attribute in the user's entry (<code>amUser.xml</code>). 56
REMARK 1-88	Reviewer	Old: A URL set for the <code>iplanet-am-auth-login-failure-url</code> attribute of the user's role entry. 56
REMARK 1-89	Reviewer	Old: A URL set for the <code>iplanet-am-auth-login-failure-url</code> attribute of the user's realm entry. 56
REMARK 1-90	Reviewer	Old: A URL set for the <code>iplanet-am-auth-login-failure-url</code> attribute as the global default. 56
REMARK 1-91	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-user-success-url</code> attribute of the user's profile (<code>amUser.xml</code>). 57
REMARK 1-92	Reviewer	Old: A URL set in the <code>clientType</code> custom files for the <code>iplanet-am-auth-login-success-url</code> attribute of the user's role entry. 57

List of Remarks

REMARK 1-93	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-success-url attribute of the user's realm entry. 58
REMARK 1-94	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-success-url attribute as a global default. 58
REMARK 1-95	Reviewer	Old: A URL set in the iplanet-am-user-success-url attribute of the user's profile (amUser.xml). 58
REMARK 1-96	Reviewer	Old: A URL set in the iplanet-am-auth-login-success-url attribute of the user's role entry. 58
REMARK 1-97	Reviewer	Old: A URL set in the iplanet-am-auth-login-success-url attribute of the user's realm entry. 58
REMARK 1-98	Reviewer	Old: A URL set in the iplanet-am-auth-login-success-url attribute as a global default. 58
REMARK 1-99	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-user-failure-url attribute of the user's entry (amUser.xml). 58
REMARK 1-100	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-failure-url attribute of the user's role entry. 58
REMARK 1-101	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-failure-url attribute of the user's realm entry. 58
REMARK 1-102	Reviewer	Old: A URL set in the clientType custom files for the iplanet-am-auth-login-failure-url attribute as a global default. 59
REMARK 1-103	Reviewer	Old: A URL set for the iplanet-am-auth-login-failure-url attribute of the user's entry (amUser.xml). 59
REMARK 1-104	Reviewer	Old: A URL set for the iplanet-am-auth-login-failure-url attribute of the user's role entry. 59
REMARK 1-105	Reviewer	Old: A URL set for the iplanet-am-auth-login-failure-url attribute of the user's realm entry. 59
REMARK 1-106	Reviewer	Old: A URL set for the iplanet-am-auth-login-failure-url attribute as the global default. 59
REMARK 2-1	Reviewer	email to Chand Basha and dsame-auth 4/13/07; FM not yet localized so no directory structure 80
REMARK 2-2	Reviewer	Please review and add info as necessary. 93
REMARK 2-3	Reviewer	Please review and add info as necessary. 94

Contents

Preface	13
 1 Authentication Configuration and Initiation	19
Configuring the Authentication Service	19
General Authentication Properties	20
▼ To Modify the General Authentication Properties Globally	20
▼ To Modify the General Authentication Properties for a Realm	21
Authentication Configuration Service	21
▼ To Assign Authentication Configuration to a Role	22
Authentication Modules	22
Active Directory	23
▼ To Define Default Values for Active Directory Authentication Global Attributes	23
Anonymous	24
▼ To Define Default Values for Anonymous Authentication Global Attributes	24
Certificate	24
▼ To Define Default Values for Certificate Authentication Global Attributes	25
Data Store	26
▼ To Define Default Values for Certificate Authentication Global Attributes	26
Federation	26
▼ To Define Default Values for Federation Authentication Global Attributes	27
HTTP Basic	27
▼ To Define Default Values for HTTP Basic Authentication Global Attributes	27
JDBC	28
▼ To Define Default Values for JDBC Authentication Global Attributes	28
LDAP	29
▼ To Define Default Values for LDAP Authentication Global Attributes	29
Membership	29
▼ To Define Default Values for Membership Authentication Global Attributes	30

MSISDN	30
▼ To Define Default Values for MSISDN Authentication Global Attributes	30
UnixXXXXX	31
Windows Desktop SSO	31
▼ To Define Default Values for Windows Desktop SSO Authentication Global Attributes	32
▼ To Configure for Windows Desktop SSO Authentication	33
Windows NT	35
▼ To Define Default Values for Windows NT Authentication Global Attributes	35
▼ To Install the SAMBA Client on Solaris	36
▼ To Install the SAMBA Client on Linux	36
Types of Authentication	36
Realm-based Authentication	38
▼ To Configure Realm-based Authentication	42
Role-based Authentication	42
▼ To Configure Role-based Authentication	46
Service-based Authentication	46
▼ To Configure Service-based Authentication	49
User-based Authentication	50
▼ To Configure User-Based Authentication	53
Authentication Level-based Authentication	53
▼ To Configure for Authentication Level-based Authentication	56
Module-based Authentication	56
▼ To Configure for Module-based Authentication	59
Configuring the Authentication Processes	59
Instantiating an Authentication Module	60
▼ To Create an Authentication Module Instance	60
Creating an Authentication Chain	60
▼ To Create an Authentication Chain	61
Initiating Authentication	62
Web-based Clients	62
Programmatic Clients	63
 2 The Authentication Service User Interface	 65
The Authentication Service User Interface	65

XML Files	66
JavaServer Pages	68
JavaScript Files	70
Cascading Style Sheets	70
Images	70
Localization Properties Files	71
Accessing the Authentication Service User Interface	72
The Authentication Service User Interface Login URL	73
Redirection Parameters	74
Organization Parameters	75
Authentication Method Parameters	77
Login Function Parameters	78
Customizing the Authentication Service User Interface	81
Modifying Login Screens	82
Generating Login Screens with JavaServer Pages	86
Adding Functionality	90
Branding the Authentication Service User Interface	90
Directories for Custom Files	91
3 Distributed Authentication	95
The Distributed Authentication Process	95
Installing the Distributed Authentication User Interface	98
▼ To Install the Distributed Authentication User Interface	98
Configuring the Distributed Authentication User Interface	99
▼ To Reconfigure the Distributed Authentication User Interface	102
Customizing the Distributed Authentication User Interface XXXXX	102
▼ To Customize the Distributed Authentication User Interface XXXXX	102
Accessing the Distributed Authentication User Interface	104
Using a Policy Agent	104
Using a Web Browser	104
4 Inside the Authentication Service	107
Client Detection	107
Client Detection Process	107
Client Data	108

A Forced Reauthentication111

Index 113

Preface

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

Who Should Use This Book

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

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

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

Before You Read This Book

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

- Because Federation Manager can be configured to work with Access Manager, you should be familiar with the [Sun Java System Access Manager documentation](#). Specifically, the *Sun Java System Access Manager 7.1 Federation and SAML Administration Guide* might be helpful if you have not yet worked with a Liberty-based deployment.
- Because Sun Java System Directory Server can be used as a data store in a Federation Manager deployment, you should be familiar with the [Sun Java System Directory Server 5 2005Q4 documentation set](#).
- Because Federation Manager contains features based on the Liberty Alliance Project specifications, you should be familiar with the [Liberty Alliance Project specifications](#).
- Because Federation Manager can work in tandem with many of the products developed as part of the Sun Java Enterprise System suite, you should be familiar with the [Sun Java Enterprise System documentation](#).

How This Book Is Organized

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

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

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

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

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

Related Books

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

- [“Federation Manager Core Documentation” on page 15](#)
- [“Sun Java System Product Documentation” on page 16](#)

Federation Manager Core Documentation

The Federation Manager documentation set contains the following titles:

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

Note – The *Sun Java System Access Manager 7.1 Federation and SAML Administration Guide* provides information about the features in Access Manager that are based on the Liberty Alliance Project and SAML specifications. Because Access Manager is also based on these specifications and might be used with Federation Manager, this guide provides useful information for a Federation Manager deployment.

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

Sun Java System Product Documentation

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

- [Sun Java System Access Manager](#)
- [Sun Java System Directory Server](#)
- [Sun Java System Web Server](#)
- [Sun Java System Application Server](#)
- [Sun Java System Web Proxy Server](#)

Contacting Sun Technical Support

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

Related Third-Party Web Site References

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

Sun Welcomes Your Feedback

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

Documentation, Support, and Training

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

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

Typographic Conventions

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

TABLE P-2 Typographic Conventions

Typeface	Meaning	Example
AaBbCc123	The names of commands, files, and directories, and onscreen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. <code>machine_name%</code> you have mail.
AaBbCc123	What you type, contrasted with onscreen computer output	<code>machine_name% su</code> Password:
<i>aabbcc123</i>	Placeholder: replace with a real name or value	The command to remove a file is <i>rm filename</i> .
<i>AaBbCc123</i>	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the <i>User's Guide</i> . A <i>cache</i> is a copy that is stored locally. Do <i>not</i> save the file. Note: Some emphasized items appear bold online.

Shell Prompts in Command Examples

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

TABLE P-3 Shell Prompts

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

Additional Sun Resources

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

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

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

◆ ◆ ◆ 1 CHAPTER 1

Authentication Configuration and Initiation

There are a multitude of authentication options in Sun Java™ System Federation Manager. Configuration of the authentication process is dependent on defining one or more instances of an authentication module, configuring an authentication chain, or both. Initiating these authentication configurations is accomplished when either a web-based or programmatic client attempts access to a protected resource. The following sections contain information regarding these processes.

- [“Configuring the Authentication Service” on page 19](#)
- [“Authentication Modules” on page 22](#)
- [“Types of Authentication” on page 36](#)
- [“Configuring the Authentication Processes” on page 59](#)
- [“Initiating Authentication” on page 62](#)

Configuring the Authentication Service

Before configuring the authentication processes that will be available in your deployment, you first configure the Federation Manager Authentication Service. The Authentication Service attributes can be configured at a number of levels.

1. Default values for Authentication Service attributes are defined in the `amAuth.xml` file and stored in the Federation Manager information tree. These global values are used if no overriding value is defined in the configured realm or a specified authentication module.
2. More specific configurations are made at the realm level for the realm's users, roles, services, and so forth.
3. Fine-grained configurations are also made at the authentication module level. When the particular module is enabled for a configured realm or authentication chain, the values defined in the module are used for authentication.

The following sections contain information regarding global and realm-level configuration.

- [“General Authentication Properties” on page 20](#)

- [“Authentication Configuration Service” on page 21](#)

See [“Authentication Modules” on page 22](#) for information regarding module-level configurations.

General Authentication Properties

The General authentication properties (also referred to as the Core Authentication module) can be defined globally for the Federation Manager deployment or more specifically for each configured realm. These properties and the values defined globally are added to each new realm when it is created. Once added to a realm new values can be defined and configured values can be modified by the realm's administrator. The values are then used if no overriding value is defined in the specified authentication module instance or authentication chain.

Note – The General authentication properties are added and enabled for the top-level realm during installation. As all new realms are configured under the top-level realm, these properties are dynamically added and enabled for any new realm when created.

The following procedures illustrate how to modify the General authentication properties.

- [“To Modify the General Authentication Properties Globally” on page 20](#)
- [“To Modify the General Authentication Properties for a Realm” on page 21](#)

▼ To Modify the General Authentication Properties Globally

Global attributes are applied across the Federation Manager deployment to customize how a service (in this case, the Authentication Service) works.

- 1 Login as the administrator of the top-level realm; by default, `amadmin`.**
- 2 Select the Configuration tab.**
- 3 Click Core in the Service Name list under the Authentication heading.**
- 4 Modify the global attributes by adding or changing the values of the Global Attributes.**
Click Help for attribute descriptions.
- 5 (Optional) Modify the attributes specific to the top-level realm by adding or changing the values of the Realm Attributes.**

These attributes values are specific to the top-level realm only. These modifications can also be made by navigating to the General Authentication properties under the top-level realm; `opensso` by default. See [“To Modify the General Authentication Properties for a Realm” on page 21](#).

- 6 Click Save.
- 7 Click Back to Configuration.
- 8 Logout.

▼ To Modify the General Authentication Properties for a Realm

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

- 1 Login as the administrator of the realm you are configuring.
- 2 Select the Access Control tab.
- 3 Click the name of the realm which you are modifying.
- 4 Select the Authentication tab.
- 5 Click Advanced Properties under the General heading.
- 6 Modify the Realm Attributes by adding or changing their values.
Click Help for attribute descriptions.
- 7 Click Save.
- 8 Click Back to Authentication.
- 9 Logout.

Authentication Configuration Service

The Authentication Configuration properties allow an administrator to define an authentication chain for a specific role in a configured realm. When you assign Authentication Configuration to a role, you can then specify a particular authentication chain. For more information, see [“Role-based Authentication” on page 42](#).

▼ To Assign Authentication Configuration to a Role

- 1 Login as the administrator of the realm you will be configuring.
- 2 Select the Access Control tab.
- 3 Click the name of the realm which you are modifying.
- 4 Select the Subjects tab.
- 5 Select the Role tab.
- 6 Click the name of the role you are configuring.
- 7 Select the Services tab.
- 8 Click Add under the Services heading.
- 9 Click the Authentication Configuration radio button.
- 10 Click Next.
- 11 Click the radio button next to the appropriate authentication chain.
You may click Empty and define the authentication chain later.
- 12 Click Finish.
- 13 Click Back to Subjects.
- 14 Logout.

Authentication Modules

An *authentication module* is a plug-in that collects information from a user requesting access to a resource (such as an identifier and a password) and checks the information against entries in a database. If a user provides information that meets the module's authentication criteria, the user is granted access to the resource. If the user provides information that does not meet the module's authentication criteria, the user is denied access to the resource. Federation Manager provides a number of modules that can be instantiated within a realm and used for different types of authentication. You can configure multiple instances of an authentication module. For example, you can configure two instances of the LDAP Authentication module, each pointing to a different LDAP database, in the same realm.

Before instantiating an authentication module for a realm, you can configure default values for the module globally. These values will then be carried over to the created instance. Once instantiated, you can modify the values as you see fit. The following sections contain details about the included authentication modules and provides the procedure you use to configure the module's global attributes.

- [“Active Directory” on page 23](#)
- [“Anonymous” on page 24](#)
- [“Certificate” on page 24](#)
- [“Data Store” on page 26](#)
- [“Federation” on page 26](#)
- [“HTTP Basic” on page 27](#)
- [“JDBC” on page 28](#)
- [“LDAP” on page 29](#)
- [“Membership” on page 29](#)
- [“MSISDN” on page 30](#)
- [“UnixXXXXX” on page 31](#)
- [“Windows Desktop SSO” on page 31](#)
- [“Windows NT” on page 35](#)

Active Directory

The Active Directory® authentication module performs authentication in a manner similar to the LDAP authentication module, but uses the Microsoft® Active Directory server as opposed to Sun Java™ System Directory Server. Although the LDAP authentication module can be configured to communicate with an Active Directory server, the Active Directory module allows both LDAP and Active Directory authentication to coexist under the same realm.

▼ To Define Default Values for Active Directory Authentication Global Attributes

- 1 **Login as the administrator of the top-level realm; by default, `amadmin`.**
- 2 **Select the Configuration tab.**
- 3 **Click Authentication.**
- 4 **Click Active Directory under the Authentication heading.**
- 5 **Configure the global attributes by adding or changing the values.**
Click Help for attribute descriptions.
- 6 **Click Save.**

- 7 Click Back to Configuration.
- 8 Logout.

Anonymous

With the Anonymous authentication module, a user can log in to Federation Manager anonymously. Granting anonymous access means that the resource can be accessed without providing a password. Anonymous access can be limited to specific types of access (for example, access for read or access for search) or to specific sub trees or individual entries within the directory.

Note – The Anonymous authentication module is enabled by default.

▼ To Define Default Values for Anonymous Authentication Global Attributes

- 1 Login as the administrator of the top-level realm; by default, `amadmin`.
- 2 Select the Configuration tab.
- 3 Click Authentication.
- 4 Click Anonymous under the Authentication heading.
- 5 Configure the global attributes by adding or changing the values.
A list of anonymous users can be defined by configuring the Valid Anonymous Users attribute. Click Help for descriptions of the other attributes.
- 6 Click Save.
- 7 Click Back to Configuration.
- 8 Logout.

Certificate

Certificate authentication involves using a personal digital certificate (PDC) to identify and authenticate a user. A PDC consists of digital data that specifies the name of an individual, company, or other entity, and certifies that the public key, included in the certificate, belongs to

that entity. (SSL-enabled servers must have a certificate and clients may optionally have a certificate.) The user is granted or denied access to a resource based on whether or not the certificate is valid. The Certificate authentication module can be configured to require a match against a PDC stored using LDAP as well as verification against a Certificate Revocation List. Each user that will authenticate using the Certificate authentication module must request a PDC for the browser being used; instructions are different depending upon the browser. See your browser's documentation for more information.

Before adding and enabling the Certificate authentication module to a realm, Federation Manager and the web container on which it is deployed must be configured for SSL and client authentication must be enabled. For more information, see XXXXXX Configuring Access Manager in SSL Mode in the Access Manager Post Installation Guide and your web container's documentation. Then, log in to Federation Manager as the administrator of the realm to add and enable the Certificate authentication module.

Note – If you are configuring the Certificate authentication module for use on an instance of Federation Manager deployed on an SSL-enabled instance of Sun Java System Web Server 6.1, and want the Web Server to accept both certificate-based and non certificate-based authentication requests, set the following value in the `obj.conf` file of the Web Server:

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

XXXXXXBefore enabling the Certificate-based module, see Chapter 6, Certificates and Keys in the Sun ONE Web Server 6.1 Administrator's Guide for these initial Web Server configuration steps. This document can be found at <http://docs.sun.com/db/prod/s1websrv#hic> Or, see the Sun ONE Application Server Administrator's Guide to Security at the following location: <http://docs.sun.com/db/prod/s1appsrv#hic>

▼ To Define Default Values for Certificate Authentication Global Attributes

- 1 Login as the administrator of the top-level realm; by default, `amadmin`.
- 2 Select the Configuration tab.
- 3 Click Authentication.
- 4 Click Certificate under the Authentication heading.
- 5 Configure the global attributes by adding or changing the values.
Click Help for attribute descriptions.
- 6 Click Save.

- 7 Click Back to Configuration.
- 8 Logout.

Data Store

The Data Store authentication module allows login using the identity repository configured as the realm's data store. Using the Data Store module removes the requirement to write an authentication plug-in. For example, if a realm's configured identity repository is an LDAPv3 data store and the realm is configured for Data Store authentication, any user defined in the LDAPv3 data store can authenticate to that realm with no additional LDAP configuration necessary.

▼ To Define Default Values for Certificate Authentication Global Attributes

- 1 Login as the administrator of the top-level realm; by default, `amadmin`.
- 2 Select the Configuration tab.
- 3 Click Authentication.
- 4 Click Data Store under the Authentication heading.
- 5 Configure the global attributes by adding or changing the values.
Click Help for attribute descriptions.
- 6 Click Save.
- 7 Click Back to Configuration.
- 8 Logout.

Federation

XXXXX

▼ To Define Default Values for Federation Authentication Global Attributes

- 1 Login as the administrator of the top-level realm; by default, `amadmin`.
- 2 Select the Configuration tab.
- 3 Click Authentication.
- 4 Click Federation under the Authentication heading.
- 5 Configure the global attributes by adding or changing the values.
Click Help for attribute descriptions.
- 6 Click Save.
- 7 Click Back to Configuration.
- 8 Logout.

HTTP Basic

The HTTP Basic authentication module uses the basic authentication feature built into the Hypertext Transport Protocol (HTTP). (There is no data encryption.) The web server issues a client request for a username and password, and returns the information to Federation Manager as part of the authorized request. Federation Manager then authenticates the user with the LDAP authentication module. Once successfully authenticated, the user will be able to reauthenticate without being prompted for a username and password.



Caution – In order for HTTP Basic to function correctly, the LDAP authentication module must be added and enabled.

▼ To Define Default Values for HTTP Basic Authentication Global Attributes

- 1 Login as the administrator of the top-level realm; by default, `amadmin`.
- 2 Select the Configuration tab.
- 3 Click Authentication.

- 4 Click HTTP Basic under the Authentication heading.
- 5 Configure the global attributes by adding or changing the values.
Click Help for attribute descriptions.
- 6 Click Save.
- 7 Click Back to Configuration.
- 8 Logout.

JDBC

The Java Database Connectivity (JDBC) authentication module provides a mechanism to allow Federation Manager to authenticate users against any SQL database that provides JDBC technology-enabled drivers. The connection to the SQL database can be either directly through a JDBC driver, or a Java Naming and Directory Interface (JNDI) connection pool.

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

▼ To Define Default Values for JDBC Authentication Global Attributes

- 1 Login as the administrator of the top-level realm; by default, `amadmin`.
- 2 Select the Configuration tab.
- 3 Click Authentication.
- 4 Click JDBC under the Authentication heading.
- 5 Configure the global attributes by adding or changing the values.
Click Help for attribute descriptions.
- 6 Click Save.
- 7 Click Back to Configuration.
- 8 Logout.

LDAP

With the LDAP authentication module, the user is required to bind to an LDAP directory server with a specific user distinguished name (DN) and password. This is the default authentication module for all realm-based authentication. If the user provides a user ID and password that are in the directory server, the user is set up with a valid Federation Manager session.

Note – The LDAP authentication module is added and enabled for the top-level realm during installation. As all new realms are configured under the top-level realm, LDAP authentication is dynamically added and enabled for all sub realms when created.

▼ To Define Default Values for LDAP Authentication Global Attributes

- 1 Login as the administrator of the top-level realm; by default, `amadmin`.
- 2 Select the Configuration tab.
- 3 Click Authentication.
- 4 Click LDAP under the Authentication heading.
- 5 Configure the global attributes by adding or changing the values.
Click Help for attribute descriptions.
- 6 Click Save.
- 7 Click Back to Configuration.
- 8 Logout.

Membership

The Membership authentication module implements authentication similar to the way in which personalized sites do: a user creates an account, personalizes it, and then uses it to access the site. XXXXXThe user can also access the viewer interface, saved on the user profile database as authorization data and user preferences.

▼ **To Define Default Values for Membership Authentication Global Attributes**

- 1 Login as the administrator of the top-level realm; by default, `amadmin`.
- 2 Select the Configuration tab.
- 3 Click Authentication.
- 4 Click Membership under the Authentication heading.
- 5 Configure the global attributes by adding or changing the values.
Click Help for attribute descriptions.
- 6 Click Save.
- 7 Click Back to Configuration.
- 8 Logout.

MSISDN

The Mobile Station Integrated Services Digital Network (MSISDN) authentication module enables authentication using the mobile subscriber ISDN associated with a particular device such as a cellular telephone. It is a non-interactive module that retrieves the requesting device's ISDN and validates it against a database to find the user that matches it.

▼ **To Define Default Values for MSISDN Authentication Global Attributes**

- 1 Login as the administrator of the top-level realm; by default, `amadmin`.
- 2 Select the Configuration tab.
- 3 Click Authentication.
- 4 Click MSISDN under the Authentication heading.
- 5 Configure the global attributes by adding or changing the values.
Click Help for attribute descriptions.
- 6 Click Save.

7 Click Back to Configuration.

8 Logout.

UnixXXXXX

Remark 1–1 Haven't heard if this module is in or out yet 4/19
Reviewer

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

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

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

```
./amunxd
```

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

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

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

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

Windows Desktop SSO

The Windows Desktop SSO authentication module is a Kerberos-based authentication plug-in module specific to Windows 2000. It allows a user who has already authenticated with a Kerberos Distribution Center (KDC) to authenticate to Federation Manager without

resubmitting the login criteria (essentially, single sign-on). The user presents the Kerberos token to Federation Manager through the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) protocol. The client must support the SPNEGO protocol to authenticate itself. In general, any client that supports this protocol should be able to use this module to authenticate to Federation Manager. Depending on the availability of the token on the client side, this module provides either a SPNEGO token or a Kerberos token (in both cases, the protocols are the same). Microsoft Internet Explorer (5.01 or later) running on Windows 2000 (or later) supports this protocol. Additionally, the token returned when using Mozilla 1.4 on Solaris 9 and 10 is only a KERBEROS token as SPNEGO is not supported on Solaris.

Tip – You must use the Java Development Kit (JDK) 1.4 or above to use the new features of Kerberos V5 authentication module, and Java GSS API to perform Kerberos based SSO in this SPNEGO module.

The following procedures are related to Windows Desktop SSO authentication.

- [“To Define Default Values for Windows Desktop SSO Authentication Global Attributes” on page 32](#)
- [“To Configure for Windows Desktop SSO Authentication” on page 33](#)

▼ **To Define Default Values for Windows Desktop SSO Authentication Global Attributes**

- 1 Login as the administrator of the top-level realm; by default, `amadmin`.**
- 2 Select the Configuration tab.**
- 3 Click Authentication.**
- 4 Click Windows Desktop SSO under the Authentication heading.**
- 5 Configure the global attributes by adding or changing the values.**

Click Help for attribute descriptions. Additionally, the values would be similar in format to the following:

Service Principal	HTTP/machine1.EXAMPLE.COM@ISQA.EXAMPLE.COM
Keytab File Name	/tmp/machine1.HTTP.keytab
Kerberos Realm	ISQA.EXAMPLE.COM
Kerberos Server Name	machine2.EXAMPLE.com
Return Principal with Domain Name	false
Authentication Level	22

- 6 Click Save.
- 7 Click Back to Configuration.
- 8 Logout.

▼ To Configure for Windows Desktop SSO Authentication

Before You Begin This procedure includes additional sub procedures. Please read all instructions first.

- 1 **Create a user account for the Federation Manager authentication module in the Windows 2000 Domain Controller using the following sub procedure.**

The following procedure applies to Microsoft Internet Explorer 6 and later. If you are using an earlier version, make sure that Federation Manager is in the browser's internet zone and enable Native Windows Authentication.

- a. **From the Start menu, go to Programs > Administration Tools.**
- b. **Select Active Directory Users and Computers.**
- c. **Create a new user using the Federation Manager host name as the User ID (login name).**

The Federation Manager host name should not include the domain name.

- 2 **Run the following command to associate the user account with a service provider name and export the keytab files to the system in which Federation Manager is deployed.**

Make sure that both keytab files are kept secure.

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

The ktpass command accepts the following parameters:

<i>hostname</i>	The host name (without the domain) on which Federation Manager is running.
<i>domainname</i>	The Federation Manager domain name.
<i>DCDOMAIN</i>	The domain name of the domain controller. This may be different from the Federation Manager domain name.
<i>password</i>	The password of the user account. Make sure that password is correct, as ktpass does not verify passwords.
<i>userName</i>	The user account ID. This should be the same as hostname.

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

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

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

as in

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

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

3 Restart the server.

4 Setup Internet Explorer using the following sub procedure.

The following procedure applies to Microsoft Internet Explorer 6 and later. If you are using an earlier version, make sure that Federation Manager is in the browser's internet zone and enable Native Windows Authentication.

- a. In the Tool menu, go to Internet Options > Advanced/Security > Security.**
- b. Select the Integrated Windows Authentication option.**
- c. Go to Security > Local Internet.**
- d. Select Custom Level.**
- e. Select the Automatic Logon Only in Intranet Zone option in the User Authentication/Logon panel.**
- f. Go to Sites and select all of the options.**
- g. Click Advanced and add Federation Manager to the local zone (if not already there).**

Windows NT

The Windows NT Authentication module allows for authentication against a Microsoft Windows NT server. Federation Manager provides the client portion of Windows NT authentication. Windows NT authentication can be configured using the following procedure.

1. Configure the NT server.

For detailed instructions, see the Windows NT server documentation.

2. Obtain and install Samba Client 2.2.2 on your system to communicate with Federation Manager.

The Samba client is software that allows the host on which it is installed to interact with a Microsoft Windows client or server. Samba can be run on many platforms and uses the TCP/IP installed on the host server for blending the two systems.

3. Add and enable the Windows NT authentication module in the appropriate realm.

The following procedures are related to Windows NT authentication.

- [“To Define Default Values for Windows NT Authentication Global Attributes” on page 35](#)
- [“To Install the SAMBA Client on Solaris” on page 36](#)
- [“To Install the SAMBA Client on Linux” on page 36](#)

▼ To Define Default Values for Windows NT Authentication Global Attributes

- 1 **Login as the administrator of the top-level realm; by default, `amadmin`.**
- 2 **Select the Configuration tab.**
- 3 **Click Authentication.**
- 4 **Click Windows NT under the Authentication heading.**
- 5 **Configure the global attributes by adding or changing the values.**
Click Help for attribute descriptions.
- 6 **Click Save.**
- 7 **Click Back to Configuration.**
- 8 **Logout.**

▼ To Install the SAMBA Client on Solaris

- 1 Download the Samba client from
<http://www.sun.com/software/download/products/3e3af224.html>.
- 2 Install the Samba client into the `/bin` directory of the system on which Federation Manager is deployed.

▼ To Install the SAMBA Client on Linux

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

- 1 Copy the client binary to the `/bin` directory of the system on which Federation Manager is deployed.
- 2 (Optional) If you have multiple interfaces, set them by configuring the `smb.conf` file so it passes to the `mbclient`.

Types of Authentication

The Authentication Service provides different ways in which authentication can be configured and applied. These types of authentication are accessed by specifying Login URL parameters (as discussed in “[The Authentication Service User Interface Login URL](#)” on page 73), or through the authentication APIs (as discussed in XXXXX Using Authentication APIs and SPIs in Access Manager 7.1 Developer's Guide). The types of authentication include:

- “[Realm-based Authentication](#)” on page 38
- “[Role-based Authentication](#)” on page 42
- “[Service-based Authentication](#)” on page 46
- “[User-based Authentication](#)” on page 50
- “[Authentication Level-based Authentication](#)” on page 53
- “[Module-based Authentication](#)” on page 56

For each of these types of authentication, the user can either pass or fail. If the user is successfully authenticated, Federation Manager does the following:

1. Confirms that the authenticated user has a profile defined in the data store and whether that profile is active.

The User Profile attribute in the Core Authentication module can be defined as Required, Dynamic, Dynamic with User Alias, or Ignored. If the value is Required, the profile is active. (This is the default case.) If the value is Dynamic or Dynamic with User Alias, the Authentication Service will create the user profile in the Data Store. If the value is Ignore, the user validation will not be done.

2. Executes the Authentication Post Processing service provider interface (SPI), if applicable.
To accomplish any post processing, the `AMPostAuthProcessInterface` must be implemented. If the Authentication Post Processing Classes attribute in the Core Authentication module contains a value, the SPI is executed. It can be executed on either successful or failed authentication or on logout.
3. The following properties are added to, or updated in, the session token and the user's session is activated.

realm	This is the DN of the realm to which the user belongs.
Principal	This is the DN of the user.
Principals	This is a list of names to which the user has authenticated. All Principals must map to the same user. The User Alias List in the user's profile contains this list. (This property may have more than one value defined as a pipe separated list.)

Note – This is a function of authentication chaining. See [“Creating an Authentication Chain” on page 60](#).

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

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

UserToken	This is a user name. (All Principals must map to the same user. The UserToken is the user name to which they map.)
Host	This is the host name or IP address for the client.
authLevel	This is the highest level to which the user has authenticated.
AuthType	This is a pipe separated list of authentication modules to which the user has authenticated (for example, <code>module1 module2 module3</code>).
clientType	This is the device type of the client browser.
Locale	This is the locale of the client.
CharSet	This is the determined character set for the client.
Role	Applicable for role-based authentication only, this is the role to which the user belongs.

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

4. Redirects the user.

The user can be redirected to either a Federation Manager page or a URL. The redirection is based on the authentication type's order of precedence as explained in the appropriate section:

- [“Successful Realm-based Authentication Redirection Precedence” on page 40](#)
- [“Successful Role-based Authentication Redirection Precedence” on page 43](#)
- [“Successful Service-based Authentication Redirection Precedence” on page 47](#)
- [“Successful User-based Authentication Redirection Precedence” on page 51](#)
- [“Successful Authentication Level-based Authentication Redirection Precedence” on page 54](#)
- [“Successful Module-based Authentication Redirection Precedence” on page 57](#)

If the user has failed authentication, Federation Manager simply redirects the user based on the authentication type's order of precedence. See the appropriate section for more information.

- [“Failed Realm-based Authentication Redirection Precedence” on page 41](#)
- [“Failed Role-based Authentication Redirection Precedence” on page 44](#)
- [“Failed Service-based Authentication Redirection Precedence” on page 48](#)
- [“Failed User-based Authentication Redirection Precedence” on page 52](#)
- [“Failed Authentication Level-based Authentication Redirection URLs” on page 55](#)
- [“Failed Module-based Authentication Redirection Precedence” on page 58](#)

Realm-based Authentication

This type of authentication allows a user to authenticate as a member of a specified realm or sub-realm. The authentication method for a realm is set by adding the Core authentication module to the realm and defining the Authentication attributes.

Note – The Core authentication module is added to the top-level realm during installation. As all new realms are configured under the top-level realm, the Core module is dynamically added and enabled for these realms when created.

The following sections contain additional information regarding realm-based authentication.

- [“Realm-based Authentication Login URLs” on page 39](#)
- [“Realm-based Authentication Redirection Precedence” on page 40](#)
- [“To Configure Realm-based Authentication” on page 42](#)

Realm-based Authentication Login URLs

To access the Authentication Service user interface for realm-based authentication, enter a login URL in the Location Bar of a web browser using the **realm=realmName** parameter as in:

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

Realm-based authentication is the default method of authentication for Federation Manager. Thus,

```
http://server_name.domain_name:port/amserver/UI/Login
```

will also access the Authentication Service user interface for realm-based authentication. Additionally, the org or domain parameter can be used as in:

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

Tip – The value of the org or domain parameter must match the value defined as the Realm Name in the realm's profile.

[Remark 1–2 Reviewer: Added org in the order below. Please confirm that this is correct? Also please explain number 4.] The realm of a request for authentication is determined by checking the following, in order of precedence:

1. The domain parameter
2. The realm parameter
3. The org parameter
4. The value of the DNS Alias Names attribute in the Administration serviceXXXXX

After calling the correct realm, the authentication module(s) to which the user will authenticate are retrieved. If there is no defined realm, it will be determined from the server name and domain specified in the login URL.

Note – If a user is a member of (and authenticated to) one realm, and tries to authenticate to a different realm, the only two parameters that are passed are **realm** and **module**. For example, assume Ann is a member of, and authenticated to, Realm A. After she tries to authenticate to Realm B, Ann receives a request to do one of the following: start a new authentication process using the module specified for Realm B, or return to her existing authenticated session with Realm A. If Ann chooses to authenticate to Realm B, only the realm name and module name (if specified) are passed and honored for determining authentication in Realm B.

Realm-based Authentication Redirection Precedence

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

- [“Successful Realm-based Authentication Redirection Precedence” on page 40](#)
- [“Failed Realm-based Authentication Redirection Precedence” on page 41](#)

Successful Realm-based Authentication Redirection Precedence

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

1. A URL set by the authentication module.
2. A URL set by a goto Login URL parameter.
3. **[Remark 1–3 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).] The value of the Success URL attribute in the user's profile specific to the client type from which the request was received.**
4. **[Remark 1–4 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's role entry.] The value of the Success URL attribute in the role entry of the user's profile specific to the client type from which the request was received.**
5. **[Remark 1–5 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.] The value of the Default Success Login URL attribute in the realm to which the user is a member specific to the client type from which the request was received.**
6. **[Remark 1–6 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute as a global default.] The value of the Default Success Login URL attribute in the top-level realm specific to the client type from which the request was received.**
7. **[Remark 1–7 Reviewer: Old: A URL set in the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).] The value of the Success URL attribute in the user's profile.**
8. **[Remark 1–8 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's role entry.] The value of the Success URL attribute in the role entry of the user's profile.**
9. **[Remark 1–9 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.] The value of the Success URL attribute in the realm to which the user is a member.**

10. **[Remark 1–10 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute as a global default.] The value of the Default Success Login URL attribute in the top-level realm.**

Failed Realm-based Authentication Redirection Precedence

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

1. A URL set by the authentication module.
2. A URL set by a `gotoOnFail` Login URL parameter.
3. **[Remark 1–11 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-user-failure-url` attribute of the user's entry (`amUser.xml`).] The value of the Failure URL attribute in the user's profile specific to the client type from which the request was received.**
4. **[Remark 1–12 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.] The value of the Failure URL attribute in the role entry of the user's profile specific to the client type from which the request was received.**
5. **[Remark 1–13 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.] The value of the Default Failure Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.**
6. **[Remark 1–14 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute as a global default.] The value of the Default Failure Login URL attribute in the top-level realm specific to the client type from which the request was received.**
7. **[Remark 1–15 Reviewer: Old: A URL set for the `iplanet-am-user-failure-url` attribute in the user's entry (`amUser.xml`).] The value of the Failure URL attribute in the user's profile.**
8. **[Remark 1–16 Reviewer: Old: A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.] The value of the Failure URL attribute in the role entry of the user's profile.**
9. **[Remark 1–17 Reviewer: Old: A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.] The value of the Default Failure Login URL attribute in the realm entry of the user's profile**
10. **[Remark 1–18 Reviewer: Old: A URL set for the `iplanet-am-auth-login-failure-url` attribute as the global default.] The value of the Default Failure Login URL attribute in the top-level realm.**

▼ To Configure Realm-based Authentication

- 1 Login as the realm's administrator.
- 2 Click the name of the realm you are configuring.
- 3 Click the Authentication tab.
- 4 Select an authentication chain from pull down menu of the Default Authentication Chain attribute.
- 5 (Optional) Select an authentication chain from pull down menu of the Administrator Authentication Chain attribute.

This attribute can be used if the authentication module for administrators must be different from the module for users.
- 6 Click Save.
- 7 Log out.
- 8 Access the Authentication Service user interface as described in [“Realm-based Authentication Login URLs” on page 39](#).

Role-based Authentication

This type of authentication allows a user to authenticate as a member of a specified role (either static or filtered) configured within a realm or sub-realm. The authentication method for a role is set by adding the Authentication Configuration Service to the role and choosing the appropriate authentication chain from the displayed choices. For authentication to be successful, the user must belong to the role and they must authenticate to each module defined in the authentication chain selected for that role. The following sections contain additional information regarding role-based authentication.

- [“Role-based Authentication Login URLs” on page 42](#)
- [“Role-based Authentication Redirection Precedence” on page 43](#)
- [“To Configure Role-based Authentication” on page 46](#)

Role-based Authentication Login URLs

To access the Authentication Service user interface for role-based authentication, enter a login URL in the Location Bar of a web browser using the **role=roleName** parameter as in:

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

The parameter sends the user to the authentication configuration defined for the specified role. A user who is not already a member of the specified role will receive an error message when they attempt to authenticate with this parameter.

Role-based Authentication Redirection Precedence

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

- [“Successful Role-based Authentication Redirection Precedence” on page 43](#)
- [“Failed Role-based Authentication Redirection Precedence” on page 44](#)

Successful Role-based Authentication Redirection Precedence

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

1. A URL set by the authentication module.
2. A URL set by a goto Login URL parameter.
3. **[Remark 1–19 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).] The value of the Success URL attribute in the user's profile specific to the client type from which the request was received.**
4. **[Remark 1–20 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the role to which the user has authenticated.] The value of the Success URL attribute in the role entry of the user's profile specific to the client type from which the request was received.**
5. **[Remark 1–21 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of another role entry of the authenticated user. (This option is a fallback if the previous redirection URL fails.)] The value of the Success URL attribute in another role entry of the user's profile specific to the client type from which the request was received.**

Note – This option is a fallback if the previous redirection URL fails.

6. **[Remark 1–22 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.] The value of the Default Success Login URL attribute in the realm to which the user is a member specific to the client type from which the request was received.**

7. **[Remark 1–23 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute as a global default.] The value of the Default Success Login URL attribute in the top-level realm specific to the client type from which the request was received.**
8. **[Remark 1–24 Reviewer: Old: A URL set in the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).] The value of the Success URL attribute in the user's profile.**
9. **[Remark 1–25 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute of the role to which the user has authenticated.] The value of the Success URL attribute in the role entry of the user's profile.**
10. **[Remark 1–26 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute of another role entry of the authenticated user. (This option is a fallback if the previous redirection URL fails.)] The value of the Success URL attribute in another role entry of the user's profile.**

Note – This option is a fallback if the previous redirection URL fails.

11. **[Remark 1–27 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.] The value of the Default Success Login URL attribute in the realm to which the user is a member.**
12. **[Remark 1–28 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute as a global default.] The value of the Default Success Login URL attribute in the top-level realm.**

Failed Role-based Authentication Redirection Precedence

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

1. A URL set by the authentication module.
2. A URL set by a goto Login URL parameter.
3. **[Remark 1–29 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-user-failure-url` attribute of the user's profile (`amUser.xml`).] The value of the Failure URL attribute in the user's profile specific to the client type from which the request was received.**
4. **[Remark 1–30 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the role to which the user has authenticated.] The value of the Failure URL attribute in the role entry of the user's profile specific to the client type from which the request was received.**

5. **[Remark 1–31 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of another role entry of the authenticated user. (This option is a fallback if the previous redirection URL fails.)] The value of the Failure URL attribute in another role entry of the user's profile specific to the client type from which the request was received.**

Note – This option is a fallback if the previous redirection URL fails.

6. **[Remark 1–32 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.] The value of the Default Failure Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.**
7. **[Remark 1–33 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute as a global default.] The value of the Default Failure Login URL attribute in the top-level realm specific to the client type from which the request was received.**
8. **[Remark 1–34 Reviewer: Old: A URL set in the `iplanet-am-user-failure-url` attribute of the user's profile (`amUser.xml`).] The value of the Failure URL attribute in the user's profile.**
9. **[Remark 1–35 Reviewer: Old: A URL set in the `iplanet-am-auth-login-failure-url` attribute of the role to which the user has authenticated.] The value of the Failure URL attribute in the role entry of the user's profile.**
10. **[Remark 1–36 Reviewer: Old: A URL set in the `iplanet-am-auth-login-failure-url` attribute of another role entry of the authenticated user. (This option is a fallback if the previous redirection URL fails.)] The value of the Failure URL attribute in another role entry of the user's profile.**

Note – This option is a fallback if the previous redirection URL fails.

11. **[Remark 1–37 Reviewer: Old: A URL set in the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.] The value of the Default Failure Login URL attribute in the realm entry of the user's profile.**
12. **[Remark 1–38 Reviewer: Old: A URL set in the `iplanet-am-auth-login-failure-url` attribute as a global default.] The value of the Default Failure Login URL attribute in the top-level realm.**

▼ To Configure Role-based Authentication

To configure role-based authentication, you add the Authentication Configuration Service to a role within a realm and choose a configured authentication chain. If you are creating a new role, the Authentication Configuration Service is not automatically assigned to it.

- 1 **Login as the administrator for the realm under which the role is configured.**
- 2 **Click the name of the realm you are configuring.**
- 3 **Click the Subjects tab.**
- 4 **Click the Roles tab.**
- 5 **Click the name of the role you are configuring.**
- 6 **Click the Services tab.**
- 7 **Click Add.**
- 8 **Select Authentication Configuration.**
- 9 **Click Next.**
- 10 **Select the appropriate authentication chain from those displayed.**
- 11 **Click Finish.**
- 12 **Log out.**
- 13 **Access the Authentication Service user interface as described in [“Role-based Authentication Login URLs”](#) on page 42.**

Service-based Authentication

This type of authentication allows a user to authenticate to a specified authentication chain in a realm or sub realm. For authentication to be successful, the user must authenticate to each module defined in the chain. The following sections contain additional information regarding service-based authentication.

- [“Service-based Authentication Login URLs”](#) on page 47
- [“Service-based Authentication Redirection Precedence”](#) on page 47
- [“To Configure Service-based Authentication”](#) on page 49

Service-based Authentication Login URLs

To access the Authentication Service user interface for service-based authentication, enter a login URL in the Location Bar of a web browser using the **service=AuthChainName** parameters as in:

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

Additionally, you can add the **realm=realmName** attribute as in:

```
http://server_name.domain_name:port/amserver/UI/Login?realm=opensso&service=ldapService
```

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

Service-based Authentication Redirection Precedence

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

- [“Successful Service-based Authentication Redirection Precedence” on page 47](#)
- [“Failed Service-based Authentication Redirection Precedence” on page 48](#)

Successful Service-based Authentication Redirection Precedence

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

1. A URL set by the authentication module.
2. A URL set by a goto Login URL parameter.
3. **[Remark 1–39 Reviewer: Old: A URL set in the clientType custom files for the iplanet-am-user-success-url attribute of the user’s profile (amUser.xml).] The value of the Success URL attribute in the user’s profile specific to the client type from which the request was received.**
4. **[Remark 1–40 Reviewer: Old: A URL set in the clientType custom files for the iplanet-am-auth-login-success-url attribute of the service to which the user has authenticated.] The value of the Success URL attribute in the service to which the user is authenticated specific to the client type from which the request was received.**
5. **[Remark 1–41 Reviewer: Old: A URL set in the clientType custom files for the iplanet-am-auth-login-success-url attribute of the user’s role entry.] The value of the Success URL attribute in the role entry of the user’s profile specific to the client type from which the request was received.**

6. **[Remark 1–42 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.] The value of the Default Success Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.**
7. **[Remark 1–43 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute as a global default.] The value of the Default Success Login URL attribute of the top-level realm specific to the client type from which the request was received.**
8. **[Remark 1–44 Reviewer: Old: A URL set in the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`)).] The value of the Success URL attribute in the user's profile.**
9. **[Remark 1–45 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute of the service to which the user has authenticated.] The value of the Success URL attribute in the service to which the user is authenticated.**
10. **[Remark 1–46 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's role entry.] The value of the Success URL attribute in the role entry of the user's profile.**
11. **[Remark 1–47 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.] The value of the Default Success Login URL attribute in the realm entry of the user's profile.**
12. **[Remark 1–48 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute as a global default.] The value of the Default Success Login URL attribute of the top-level realm.**

Failed Service-based Authentication Redirection Precedence

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

1. A URL set by the authentication module.
2. A URL set by a `goto` Login URL parameter.
3. **[Remark 1–49 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-user-failure-url` attribute of the user's profile (`amUser.xml`)).] The value of the Failure URL attribute in the user's profile specific to the client type from which the request was received.**
4. **[Remark 1–50 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the service to which the user has authenticated.] The value of the Failure URL attribute of the service to which the user has authenticated specific to the client type from which the request was received.**

5. [Remark 1–51 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.] The value of the Failure URL attribute in the role entry of the user's profile specific to the client type from which the request was received.
6. [Remark 1–52 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.] The value of the Default Failure Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.
7. [Remark 1–53 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute as a global default.] The value of the Default Failure Login URL attribute in the top-level realm specific to the client type from which the request was received.
8. [Remark 1–54 Reviewer: Old: A URL set in the `iplanet-am-user-failure-url` attribute of the user's profile (`amUser.xml`)).] The value of the Failure URL attribute in the user's profile.
9. [Remark 1–55 Reviewer: Old: A URL set in the `iplanet-am-auth-login-failure-url` attribute of the service to which the user has authenticated.] The value of the Failure URL attribute of the service to which the user has authenticated.
10. [Remark 1–56 Reviewer: Old: A URL set in the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.] The value of the Failure URL attribute in the role entry of the user's profile.
11. [Remark 1–57 Reviewer: Old: A URL set in the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.] The value of the Default Failure Login URL attribute in the realm entry of the user's profile
12. [Remark 1–58 Reviewer: Old: A URL set in the `iplanet-am-auth-login-failure-url` attribute as a global default.] The value of the Default Failure Login URL attribute in the top-level realm.

▼ To Configure Service-based Authentication

To configure for service-based authentication, you create an authentication chain within a realm and force authentication with that chain by accessing the Authentication Service user interface with the appropriate login URL.

- 1 Login as the administrator for the realm under which you are creating an authentication chain.
- 2 Click the name of the realm.
- 3 Click the Authentication tab.
- 4 (Optional) Create the appropriate authentication module instances.

- 5 **Create an authentication chain.**
- 6 **Click Save.**
- 7 **Log out.**
- 8 **Access the Authentication Service user interface as described in “[Service-based Authentication Login URLs](#)” on page 47.**

User-based Authentication

This type of authentication allows a user to authenticate using an authentication chain defined as the value of the User Authentication Configuration attribute in the user's profile. For authentication to be successful, the user must authenticate to each module defined in the chain. The following sections contain additional information regarding user-based authentication.

- “[User-based Authentication Login URLs](#)” on page 50
- “[User-based Authentication Redirection Precedence](#)” on page 51
- “[To Configure User-Based Authentication](#)” on page 53

User-based Authentication Login URLs

To access the Authentication Service user interface for user-based authentication, enter a login URL in the Location Bar of a web browser using the **user=userName** parameter as in:

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

Additionally, you can add the **realm=realmName** attribute as in:

`http://server_name.domain_name:port/amserver/UI/Login?realm=opensso&user=eileenA`

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



Caution – On receiving a request for user-based authentication, the Authentication Service first verifies that the user is a valid user and then retrieves the User Authentication Configuration data. In the case where there is more than one valid user profile associated with the value passed by the Login URL, all profiles must map to the specified user. The User Alias List attribute is where mappings to the user's other profiles are defined. If mapping fails, the user is denied a valid session. An exception would be if one of the users is a top-level administrator; in this case, user mapping validation is not done and the user is given top-level administrator rights.

User-based Authentication Redirection Precedence

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

- [“Successful User-based Authentication Redirection Precedence” on page 51](#)
- [“Failed User-based Authentication Redirection Precedence” on page 52](#)

Successful User-based Authentication Redirection Precedence

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

1. A URL set by the authentication module.
2. A URL set by a goto Login URL parameter.
3. **[Remark 1–59 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).] The value of the Success URL attribute in the user's profile specific to the client type from which the request was received.**
4. **[Remark 1–60 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's role entry.] The value of the Success URL attribute in the role entry of the user's profile specific to the client type from which the request was received.**
5. **[Remark 1–61 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.] The value of the Default Success Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.**
6. **[Remark 1–62 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute as a global default.] The value of the Default Success Login URL attribute in the top-level realm specific to the client type from which the request was received.**
7. **[Remark 1–63 Reviewer: Old: A URL set in the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).] The value of the Success URL attribute in the user's profile.**
8. **[Remark 1–64 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's role entry.] The value of the Success URL attribute in the role entry of the user's profile.**
9. **[Remark 1–65 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.] The value of the Default Success Login URL attribute in the realm entry of the user's profile.**

10. **[Remark 1–66 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute as a global default.] The value of the Default Success Login URL attribute in the top-level realm.**

Failed User-based Authentication Redirection Precedence

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

1. A URL set by the authentication module.
2. A URL set by a `gotoOnFail` Login URL parameter.
3. **[Remark 1–67 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-user-failure-url` attribute of the user's entry (`amUser.xml`).] The value of the Failure URL attribute in the user's profile specific to the client type from which the request was received.**
4. **[Remark 1–68 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.] The value of the Failure URL attribute in the role entry of the user's profile specific to the client type from which the request was received.**
5. **[Remark 1–69 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.] The value of the Default Failure Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.**
6. **[Remark 1–70 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute as a global default.] The value of the Default Failure Login URL attribute in the top-level realm specific to the client type from which the request was received.**
7. **[Remark 1–71 Reviewer: Old: A URL set for the `iplanet-am-user-failure-url` attribute in the user's entry (`amUser.xml`).] The value of the Failure URL attribute in the user's profile.**
8. **[Remark 1–72 Reviewer: Old: A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.] The value of the Failure URL attribute in the role entry of the user's profile.**
9. **[Remark 1–73 Reviewer: Old: A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.] The value of the Failure URL attribute in the realm entry of the user's profile**
10. **[Remark 1–74 Reviewer: Old: A URL set for the `iplanet-am-auth-login-failure-url` attribute as the global default.] The value of the Default Failure Login URL attribute in the top-level realm.**

▼ To Configure User-Based Authentication

- 1 Login as the administrator for the realm under which the user is configured.
- 2 Click the name of the realm.
- 3 Click the Subjects tab and click Users.
- 4 Click the name of the user you wish to modify.
The User Profile is displayed.
- 5 Select the authentication chain you want the user to use for the User Authentication Configuration attribute.
- 6 Click Save.
- 7 Log out.
- 8 Access the Authentication Service user interface as described in [“User-based Authentication Login URLs” on page 50](#).

Authentication Level-based Authentication

Each authentication module can be associated with an integer that defines its *authentication level*. A higher authentication level value indicates a higher level of trust for the user once successful authentication has occurred. Once authenticated, the authentication level is stored in the user's session token. When a user requests access to a service, the service can determine if the user is allowed by checking the authentication level set in the user's session token. If the level does not reach the level of trust needed by the service, the user is redirected to complete an authentication process with the appropriate authentication level.

Note – If the user is required to authenticate to multiple authentication modules, and does so successfully, the highest authentication level value will be set in the session token.

The following sections contain additional information regarding authentication level-based authentication.

- [“Authentication Level-based Authentication Login URLs” on page 54](#)
- [“Authentication Level-based Authentication Redirection Precedence” on page 54](#)
- [“To Configure for Authentication Level-based Authentication” on page 56](#)

Authentication Level-based Authentication Login URLs

To access the Authentication Service user interface for authentication level-based authentication, enter a login URL in the Location Bar of a web browser using the **authlevel=authlevelValue** parameter as in:

```
http://server_name.domain_name:port/amserver/UI/Login?authlevel=7
```

With this URL, all authentication modules whose authentication level is larger or equal to the value defined (in this case, 7) will be displayed as a menu from which the user can choose. (If only one matching module is found, the login page for that authentication module will be displayed.) After viewing the relevant list of modules, the user must choose one with which to authenticate.

Note – Once the user selects the module, the authentication process is the same as that described in “[Module-based Authentication](#)” on page 56.

Additionally, you can add the **realm=realmName** attribute as in:

```
http://server_name.domain_name:port/amserver/UI/Login?realm=opensso&authlevel=7
```

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

Authentication Level-based Authentication Redirection Precedence

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

- “[Successful Authentication Level-based Authentication Redirection Precedence](#)” on page 54
- “[Failed Authentication Level-based Authentication Redirection URLs](#)” on page 55

Successful Authentication Level-based Authentication Redirection Precedence

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

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

3. **[Remark 1–75 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).] The value of the Success URL attribute in the user's profile specific to the client type from which the request was received.**
4. **[Remark 1–76 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's role entry.] The value of the Success URL attribute in the role entry of the user's profile specific to the client type from which the request was received.**
5. **[Remark 1–77 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.] The value of the Default Success Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.**
6. **[Remark 1–78 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute as a global default.] The value of the Default Success Login URL attribute in the top-level realm specific to the client type from which the request was received.**
7. **[Remark 1–79 Reviewer: Old: A URL set in the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).] The value of the Success URL attribute in the user's profile.**
8. **[Remark 1–80 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's role entry.] The value of the Success URL attribute in the role entry of the user's profile.**
9. **[Remark 1–81 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.] The value of the Default Success Login URL attribute in the realm entry of the user's profile**
10. **[Remark 1–82 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute as a global default.] The value of the Default Success Login URL attribute in the top-level realm.**

Failed Authentication Level-based Authentication Redirection URLs

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

1. A URL set by the authentication module.
2. A URL set by a `gotoOnFail` Login URL parameter.
3. **[Remark 1–83 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-user-failure-url` attribute of the user's entry (`amUser.xml`).] The value of the Failure URL attribute in the user's profile specific to the client type from which the request was received.**

4. [Remark 1–84 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.] The value of the Failure URL attribute in the role entry of the user's profile specific to the client type from which the request was received.
5. [Remark 1–85 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.] The value of the Default Failure Login URL attribute in the realm entry of the user's profile specific to the client type from which the request was received.
6. [Remark 1–86 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute as a global default.] The value of the Default Failure Login URL attribute in the top-level realm specific to the client type from which the request was received.
7. [Remark 1–87 Reviewer: Old: A URL set for the `iplanet-am-user-failure-url` attribute in the user's entry (`amUser.xml`).] The value of the Failure URL attribute in the user's profile.
8. [Remark 1–88 Reviewer: Old: A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.] The value of the Failure URL attribute in the role entry of the user's profile.
9. [Remark 1–89 Reviewer: Old: A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.] The value of the Default Failure Login URL attribute in the realm entry of the user's profile.
10. [Remark 1–90 Reviewer: Old: A URL set for the `iplanet-am-auth-login-failure-url` attribute as the global default.] The value of the Default Failure Login URL attribute in the top-level realm.

▼ To Configure for Authentication Level-based Authentication

Authentication levels are assigned when creating an Authentication Module Instance.

- 1 See [“To Create an Authentication Module Instance” on page 60](#) to define an authentication level for an authentication module instance.
- 2 Access the Authentication Service user interface as described in [“Authentication Level-based Authentication Login URLs” on page 54](#).

Module-based Authentication

This type of authentication allows an administrator to specify the authentication module to which a user will authenticate. The specified module must be registered to the realm or sub-realm that the user is accessing and the realm's Module-based Authentication attribute (found in the Core Authentication Service) must be enabled. The following sections contain additional information regarding module-based authentication.

- [“Module-based Authentication Login URLs” on page 57](#)
- [“Module-based Authentication Redirection Precedence” on page 57](#)
- [“To Configure for Module-based Authentication” on page 59](#)

Module-based Authentication Login URLs

To access the Authentication Service user interface for module-based authentication, enter a login URL in the Location Bar of a web browser using the **module=authModuleName** parameter as in:

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

Additionally, you can add the **realm=realmName** attribute as in:

```
http://server_name.domain_name:port/amserver/UI/Login?realm=opensso&module=LDAP
```

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

Module-based Authentication Redirection Precedence

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

- [“Successful Module-based Authentication Redirection Precedence” on page 57](#)
- [“Failed Module-based Authentication Redirection Precedence” on page 58](#)

Successful Module-based Authentication Redirection Precedence

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

1. A URL set by the authentication module.
2. A URL set by a goto Login URL parameter.
3. **[Remark 1–91 Reviewer: Old: A URL set in the clientType custom files for the iplanet-am-user-success-url attribute of the user’s profile (amUser.xml).] The value of the Success URL attribute in the user’s profile specific to the client type from which the request was received.**
4. **[Remark 1–92 Reviewer: Old: A URL set in the clientType custom files for the iplanet-am-auth-login-success-url attribute of the user’s role entry.] The value of the Success URL attribute in the role entry of the user’s profile specific to the client type from which the request was received.**

5. **[Remark 1–93 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.]** The value of the **Default Success Login URL** attribute in the realm entry of the user's profile specific to the client type from which the request was received.
6. **[Remark 1–94 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-success-url` attribute as a global default.]** The value of the **Default Success Login URL** attribute in the top-level realm specific to the client type from which the request was received.
7. **[Remark 1–95 Reviewer: Old: A URL set in the `iplanet-am-user-success-url` attribute of the user's profile (`amUser.xml`).]** The value of the **Success URL** attribute in the user's profile.
8. **[Remark 1–96 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's role entry.]** The value of the **Success URL** attribute in the role entry of the user's profile.
9. **[Remark 1–97 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute of the user's realm entry.]** The value of the **Default Success Login URL** attribute in the realm entry of the user's profile.
10. **[Remark 1–98 Reviewer: Old: A URL set in the `iplanet-am-auth-login-success-url` attribute as a global default.]** The value of the **Default Success Login URL** attribute in the top-level realm.

Failed Module-based Authentication Redirection Precedence

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

1. A URL set by the authentication module.
2. A URL set by a `gotoOnFail` Login URL parameter.
3. **[Remark 1–99 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-user-failure-url` attribute of the user's entry (`amUser.xml`).]** The value of the **Failure URL** attribute in the user's profile specific to the client type from which the request was received.
4. **[Remark 1–100 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.]** The value of the **Failure URL** attribute in the role entry of the user's profile specific to the client type from which the request was received.
5. **[Remark 1–101 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.]** The value of the **Default Failure Login URL** attribute in the realm entry of the user's profile specific to the client type from which the request was received.

6. [Remark 1–102 Reviewer: Old: A URL set in the `clientType` custom files for the `iplanet-am-auth-login-failure-url` attribute as a global default.] The value of the Default Failure Login URL attribute in the top-level realm specific to the client type from which the request was received.
7. [Remark 1–103 Reviewer: Old: A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's entry (`amUser.xml`).] The value of the Failure URL attribute in the user's profile.
8. [Remark 1–104 Reviewer: Old: A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's role entry.] The value of the Failure URL attribute in the role entry of the user's profile.
9. [Remark 1–105 Reviewer: Old: A URL set for the `iplanet-am-auth-login-failure-url` attribute of the user's realm entry.] The value of the Default Failure Login URL attribute in the realm entry of the user's profile.
10. [Remark 1–106 Reviewer: Old: A URL set for the `iplanet-am-auth-login-failure-url` attribute as the global default.] The value of the Default Failure Login URL attribute in the top-level realm.

▼ To Configure for Module-based Authentication

- 1 Login as the administrator for the realm you are modifying.
- 2 Click the name of the realm.
- 3 Click the Authentication tab.
- 4 Enable Module-based Authentication if not already.
- 5 Click Save.
- 6 Log out.
- 7 Access the Authentication Service user interface as described in [“Module-based Authentication Login URLs” on page 57](#).

Configuring the Authentication Processes

Authentication requirements are configured by instantiating authentication modules or configuring authentication chains. The following sections contain more information.

- [“To Create an Authentication Module Instance” on page 60](#)
- [“To Create an Authentication Chain” on page 61](#)

Instantiating an Authentication Module

Federation Manager provides the ability to configure multiple instances of an authentication module. (See “[Authentication Modules](#)” on page 22 for information.) One or more authentication module instances are created within a realm to enable an appropriate authentication process for that realm. The following procedure explains how this is done.

▼ To Create an Authentication Module Instance

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

Note – The Administrator Authentication Configuration button under Advanced Properties defines the authentication module instance for administrators only. This attribute can be used if the authentication module for administrators needs to be different from the module for end users. The modules configured in this attribute are picked up when the Federation Manager administrator interface is accessed.

- 3 Click **New** under **Module Instances**.
- 4 Enter a unique **Name** for the authentication module instance.
- 5 Select the **Type** of authentication module for this instance.
- 6 Click **OK**.
- 7 Click the name of the newly created module instance and edit the properties for that module. Click **Help** for attribute definitions.
- 8 (Optional) Repeat these steps to add additional module instances.

Creating an Authentication Chain

One or more authentication modules instances can be configured so a user must pass authentication credentials to all of them. This is referred to as *authentication chaining*. When an authentication chain is configured, a user interacts with each of the authentication module instances in the chain, from the top down, in order to pass the authentication process. A criteria flag is assigned to each instance in the chain that defines how to continue processing the ordered group of modules if, for example, authentication to one of the modules fails. Once authentication to all modules in the chain has been successfully achieved, the Authentication

Service validates that all user identifiers used map to the same user. (These identifiers must be configured in the User Alias List attribute in the user's profile.) If this validation is successful, a session token is issued for the user and the policy evaluation process begins. Authentication chains can be assigned to a realm, a user, a role, or a service.

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

▼ To Create an Authentication Chain

1 Click the name of the realm to which you will add the new authentication chain.

2 Select the Authentication tab.

3 Click New under Authentication Chaining.

4 Enter a name for the authentication chain and click OK.

The authentication chain's Properties page will be displayed.

5 Click Add to define one or more authentication module instances for the chain.

6 Chose one or more of the displayed module instances for the chain.

The instances displayed are picked up from the realm's configured Module Instances. See [“To Create an Authentication Module Instance” on page 60](#).

7 Select the criteria for the particular instance in the authentication chain.

These flags establish enforcement criteria for the specified authentication module. There is a hierarchy for this enforcement; Required being the highest and Optional the lowest.

Required Authentication to this module is required to succeed. If any of the required modules in the chain fail, authentication will ultimately fail. However, whether a required module succeeds or not, the authentication process will continue with the next module in the chain.

Requisite Authentication to this module is required to succeed. If it succeeds, the authentication process continues with the next module in the chain. If it fails, control immediately returns back to the requesting application.

Sufficient The module instance is not required to succeed. If it does succeed, control immediately returns back to the requesting application. If it fails, the authentication process continues with the next module in the chain.

Optional The module instance is not required to succeed. Whether it succeeds or fails, the authentication process continues with the next module in the chain.

Overall authentication succeeds only if all Required and Requisite modules succeed. If a Sufficient module is configured and succeeds, then only Required and Requisite modules prior to the Sufficient module need to have succeeded for overall authentication to succeed. If no Required or Requisite modules are configured for an application, then at least one Sufficient or Optional module must succeed.

8 Enter options for the chain.

This allows you to define additional options for the module as **key=value** pairs as in **debug="true"**. Multiple options are separated by a space. There is no limit to the number of options.

9 Define the remaining attributes.

Successful Login URL

This attribute specifies the URL that the user will be redirected to upon a successful authentication. The format is *clientType/URL*.

Failed Login URL

Specifies the URL that the user will be redirected to upon a failed authentication

Authentication Post Processing Class

Defines the name of the Java class used to customize the post authentication process after a login success or failure

10 Click Save.

Initiating Authentication

Clients can trigger the authentication process in several ways. The following sections explain these options.

- [“Web-based Clients” on page 62](#)
- [“Programmatic Clients” on page 63](#)

Web-based Clients

Web-based clients attempting to access resources protected by policy agents are automatically redirected to the Authentication Service. The policy agent intercepts the request and redirects it based on its configured URL. The URL might be an Authentication Service User Interface login URL or the URL of the Distributed Authentication User Interface. See the following for additional information:

- [“The Authentication Service User Interface Login URL” on page 73](#)
- XXXXXlink to Dist AuthXXXXX

- *Sun Java System Access Manager Policy Agent 2.2 User's Guide*

Programmatic Clients

Federation Manager provides both Java APIs and C APIs for writing authentication clients that remote applications can use to gain access to the Authentication Service. Communication between the APIs and the Authentication Service occurs by transporting XML messages over HTTP(S).

Note – The `remote-auth.dtd`, located in `openfm/WEB-INF`, is the template used to format the XML request messages sent to Federation Manager and to parse the XML messages received back from the external application.

Triggering authentication programmatically is a two-step process. First, you create an authentication context object to encapsulate the identity. It contains information other than credentials, such as the type of authentication that is to be performed. Second, you call a login method on the authentication context object. See the following for additional information:

- XXXXX[Link to chapter on Auth API/SPI](#)
- XXXXX[Link to C API Reference](#)
- XXXXX[Link to Java API Ref](#)

◆ ◆ ◆ CHAPTER 2

The Authentication Service User Interface

The Authentication Service implements a web-based user interface for all authentication modules installed in Sun Java System Federation Manager. It is separate from the administration interface (the Federation Manager Console). The Authentication Service User Interface provides a dynamic and customizable means for gathering authentication credentials by displaying login requirement screens (based on the invoked authentication module) to a user requesting access.

- [“The Authentication Service User Interface” on page 65](#)
- [“Accessing the Authentication Service User Interface” on page 72](#)
- [“The Authentication Service User Interface Login URL” on page 73](#)
- [“Customizing the Authentication Service User Interface” on page 81](#)

The Authentication Service User Interface

When a user requests access to a protected resource, the Authentication Service presents a login page, prompting the user for the authentication credentials appropriate to the invoked authentication module. Once the credentials have been passed to Federation Manager and authentication has succeeded, the user can gain access to the requested resource based on the specific privileges configured for the user. The login functionality of the Authentication Service is referred to as the Authentication Service User Interface. The Authentication Service User Interface can be used to authenticate:

- Administrators accessing the Federation Manager Console.
- Users accessing their own profiles with the Federation Manager Console.
- A user accessing a resource defined by a redirection URL parameter appended to the login URL.
- A user accessing a resource protected by a policy agent.

Following is a screen capture of the Authentication Service User Interface for Data Store authentication.



- “XML Files” on page 66
- “JavaServer Pages” on page 68
- “JavaScript Files” on page 70
- “Cascading Style Sheets” on page 70
- “Images” on page 70
- “Localization Properties Files” on page 71

The properties specific to each authentication module's user interface (field labels, required credentials, callback information and so forth) are defined in eXtensible Markup Language

(XML) files. There is one user interface configuration file for each of the installed modules. The `Callbacks` element in each file is used to define the separate screens that can be called during the authentication process. By default, authentication module user interface configuration files are located in the `/openfm/config/auth/default` directory. They are based on the Authentication Module Properties DTD file, `Auth_Module_Properties.dtd`, located in `/openfm/WEB-INF`. The following table lists the files.

TABLE 2-1 List of Authentication Module Configuration Files

File Name	Purpose
<code>AD.xml</code>	Defines a login screen for use with Active Directory authentication.
<code>Anonymous.xml</code>	For anonymous authentication, although there are no specific credentials required to authenticate.
<code>Application.xml</code>	Needed for application authentication.
<code>Cert.xml</code>	For certificate-based authentication although there are no specific credentials required to authenticate.
<code>DataStore.xml</code>	Defines a login screen for use with authentication against a realm's configured data store.
<code>Federation.xml</code>	Defines a login screen for use with SAML authentication and federation.
<code>HTTPBasic.xml</code>	Defines one screen with a header only as credentials are requested via the user's web browser.
<code>JDBC.xml</code>	Defines a login screen for use with Java Database Connectivity (JDBC) authentication.
<code>LDAP.xml</code>	Defines a login screen, a Change Password screen and two error message screens (Reset Password and User Inactive).
<code>Membership.xml</code>	Data interface which can be customized for any domain.
<code>MSISDN.xml</code>	Defines a login screen for use with Mobile Subscriber ISDN (MSISDN).
<code>NT.xml</code>	Defines a login screen for use with NT authentication.
<code>WindowsDesktopSSO.xml</code>	Defines a login screen for use with Windows 2000 single sign-on.
<code>Unix.xml</code>	XXXXXDefines a login screen and an Expired Password screen.

Note – See [“Modifying Login Screens” on page 82](#) for more analysis of the XML files.

JavaServer Pages

All Authentication Service User Interface screens are generated using JavaServer Pages™ (JSP™). These JSP contain HTML code to display static text and graphics, as well as JATO tags to generate information. When the page is displayed in a web browser, it will contain both the static content and dynamic content retrieved via the application code. By default, the JSP are installed in the /openfm/config/auth/default directory. The following table lists these files.

TABLE 2-2 JSP Templates

File Name	Purpose
account_expired.jsp	Informs the user that their account has expired and should contact the system administrator.
auth_error_template.jsp	Informs the user when an internal authentication error has occurred. This usually indicates an Authentication Service configuration issue.
authException.jsp	Informs the user that an error has occurred during authentication.
configuration.jsp	Configuration error page that displays during the Self-Registration process.
disclaimer.jsp	This is a customizable disclaimer page used in the Self-registration authentication module.
Exception.jsp	Informs the user that an error has occurred.
invalidAuthlevel.jsp	Informs the user that the authentication level invoked was invalid.
invalid_domain.jsp	Informs the user that no such domain exists.
invalidPassword.jsp	Informs the user that the password entered does not contain enough characters.
invalidPCookieUserid.jsp	Informs the user that a persistent cookie user name does not exist in the persistent cookie domain.
Login.jsp	This is a Login/Password template.
login_denied.jsp	Informs the user that no profile has been found in this domain.
login_failed_template.jsp	Informs the user that authentication has failed.
Logout.jsp	Informs the user that they have logged out.
maxSessions.jsp	Informs the user that the maximum sessions have been reached.

TABLE 2-2 JSP Templates *(Continued)*

File Name	Purpose
membership.jsp	A login page for the Self-registration module.
Message.jsp	A generic message template for a general error not defined in one of the other error message pages.
missingReqField.jsp	Informs the user that a required field has not been completed.
module_denied.jsp	Informs the user that the user does not have access to the module.
module_template.jsp	A customizable interface page that can be used for a custom authentication module.
new_org.jsp	This page is displayed when a user with a valid session in one organization wants to login to another organization.
noConfig.jsp	Informs the user that no module configuration has been defined.
noConfirmation.jsp	Informs the user that the password confirmation field has not been entered.
noPassword.jsp	Informs the user that no password has been entered.
noUserName.jsp	Informs the user that no user name has been entered. It links back to the login page.
noUserProfile.jsp	Informs the user that no profile has been found. It gives the option to try again or select New User.
org_inactive.jsp	Informs the user that the organization to which they are attempting authentication is no longer active.
passwordMismatch.jsp	This page is called when the password and confirming password do not match.
profileException.jsp	Informs the user that an error has occurred while storing the user profile.
Redirect.jsp	This page carries a link to a page that has been moved.
register.jsp	A user self-registration page.
session_timeout.jsp	Informs the user that their current login session has timed out.
userDenied.jsp	Informs the user that they do not possess the necessary role (for role-based authentication.)
userExists.jsp	This page is called if a new user is registering with a user name that already exists.
user_inactive.jsp	Informs the user that they are not active.

TABLE 2-2 JSP Templates *(Continued)*

File Name	Purpose
userPasswordSame.jsp	Called if a new user is registering with a user name field and password field have the same value.
wrongPassword.jsp	Informs the user that the password entered is invalid.

JavaScript Files

JavaScript files are parsed within the Login.jsp login screen file. The JavaScript files are located in the /openfm/js directory. The following table lists the files parsed.

auth.js	Used by Login.jsp to parse all module files for displaying the login screens.
browserVersion.js	Used by Login.jsp to detect the client type.

Cascading Style Sheets

Characteristics of the Authentication Service User Interface such as fonts and font weights, background colors, and link colors are specified in cascading style sheets (CSS). A number of browser-specific style sheets are located in the /openfm/css directory. The following table provides a brief description of each.

TABLE 2-3 Cascading Style Sheets

File Name	Purpose
css_generic.css	Configured for generic web browsers.
css_ie5win.css	Configured for Microsoft® Internet Explorer v.5 for Windows®.
css_ns4sol.css	Configured for Netscape™ Communicator v. 4 for Solaris™.
css_ns4win.css	Configured for Netscape Communicator v.4 for Windows.
styles.css	Used in JSP as a default style sheet.

Images

The default Authentication Service User Interface is branded with Sun Microsystems, Inc. logos and images. The files are located in the /openfm/login_images directory. The following table provides a brief description for each image used.

TABLE 2-4 Authentication Service User Interface Images

File Name	Purpose
Identity_LogIn.gif	Sun Java System Federation Manager banner
Java.gif	Java coffee-cup logo
error_32_sunplex.gif	Error image
info_32_sunplex.gif	Information image
logo_sun.gif	Sun Microsystems logo
spacer.gif	A one pixel clear image used for layout purposes.
warning_32_sunplex.gif	Warning image

Localization Properties Files

A *localization properties file*, also referred to as an internationalization (i18n) properties file, specifies the screen text and error messages that an administrator or user will see when directed to an authentication module's configuration page. Localization properties files are located in the `/openfm/WEB-INF/classes` directory. The files in this directory are global to the instance in which they are nested.

Each authentication module has its own properties file that follows the naming format `amAuthmodulename.properties`; for example, `amAuthLDAP.properties`. As the default character set of Federation Manager is ISO-8859-1, all values are in English. Java applications though can be adapted to various languages without code changes by translating the data following the equal sign (=) in each key/value pair. The modified file is then copied into the appropriate locale directory. (See [“Directories for Custom Files” on page 91](#).) The following table summarizes the localization properties files for each module.

TABLE 2-5 List of Localization Properties Files

File Name	Purpose
<code>amAuth.properties</code>	Defines the parent Core Authentication Service.
<code>amAuthAD.properties</code>	Defines the Active Directory Authentication Module.
<code>amAuthAnonymous.properties</code>	Defines the Anonymous Authentication Module.
<code>amAuthApplication.properties</code>	For Access Manager internal use only. Do not remove or modify this file.
<code>amAuthCert.properties</code>	Defines the Certificate Authentication Module.
<code>amAuthConfig.properties</code>	Defines the Authentication Configuration Module.

TABLE 2-5 List of Localization Properties Files *(Continued)*

File Name	Purpose
amAuthContext.properties	Defines the localized error messages for the AuthContext Java class.
amAuthContextLocal.properties	For Access Manager internal use only. Do not remove or modify this file.
amAuthDataStore.properties	Defines the Data Store Authentication Module.
amAuthFederation.properties	Defines the Federation Authentication Module.
amAuthHTTPBasic.properties	Defines the HTTP Basic Authentication Module.
amAuthJDBC.properties	Defines the Java Database Connectivity (JDBC) Authentication Module.
amAuthLDAP.properties	Defines the LDAP Authentication Module.
amAuthMembership.properties	Defines the Membership Authentication Module.
amAuthMSISDN.properties	Defines the Mobile Subscriber ISDN Authentication Module.
amAuthNT.properties	Defines the Windows NT Authentication Module.
amAuthUI.properties	Defines labels used in the authentication user interface.
amAuthUnix.properties	Defines the UNIX Authentication Module.

Note – The alphanumeric keys in the localization properties files (a1, a2, and so forth) map to the `i18nKey` attribute fields defined in the authentication module's service XML file. These files follow the naming format `amAuth $module$ name.xml`. The alphanumeric keys determine the order in which the fields are displayed on the module's service page in the Federation Manager Console. They are taken in alphabetical and then numerical order (a1, a2 is followed by b1, b2 and so forth). For example, if a new attribute is added and needs to be displayed at the top of the service page, the alphanumeric key should have a value of a1. The second attribute should then have a value of either a2 or b1, and so forth. Note also that a10 comes before a2.

Accessing the Authentication Service User Interface

To access the default Authentication Service User Interface login page, use the following URL:

protocol://server_name.domain_name:port/deployURI/UI/Login

Note – More information on the URLs used to access the Authentication Service User Interface login page can be found in [“The Authentication Service User Interface Login URL” on page 73](#).

To access the default Authentication Service User Interface logout page, use the following URL:

protocol://server_name.domain_name:port/deployURI/UI/Logout

The Authentication Service User Interface Login URL

The Authentication Service User Interface is accessed by entering a login URL in the Location Bar of a web browser. This default login URL is:

http://server_name.domain_name:port/deployURI/UI/Login

Note – The default *deployURI* configured during installation is *openfm*.

The login URL can be appended with parameters to define specific authentication requirements. A *parameter* is a name/value pair appended to the end of a URL. The parameter starts with a question mark (?) and takes the form *name=value*. A number of parameters can be combined in one login URL. If more than one parameter exists, they are separated by an ampersand (&). If parameter combinations are used, they must adhere to the following guidelines:

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

An example Authentication Service User Interface login URL appended with parameters might be:

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

The following sections describe the parameters supported by the Authentication Service User Interface. Additional examples and specific authentication information can be found in [“Types of Authentication” on page 36](#).

- “Redirection Parameters” on page 74
- “Organization Parameters” on page 75
- “Authentication Method Parameters” on page 77
- “Login Function Parameters” on page 78

Tip – To simplify an Authentication Service User Interface login URL for distribution throughout an enterprise, an administrator might configure an HTML page with a simple URL that possesses links to the more complicated login URLs.

Redirection Parameters

The following sections contain information on the redirection parameters that can be appended to the Authentication Service User Interface login URL:

- “goto Parameter” on page 74
- “gotoOnFail Parameter” on page 74

goto Parameter

A **goto=successful_authentication_URL** parameter overrides the value defined in the realm's Default Success Login URL. It redirects to the specified URL when a user has been successfully authenticated. A **goto=logout_URL** parameter can also be used to redirect to a specified URL when the user is logging out. An example of a login URL with a goto redirect is:

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

An example of a logout URL with a goto redirect is:

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

Note – There is an order of precedence in which Federation Manager looks for successful authentication redirection URLs. Because these redirection URLs and their order are based on the method of authentication, this order (and related information) is detailed in “[Types of Authentication](#)” on page 36.

gotoOnFail Parameter

A **gotoOnFail=failed_authentication_URL** parameter overrides the value defined in the realm's Default Failure Login URL. It will redirect to the specified URL if a user has failed the authentication process. An example of a login URL with a gotoOnFail redirect is:

`http://server_name.domain_name:port/openfm/UI/Login?gotoOnFail=
http://www.sun.com/auth_fail.html`

Note – There is an order of precedence in which Federation Manager looks for failed authentication redirection URLs. Because these redirection URLs and their order are based on the method of authentication, this order (and related information) is detailed in [“Types of Authentication” on page 36](#).

Organization Parameters

The following sections contain information on the parameters that can be appended to the Authentication Service User Interface login URL to access a specific organization:

- [“realm Parameter” on page 75](#)
- [“org Parameter” on page 76](#)
- [“domain Parameter” on page 76](#)

realm Parameter

The **realm=realmName** parameter allows a user to authenticate to a specified realm. The value of the realm parameter must match the value defined in the Realm Name attribute of the realm's profile. A user who is not already a member of the specified realm will receive an error message when they attempt to authenticate to it.

Note – A user profile can be dynamically created if all of the following are true:

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

An example of a login URL with a realm parameter is:

`http://server_name.domain_name:port/openfm/UI/Login?realm=opensso`

From the value of this parameter, the correct login page is displayed (based on the realm and a locale setting). If this parameter is not set, the default is the top-level realm based on the server and domain set in the URL.

org Parameter

The **org=realmName** parameter allows a user to authenticate to a specified realm. The value of the org parameter must match the value defined in the Realm Name attribute of the realm's profile. A user who is not already a member of the specified realm will receive an error message when they attempt to authenticate to it.

Note – A user profile can be dynamically created if all of the following are true:

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

An example of a login URL with an org parameter is:

```
http://server_name.domain_name:port/openfm/UI/Login?org=opensso
```

From the value of this parameter, the correct login page is displayed (based on the realm and its locale setting). If this parameter is not set, the default is the top-level realm based on the server and domain set in the URL.

Note – The use and functionality of the org parameter is the same as that of the realm parameter.

domain Parameter

The **domain=realmName** parameter allows a user to authenticate to a specified realm. The value of the domain parameter must match the value defined in the Realm Name attribute of the realm's profile. A user who is not already a member of the specified realm will receive an error message when they attempt to authenticate to it.

Note – A user profile can be dynamically created if all of the following are true:

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

An example of a login URL with a domain parameter is:

```
http://server_name.domain_name:port/openfm/UI/Login?domain=opensso
```

From the value of this parameter, the correct login page is displayed (based on the realm and its locale setting). If this parameter is not set, the default is the top-level realm based on the server and domain set in the URL.

Note – The use and functionality of the domain parameter is the same as that of the realm and org parameters.

Authentication Method Parameters

The following sections contain information on the parameters that can be appended to the Authentication Service User Interface login URL to access a specified authentication module or chain:

- [“authlevel Parameter” on page 77](#)
- [“module Parameter” on page 77](#)
- [“role Parameter” on page 78](#)
- [“service Parameter” on page 78](#)
- [“user Parameter” on page 78](#)

authlevel Parameter

When an authentication Module Instance is created for a realm, it is given an authentication level value. An **authlevel=value** parameter tells the Authentication Service to call a module with an authentication level equal to or greater than the specified value. An example of a login URL with a specified authentication level is:

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

See [“To Create an Authentication Module Instance” on page 60](#) and [“Authentication Level-based Authentication” on page 53](#) for more information.

module Parameter

The **module=moduleName** parameter allows authentication via the specified authentication module. Any module registered under the realm to which the user belongs and selected as one of that realm’s Module Instances in the Core attributes of the Authentication Service can be specified. (See [“To Create an Authentication Module Instance” on page 60](#) for more information.) An example of a login URL with a specified authentication module instance is:

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



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

role **Parameter**

A **role=roleName** parameter sends the user to the authentication process configured for the specified role. A user who is not already a member of the specified role will receive an error message when they attempt to authenticate with this parameter. An example of a login URL that specifies authentication based on a particular role is:

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

service **Parameter**

The **service=serviceName** parameter allows a user to authenticate via a configured authentication chain. (See [“To Create an Authentication Chain” on page 61](#) for more information.) An example of a login URL with a specified authentication chain is:

```
http://server_name.domain_name:port/openfm/UI/Login?service=newchain1
```

user **Parameter**

The **user=userName** parameter forces authentication based on the authentication chain defined in the User Authentication Configuration attribute of the user’s profile (rather than that defined by the realm of which the user is a member). With this option, one user can be configured to authenticate using the Certification module while another can be configured to authenticate using the LDAP module. An example of a login URL that specifies user-based authentication is:

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

Login Function Parameters

The following sections contain information on the parameters that can be appended to the Authentication Service User Interface login URL to specify certain requirements for the login process itself:

- [“arg=newsession Parameter” on page 78](#)
- [“iSPCookie=yes Parameter” on page 79](#)
- [“IDTokenN Parameters” on page 79](#)
- [“locale Parameter” on page 80](#)

arg=newsession **Parameter**

The **arg=newsession** parameter is used to end a user’s current session and begin a new one. The Authentication Service will destroy a user’s existing session token and perform a new login in one request. This option is typically used with the Anonymous Authentication module; the user first authenticates anonymously, and then hits the register or login link. An example of a login URL with the **arg=newsession** parameter is simply:

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

iPSPCookie=yes Parameter

The **iPSPCookie=yes** parameter allows a user to login with a persistent cookie. A *persistent cookie* is one that continues to exist after the browser window is closed. In order to use this parameter, the realm to which the user is logging in must have Persistent Cookies Mode enabled in the Core attributes of the Authentication Service. Once the user authenticates and the browser is closed, the user can login with a new browser session and will be directed to the Federation Manager Console without having to reauthenticate. This will work until the value of the Persistent Cookie Maximum Time attribute, also specified in the Core attributes of the Authentication Service, elapses. An example of a login URL which specifies a persistent cookie for a successfully authenticated user is:

`http://server_name.domain_name:port/openfm/UI/Login?iPSPCookie=yes`

IDTokenN Parameters

The **IDTokenN=value** parameter enables a user to pass authentication credentials using a URL or HTML forms. Using this parameter, a user can be authenticated without accessing the Authentication Service User Interface. This process is called *Zero Page Login*. Zero Page Login works only for authentication modules that collect credentials using one login page. The value of *N* in the parameter is replaced with a number that maps to the fields on the authentication module's login page; the values for the numbered parameters would be the same as the values the user would enter on the login page itself. For example, the LDAP authentication module would use IDToken1 for the User Name information, and IDToken2 for the user's password. In this case, the login URL might be:

`http://server_name.domain_name:port/openfm/UI/Login?
module=LDAP&IDToken1=jsmith&IDToken2=1234`

Note – module=LDAP can be omitted if LDAP is the default authentication module.

This parameter can also be used for Anonymous authentication. In that case, the login URL might be:

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



Caution – The token names Login.Token0, Login.Token1, ... Login.TokenN are still supported for this parameter but will be deprecated in a future release. It is recommended to use the new **IDTokenN=value** parameter.

locale **Parameter**

Remark 2–1 **email to Chand Basha and dsame-auth 4/13/07; FM not yet localized so no directory structure**
Reviewer

Federation Manager has the capability to display login screens localized into languages other than English. The `locale=localeName` parameter allows the specified language locale to take precedence over any other defined locales. Out-of-the-box, the following locales are supported in Federation Manager:

en_US	English (default)
fr	French
de	German
es	Spanish
ja	Japanese
zh_cn	Simplified Chinese

A locale-specific login page is displayed after searching for the locale configuration in the following places, order-specific:

1. Value of `locale` parameter in login URL
The value of the `locale=localeName` parameter takes precedence over all other defined locales.
2. Web browser locale setting
The value from the browser is used if not found elsewhere.

The locale derived from this pecking order is stored in the user's session token and Federation Manager uses it to load the localized authentication module only. After successful authentication, the locale defined in the User Preferred Language attribute of the user's profile is used. If none is set, the locale used for authentication will be carried over. An example of a login URL with a specified locale is:

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

Note – If the user parameter is used, the locale of the Authentication Service User Interface is based on the value of the User Authentication Configuration attribute of the user's profile, if defined. However, if the `locale` parameter is also specified, that setting will override the user's setting for the Authentication Service User Interface only.

Customizing the Authentication Service User Interface

Federation Manager provides customization support for the Authentication Service User Interface. When the Federation Manager web archive (WAR), `openfm.war`, is deployed, the files in the WAR are exploded into a *staging* directory. The staging directory contains all the files that comprise the Federation Manager. From the staging directory, the web container deploys an instance of the web application by copying these files to a second *instance* directory. Thus, any modifications to the Federation Manager web application must first be made to the staging directory. The staging directory is then re-archived and the resulting WAR is redeployed. If you are using Sun Java System Web Server for your web container, the paths to these two directories might look like this:

staging directory	<code>/var/opt/SUNWwbsvr7/admin-server/config-store/serverName.domain/web-app/serverName.domain/openfm</code>
instance directory	<code>/var/opt/SUNWwbsvr7/virtual-serverName.domain/web-app/serverName.domain/openfm</code>

Look in your web containers documentation for the proprietary paths to deployed WAR directories. See XXXXXUpdating and Redeploying services.war in this chapter for instructions. See XXXXXChapter 10, Updating and Redeploying Access Manager WAR Files for general information on updating and redeploying Access Manager .war files.

Note – Where necessary, *web-container-staging* and *web-container-instance* are used to differentiate the staging directory from the instance directory. Because Federation Manager is a deployed .war, the files are looked up from the deployed location (that is, the instance directory). Changes made to the instance directory will be lost if the application is redeployed without porting the changes to the staging directory.

You can modify many of the files that comprise the Authentication Service User Interface. These files (documented in [“The Authentication Service User Interface” on page 65](#)) are used to convey the graphical-based representations of the login requirement screens, logout screens, and error messages for each authentication module. Changes can be made on a number of levels; for example, to reflect authentication to different organizations (or configured realms) via branding, to define authentication methods for services based on security needs, or to support different types of client applications. The Authentication Service User Interface can be customized for the following levels:

Top-level realm	Files can be customized for a user authenticating to the Federation Manager top-level organization.
Sub realm	Files can be customized for a user authenticating to any sub-realm be it sub to the top-level realm or any specific sublevel realm.

Locale	Files can be customized to display a translated user interface at any realm level.
Client type	Files can be customized to support multiple client types (HTML for computer browsers, WML for wireless browsers, and so forth).
Service	Files can be customized for a user authenticating to a specific service.

The following sections contain more specific information on customizing the Authentication Service User Interface.

- [“Modifying Login Screens” on page 82](#)
- [“Generating Login Screens with JavaServer Pages” on page 86](#)
- [“Adding Functionality” on page 90](#)
- [“Branding the Authentication Service User Interface” on page 90](#)
- [“Directories for Custom Files” on page 91](#)

A customized interface returned to the requestor is located by the Authentication Service based on a particular hierarchy. See this final section for more information.

Modifying Login Screens

The properties specific to each authentication module's login screens (input field labels, required credentials, number of screens, callback information and so forth) are defined in XML files. There is one user interface configuration file for each of the installed modules. (See [“XML Files” on page 66](#) for a listing.) The file names follow the format *modulename.xml*; for example, *AD.xml* or *Membership.xml*. This section explains how different parts of these XML files can be modified to customize the Authentication Service User Interface. *LDAP.xml*, reproduced below, will be used to illustrate.

EXAMPLE 2-1 *LDAP.xml*

```
<!DOCTYPE ModuleProperties PUBLIC "-//iPlanet//Authentication Module
  Properties XML Interface 1.0 DTD//EN"
    "jar://com/sun/identity/authentication/Auth_Module_Properties.dtd">

<ModuleProperties moduleName="LDAP" version="1.0" >
  <Callbacks length="2" order="1" timeout="120" header="This server uses
    LDAP Authentication" >
    <NameCallback>
      <Prompt> User Name: </Prompt>
    </NameCallback>
    <PasswordCallback echoPassword="false" >
      <Prompt> Password: </Prompt>
    </PasswordCallback>
```

EXAMPLE 2-1 LDAP.xml (Continued)

```

</Callbacks>
<Callbacks length="4" order="2" timeout="120" header="Change Password
<BR></BR>#REPLACE#<BR></BR>">
  <PasswordCallback echoPassword="false" >
    <Prompt>Old Password </Prompt>
  </PasswordCallback>
  <PasswordCallback echoPassword="false" >
    <Prompt> New Password </Prompt>
  </PasswordCallback>
  <PasswordCallback echoPassword="false" >
    <Prompt> Confirm Password </Prompt>
  </PasswordCallback>
  <ConfirmationCallback>
    <OptionValues>
      <OptionValue>
        <Value> Submit </Value>
      </OptionValue>
      <OptionValue>
        <Value> Cancel </Value>
      </OptionValue>
    </OptionValues>
  </ConfirmationCallback>
</Callbacks>
<Callbacks length="0" order="3" timeout="120" header=" Your password
has expired. Please contact service desk to reset your password"
error="true" />
<Callbacks length="0" order="4" timeout="120"
template="user_inactive.jsp" error="true"/>

</ModuleProperties>

```

The `Callbacks` elements in `LDAP.xml` define four different screens for logging in using LDAP authentication: a login screen, a Change Password screen and two error message screens. When prompted, the Authentication Service User Interface constructs callback objects to request the required credentials based on the LDAP module's properties. The client then collects the required information from the user and returns the object to the module for the appropriate action.

To customize the properties of a specific authentication module's user interface, you edit the relevant tag in the module's XML file. For example, the text defined as the value of the `header` attribute and the `prompt` element nested within the `NameCallback` tag in `LDAP.xml` (*This server uses LDAP Authentication* and *User Name:* respectively) are just two of the fields that can be changed. The following sections contain more specific information.

- [“Callbacks Nested Elements” on page 84](#)
- [“Callbacks Attributes” on page 85](#)

Callbacks **Nested Elements**

The following table describes nested elements for the `Callbacks` element.

TABLE 2-6 `Callbacks` Nested Elements

Element	Required	Description
<code>NameCallback</code>	*	<code>prompt</code> tag defines a label for the user identifier field.
<code>PasswordCallback</code>	*	<code>prompt</code> tag defines a label for the password field.
<code>ChoiceCallback</code>	*	Displays a list of choices as radio buttons (by default). Note – See “Login.jsp Common Login Page” on page 86 to change the radio buttons to check boxes. <code>prompt</code> tag defines a label that explains the choices. <code>value</code> tag defines a label for each choice. The XML code below illustrates this. <pre><ChoiceCallback attribute="uid" > <Prompt>A user already exists with the user name you entered. Please choose one of the following user names, or create your own:</Prompt> <ChoiceValues> <ChoiceValue> <Value>Create My Own</Value> </ChoiceValue> </ChoiceValues> </ChoiceCallback></pre>

TABLE 2-6 Callbacks Nested Elements (Continued)

Element	Required	Description
ConfirmationCallback	*	The nested elements OptionValues/OptionValue can provide an array of button text to be rendered on the page. The XML code below illustrates this. <pre><ConfirmationCallback> <OptionValues> <OptionValue> <Value> <required button text> </Value> </OptionValue> </OptionValues> </ConfirmationCallback></pre> If there is only one button on the page, the module is not required to send ConfirmationCallback. If it is not provided, the appropriate button text is picked up from amAuthUI.properties, the global Authentication Service User Interface internationalization properties file located in /openfm/WEB-INF/classes. The LogIn property has a default value of Log In. Changing this value will change the default button text for all authentication modules. To customize the button text per module use the ConfirmationCallback element. Note – The ConfirmationCallback element also receives selected button information from the Authentication Service User Interface (such as which button the user has clicked).
HttpCallback	*	Used by authentication modules with HTTP-based handshaking negotiation.
SAMLCallback		Used for passing either Web artifact or SAML POST response from SAML Service to the SAML authentication module when this module requests for the respective credentials. This authentication module behaves as SAML recipient for both (Web artifact or SAML POST response) and retrieves and validates SAML assertions.

Callbacks **Attributes**

Following is a list that describes attributes for the Callbacks element.

length Displays the number of nested callbacks in the Callbacks element.

Tip – When adding additional callbacks (screens), remember to adjust the length attribute accordingly.

order Defines the sequence of the Callbacks elements.

timeout	Number of seconds the user has to enter credentials before the page times out. Default is 60.
template	Defines a .jsp template to be used to dynamically render the authentication module's interface. If none is defined, Login.jsp is used.

Tip – See “[Login.jsp Common Login Page](#)” on page 86 for more information on Login.jsp.

image	The page-level image attributes for the customization.
header	Text header information to be displayed. Default is Authentication.
error	Indicates whether authentication module needs to terminate the authentication process. If yes, the value is true. Default is false.

Generating Login Screens with JavaServer Pages

The Authentication Service User Interface has a common login page used to render the login screens for all authentication modules (except Membership). Login.jsp collects the module's callback information from the user interface configuration file (as discussed in “[XML Files](#)” on page 66), style information from the appropriate style sheets (as discussed in “[Cascading Style Sheets](#)” on page 70), and images from the appropriate directory (as discussed in “[Images](#)” on page 70) to render the Authentication Service User Interface. The Membership module uses membership.jsp to collect this information and render its login screens and register.jsp to render a self-registration page for first time users. This section contains information on these JavaServer Pages.

- “[Login.jsp Common Login Page](#)” on page 86
- “[membership.jsp and register.jsp for Membership Authentication](#)” on page 88

Login.jsp Common Login Page

For the modules that use it, Login.jsp dynamically displays all text and graphical elements for the invoked authentication module at run time. For example, when a user invokes the LDAP authentication module, the LDAP module header, and the User Name and Password fields are displayed by Login.jsp based on the values in LDAP.xml. The common login page contains code to display the elements listed in the following table. The table includes information on how to customize these elements.

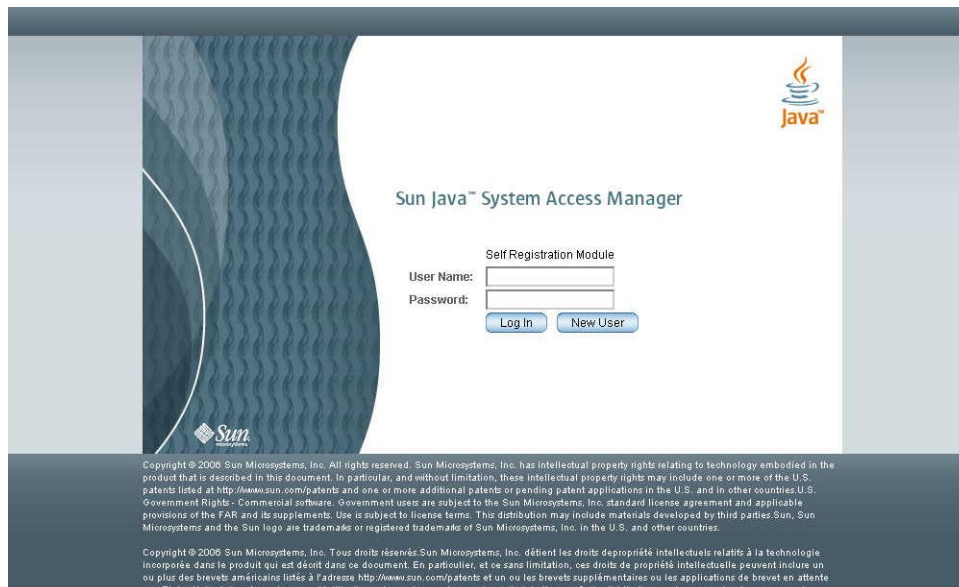
TABLE 2-7 Common Login Page Elements and How to Modify Them

Element	How to Modify
■ Module Header text ■ User Name field size and label ■ Password field size and label	See “Modifying Login Screens” on page 82 for information on how to modify these fields.
Choice values display and label	When the interface displays a series of choices, the field uses radio buttons, by default. Change the value of input type in the following code extract from <code>Login.jsp</code> to <code>checkbox</code> to display the choices with check boxes instead. <pre><jato:content name="selectedChoice"> <input type="radio" name="IDToken<jato:text name="txtParentIndex" />" id="IDToken<jato:text name="txtParentIndex" />" value="<jato:text name="txtIndex" />" class="Rb" checked><jato:text name="txtChoice" />
 </jato:content> <jato:content name="unselectedChoice"> <input type="radio" name="IDToken<jato:text name="txtParentIndex" />" id="IDToken<jato:text name="txtParentIndex" />" value="<jato:text name="txtIndex" />" class="Rb" ><jato:text name="txtChoice" />
 </jato:content></pre>
Images (at the module level)	Change any referenced image to reflect proprietary branding. Images can be in either GIF or JPG format. See “Customizing Images” on page 91 .
Login button	■ The button color and background can be customized by modifying the browser-specific style sheets. See “Customizing Styles” on page 91. ■ The button text can be customized by modifying the authentication module XML file. See “Modifying Login Screens” on page 82.
<code>styles.css</code>	The reference to this style sheet can be changed to a style sheet that reflects proprietary styles.

Tip – Other elements such as a company logo and custom JavaScript™ can be embedded into the page.

membership.jsp and register.jsp for Membership Authentication

The Membership authentication module is implemented similarly to personalized portals. When enabled, a user can create an account, personalize it, and access it as a registered user. The Membership Authentication Service User Interface is comprised of membership.jsp, the Membership login page, and register.jsp, the self registration page. Similar to Login.jsp, membership.jsp generates a login screen as captured below.



The screenshot shows the 'Sun Java™ System Access Manager' login interface. On the left is a decorative blue wavy background with the Sun logo. The main area is white and contains the title 'Sun Java™ System Access Manager' and a 'Self Registration Module' section. This section has input fields for 'User Name:' and 'Password:', followed by 'Log In' and 'New User' buttons. The bottom of the page features a dark grey footer with copyright text for Sun Microsystems, Inc. (© 2000) in both English and French.

A user with an account can log in using this screen or a user with no account can click the New User button to display the self registration page. register.jsp generates the self registration page as captured below.

Sun Java™ System Access Manager

* Indicates required field

Self Registration

* User Name:

* Password:

* Confirm Password:

* First Name:

* Last Name:

* Full Name:

Email Address:

Copyright © 2006 Sun Microsystems, Inc. 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. U.S. Government Rights: Commercial software, Government uses are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements. Use is subject to license terms. This distribution may include materials developed by third parties. Sun, Sun Microsystems and the Sun logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Copyright © 2006 Sun Microsystems, Inc. Tous droits réservés Sun Microsystems, Inc. délient les droits de propriété intellectuelle relatifs à la technologie

Although `register.jsp`, the default self registration page, is generic and can work with any domain, you can customize it. The self registration page is generated using the following code from `Membership.xml`:

```
<Callbacks length="8" order="16" timeout="300"
header="Self Registration" template="register.jsp">
  <NameCallback isRequired="true" attribute="uid">
    <Prompt> User Name: </Prompt>
  </NameCallback>
  <PasswordCallback echoPassword="false" isRequired="true" attribute="userPassword">
    <Prompt> Password: </Prompt>
  </PasswordCallback>
  <PasswordCallback echoPassword="false" isRequired="true">
    <Prompt> Confirm Password: </Prompt>
  </PasswordCallback>
  <NameCallback isRequired="true" attribute="givenname">
    <Prompt> First Name: </Prompt>
  </NameCallback>
  <NameCallback isRequired="true" attribute="sn">
    <Prompt> Last Name: </Prompt>
  </NameCallback>
  <NameCallback isRequired="true" attribute="cn">
    <Prompt> Full Name: </Prompt>
  </NameCallback>
  <NameCallback attribute="mail">
    <Prompt> Email Address: </Prompt>
  </NameCallback>
```

```

    <ConfirmationCallback>
      <OptionValues>
        <OptionValue>
          <Value> Register </Value>
        </OptionValue>
        <OptionValue>
          <Value> Cancel </Value>
        </OptionValue>
      </OptionValues>
    </ConfirmationCallback>
  </Callbacks>

```

By modifying `Membership.xml` (located in the `/openfm/config/auth/default` directory), you can change which data is requested, which is required, and which is optional. (The first three fields are required by default.) For example, to add a user's telephone number as requested data, you would add the following `NameCallback` element to `Membership.xml`.

```

<NameCallback isRequired="true" attribute="telphonenumber">
  <Prompt> Tel:</Prompt>
</NameCallback>

```

You can only specify an attribute that is part of the User Profile. Administrators can add their own user attributes to the User Profile. By default, you can specify any attributes from the following object classes:

- top
- person
- organizationalPerson
- inetOrgPerson
- iplanet-am-user-service
- inetuser

More information on modifying the login screen XML files is in [“Modifying Login Screens” on page 82](#).

Adding Functionality

JavaScript files are parsed within the common login page, `Login.jsp`. They are located in the `/openfm/js` directory. You can add custom functions to these JavaScript files. See [“JavaScript Files” on page 70](#) for a list of those included.

Branding the Authentication Service User Interface

The default Authentication Service User Interface is branded with Sun Microsystems, Inc. logos and images. There are a number of ways to customize the Authentication Service User Interface

to take on the characteristic look of web sites associated with your company. Logos, fonts, background colors and so forth can replace the default configurations. The following sections explore these options.

- [“Customizing Styles” on page 91](#)
- [“Customizing Images” on page 91](#)

Customizing Styles

Characteristics of the web pages such as fonts, font weights, background colors, and link colors are specified in predefined style sheets. To customize the Authentication Service User Interface, you can modify these style sheets. For example, you can change the background-color attribute for a button as follows:

```
.button-content-enabled { background-color: red; }
button-link:link, a.button-link:visited { color: #000;
background-color: red;
text-decoration: none; }
```

You must modify the appropriate style sheet file for the each browser. See [“Cascading Style Sheets” on page 70](#) for a list of the included style sheets. They are located in the `/openfm/css` directory.

Customizing Images

Images are directly referenced in JSP pages. The default Sun Microsystems, Inc. logos and images are located in `/openfm/login_images`. These images can be replaced with images appropriate to your company. Either GIF or JPG images can be used. See [“Images” on page 70](#) for a list of the included images.

Additionally, you could customize the JSP itself to point to a new image name and directory of your choice. Using this method allows customized JSP in different realms to point to different images.

Directories for Custom Files

Customized files for the Authentication Service User Interface must be stored in custom directories created within the `openfm/config/auth/` directory. Drilling down further, these directories and the paths that lead to them are defined based on the level of customization; for example, are you customizing organizational branding or are you adding Wireless Markup Language (WML) files to expand the client types that can access the interface? To illustrate this concept, let's assume the following:

- Federation Manager top-level realm: opensso
- Sub-realm: Sun
- Locales: English (en) and Japanese (ja)

As there are two locales to be configured for access to the Sun Authentication Service User Interface, modified files will reside in a directory based on their respective locale:

- openfm/config/auth/Sun_en/
- openfm/config/auth/Sun_ja/

Now, suppose Human Resources wants its Paycheck service accessible to web browsers and cell phones in both languages. Each of the localized directories listed above will then contain the following subdirectories:

- openfm/config/auth/Sun_en/...
 - ...HR/html/paycheck
 - ...HR/wml/paycheck
- openfm/config/auth/Sun_ja/...
 - ...HR/html/paycheck
 - ...HR/wml/paycheck

Now, suppose Human Resources wants users with Nokia phones and users with Motorola phones to see different WML files. Each of the localized WML directories listed above will then contain the following subdirectories:

- openfm/config/auth/Sun_en/HR/wml...
 - ...nokia/paycheck
 - ...motorola/paycheck
- openfm/config/auth/Sun_ja/HR/wml...
 - ...nokia/paycheck
 - ...motorola/paycheck

Thus, the following directories are created to retrieve the appropriate customized Authentication Service User Interface.

- openfm/config/auth/Sun_en/HR/html/paycheck
- openfm/config/auth/Sun_en/HR/wml/nokia/paycheck
- openfm/config/auth/Sun_en/HR/wml/motorola/paycheck
- openfm/config/auth/Sun_ja/HR/html/paycheck
- openfm/config/auth/Sun_ja/HR/wml/nokia/paycheck
- openfm/config/auth/Sun_ja/HR/wml/motorola/paycheck

The following sections contain information on how to create these directories and how the Authentication Service accesses them.

- [“Creating the Custom Directories” on page 93](#)
- [“Searching the Custom Directories” on page 94](#)

Creating the Custom Directories

The example directory hierarchy uses the directory paths defined and explained in the following table.

TABLE 2–8 Custom Directory Paths

Directory Path Formulas	Purpose
<i>realm_locale/realmPath/filePath/</i>	Contains language-appropriate files with which the specified realm's Authentication Service User Interface will be rendered. This directory might be used if multiple locales are available.
<i>realm/realmPath/filePath/</i>	Contains default files with which the specified realm's Authentication Service User Interface will be rendered. This directory might be used if only one locale is available.
<i>default_locale/realmPath/filePath/</i>	Contains the files to generate the Authentication Service User Interface for top-level realm in the language corresponding to the defined locale. <i>default_en</i> is created at deployment.
<i>default/realmPath/filePath/</i>	Contains the files to generate the Authentication Service User Interface for the top-level realm if no other custom directories are found. <i>default</i> is created at deployment using English files. Caution – The <i>default</i> directory contains all the JSP templates that can be customized as well as the user interface configuration files that contain the callback messages. If you want to customize the JSP and XML files, copy them to a custom <i>default_locale</i> , <i>realm</i> , or <i>realm_locale</i> directory for modification. This directory should be left untouched.

In the directory path formulas, assume the following:

- *realmPath* represents one or more nested sub realm directories as in *subRealm1/subRealm2/....* HR is the sub realm in the example. Additional sub realm paths can be created; for example, HR/employees and HR/contractors.
- *filePath* represents one or more nested client type and service directories as *clientPath/serviceName* where:
 - *clientPath* represents one or more nested client type directories as in *clientType/sub-clientType/... wml* is the client type in the example. Additional sub client types in the example are *wml/nokia* and *wml/motorola*
 - *serviceName* represents one or more nested service directories. *paycheck* is the service directory in the example.

[Remark 2–2 Reviewer: Please review and add info as necessary.] In these paths, defining *realm_locale*, *subRealm*, *clientPath*, and *serviceName* are optional. The realm name you specify

must match the Realm Name attribute set in Federation Manager. For example, if the value of Realm Name is SunMicrosystems, the realm's customized directory should also be SunMicrosystems.

Searching the Custom Directories

Remark 2–3 **Reviewer** Please review and add info as necessary.

Customization of the Authentication Service User Interface is supported at the realm, sub realm, client type and service levels. The Authentication Service finds the customized files by searching the custom directories created in `openfm/config/auth/` in the following order:

- `realm_locale`
- `realm`
- `default_locale`
- `default`

Federation Manager searches for a directory that most closely matches the top-level realm name and locale. If a match is found, it then follows the directory path to find matching sub realms, and then client types. If the desired files are not found, or if the directory paths do not exist, Federation Manager strips off the left most component of the path and attempts to find the desired file in the new path.

Note – Any Authentication Service User Interface messages that are not picked up from these directories are picked up from `amAuthUI_locale.properties` as defined in the JSP. The `amFileLookup` debug file is located in `openfm/debug` and can be used as a troubleshooting tool.

 CHAPTER 3

Distributed Authentication

Sun Java™ System Federation Manager provides a remote authentication interface component to enable secure authentication distributed across firewalls. Installing the Distributed Authentication User Interface on one or more web containers within the non-secure layer of a Federation Manager deployment eliminates the exposure of service URLs to the end user. The following sections contain information regarding distributed authentication.

- [“The Distributed Authentication Process” on page 95](#)
- [“Installing the Distributed Authentication User Interface” on page 98](#)
- [“Configuring the Distributed Authentication User Interface” on page 99](#)
- [“Customizing the Distributed Authentication User Interface XXXXX” on page 102](#)
- [“Accessing the Distributed Authentication User Interface” on page 104](#)

The Distributed Authentication Process

A Distributed Authentication User Interface exists to provide a remote authentication interface between end users and an instance of Federation Manager. The Distributed Authentication User Interface enables a policy agent or an application deployed in a non-secured area of the deployment to communicate with the Federation Manager Authentication Service installed in a secured area of the deployment. A typical scenario would work as follows:

1. An end user sends an HTTP or HTTPS request to access a protected resource using a web-based client.
2. If the request does not have a cookie containing an SSO token, the policy agent protecting the resource issues a redirect to its configured authentication URL; in this case, the URL of the Distributed Authentication User Interface.

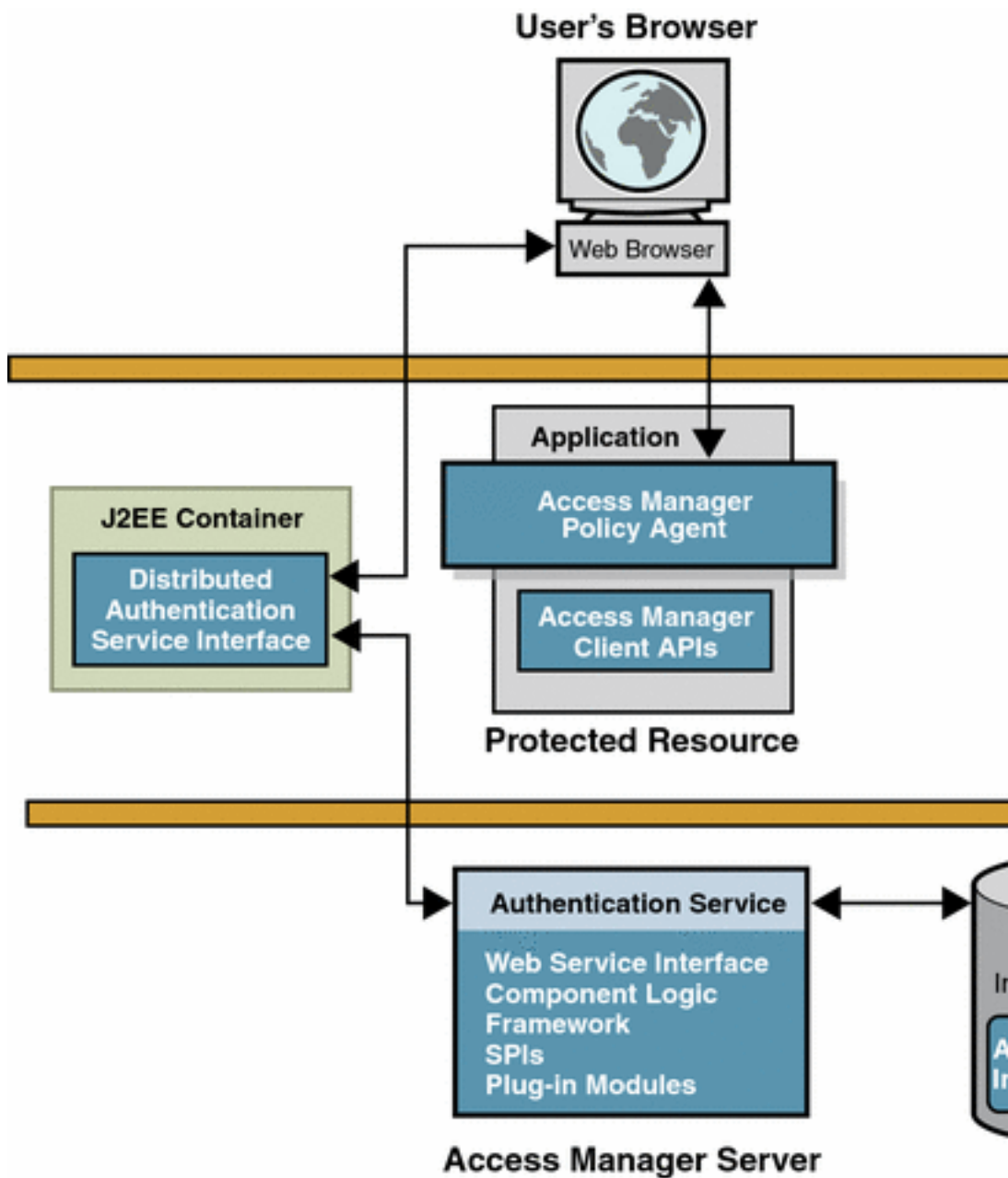
Note – The Distributed Authentication User Interface is installed in the demilitarized zone (DMZ) and usually accessed through a load balancer.

3. The request is redirected to the Distributed Authentication User Interface.
4. The Distributed Authentication User Interface communicates the request to an instance of Federation Manager behind a firewall to determine the appropriate authentication method.
5. Federation Manager determines the appropriate authentication method and returns the callback information to be presented back to the user by the Distributed Authentication User Interface.
6. Using the information from the Federation Manager instance, the Distributed Authentication User Interface server returns a login page to the user's web browser.
7. The end user replies with the login credentials (such as user name and password).
8. The Distributed Authentication User Interface sends the end user's credentials to the Federation Manager instance behind the firewall.

Note – The Federation Manager Client SDK must be installed on the same web container as the Distributed Authentication User Interface as it is used to transport the credentials. See XXXXXX for more information.

9. Federation Manager attempts to authenticate the end user:
 - If the authentication is successful, Federation Manager returns an SSO token, and the Distributed Authentication User Interface redirects the end user to the protected resource.
 - If the authentication is not successful, Federation Manager returns the appropriate error information.

The following figure illustrates this scenario.



Installing the Distributed Authentication User Interface

You can install the Distributed Authentication User Interface on a servlet-compliant web container within the non-secure layer of a Federation Manager deployment. The remote component then works with the authentication client APIs and utility classes to authenticate users.

Tip – Before installing the Distributed Authentication User Interface, the following considerations should be taken into account.

1. The Distributed Authentication User Interface must use the same password encryption key as the Federation Manager server instances with which it communicates.
 2. If you are deploying multiple instances of the Distributed Authentication User Interface behind a load balancer, stickiness is not required. (The load balancer is not required to talk to one instance to complete the authentication process.)
 3. The HTTP Basic and MSISDN authentication modules are not supported through the Distributed Authentication User Interface.
-

▼ To Install the Distributed Authentication User Interface

Before You Begin The Distributed Authentication User Interface has the following dependencies:

- One of the following web containers:

Note – See the XXXXXRelease Notes for specific versions.

- Sun Java System Application Server
- Sun Java System Web Server
- BEA WebLogic Server
- IBM WebSphere Application Server
- The `SUNWjato` (Jato) and the `SUNWamclnt` (Federation Manager Client SDK) packages installed on the web container.

Note – The Distributed Authentication User Interface uses a Jato presentation framework. Jato is an open-source Java/XML translator.

- An instance of Federation Manager available remotely with which the Client SDK will be able to communicate using HTTPS.

- 1 **Log in as root to the machine on which you are installing the Distributed Authentication User Interface.**
- 2 **Install the appropriate web container.**
 - For Application Server or Web Server, use the Sun Java Enterprise System installer.
 - For BEA WebLogic Server or IBM WebSphere Application Server, see the respective product documentation for instructions.
- 3 **Install the Distributed Authentication User Interface using the Federation Manager installer.**

Configuring the Distributed Authentication User Interface

After installing the Distributed Authentication User Interface, you might need to reconfigure your deployment by modifying the attributes in the `amsamplesilent` file.

EXAMPLE 3-1 Distributed Authentication User Interface Sample Configuration File

```
DEPLOY_LEVEL=8
DISTAUTH_PROTOCOL=http
DISTAUTH_HOST=distauth.example.com
DISTAUTH_PORT=80
APPLICATION_USER=username
APPLICATION_PASSWD=application-user-password
AM_ENC_SECRET=am-secret-password
AM_ENC_LOCAL=am-password-encryption-key-used-by-the-Access-Manager-server
DEBUG_LEVEL=error
DEBUG_DIR=/var/opt/SUNWam/logs
```

The following table lists and describes the attributes you can modify.

TABLE 3-1 Distributed Authentication User Interface Configuration Attributes

Attribute	Description
DEPLOY_LEVEL	■ DEPLOY_LEVEL=8 signifies that you are reconfiguring an existing Distributed Authentication User Interface. ■ DEPLOY_LEVEL=18 signifies that you are uninstalling a Distributed Authentication User Interface.

TABLE 3-1 Distributed Authentication User Interface Configuration Attributes *(Continued)*

Attribute	Description
■ SERVER_HOST ■ SERVER_PORT ■ SERVER_DEPLOY_URI ■ CONSOLE_DEPLOY_URI ■ ADMINPASSWD ■ AMLDAPUSERPASSWD ■ COOKIE_DOMAIN ■ AM_ENC_PWD	Values that correspond to the installation of Federation Manager. Caution – The value of AM_ENC_PWD (the password encryption key) must be set to the same value used by the instance of Federation Manager.
■ DS_HOST ■ DS_DIRMGRPASSWD ■ ROOT_SUFFIX	Values that correspond to the instance of Directory Server that was used for the installation of Federation Manager.
■ NEW_OWNER ■ NEW_GROUP	Runtime user and group that will own the processes on the web container in which the Distributed Authentication User Interface will be deployed.
PAM_SERVICE_NAME	If the Linux application environment is being run on the host to which the Distributed Authentication User Interface is deployed, set this value to password.
WEB_CONTAINER	Value corresponds to the web container on which the Distributed Authentication User Interface is or will be deployed. For example, if the web container is Sun Java System Web Server 7, set WEB_CONTAINER=WS. ■ WS for Sun Java System Web Server 7 ■ WS6 for Sun Java System Web Server 6.1 SP5 ■ AS8 (default) for Sun Java System Application Server 8.1 ■ WL8 for BEA WebLogic Server 8.1 ■ WAS5 for IBM WebSphere Application Server 5.1
DISTAUTH_PROTOCOL	Protocol (http or https) used by the web container in which the Distributed Authentication User Interface is or will be deployed. The default is http.
DISTAUTH_HOST	Fully qualified host name for the machine on which the Distributed Authentication User Interface is located. The default is distAuth_sample.com
DISTAUTH_PORT	Port on DISTAUTH_HOST to which the Distributed Authentication User Interface has been or will be deployed. The default is 80.
APPLICATION_USER	User name for the application. The default is username.
APPLICATION_PASSWD	Password of the user for the application. The default is none.
AM_ENC_SECRET	Password encryption secret key from the server. The default is none.

TABLE 3-1 Distributed Authentication User Interface Configuration Attributes *(Continued)*

Attribute	Description
AM_ENC_LOCAL	Password encryption key. The default is none.
DEBUG_LEVEL	Level for the debug service. Values can be: ■ error (default) ■ warning ■ message
DEBUG_DIR	Directory where the debug files will be created. Default directories are:XXXXX ■ Solaris systems: /openfm/debug ■ Linux and HP-UX systems: /var/opt/sun/identity/logs ■ Windows systems: <i>javaes-install-dir</i>\identity\logs where <i>javaes-install-dir</i> represents the Java ES 5 installation directory. The default value is C:\Program Files\Sun\JavaES5.
BASEDIR	The directory to which the Distributed Authentication User Interface was installed.
■ CONSOLE_HOST ■ CONSOLE_PORT ■ CONSOLE_PROTOCOL	Corresponding values for the host on which the Federation Manager Console has been deployed.
CONSOLE_REMOTE	Specifies whether the Federation Manager Console uses a different web container than the Federation Manager server. The default value is false.
DISTAUTH_DEPLOY_URI	Deployment URI that will be used on the local host by the Distributed Authentication User Interface. The default value is /amdistauth.XXXXXX

The following procedure illustrates how to reconfigure the Distributed Authentication User Interface.

If you are using the Configure Now option, see Distributed Authentication UI Server Configuration Variables for the values that you must specify during installation. # If you specified the Configure Later option during the previous step, or if you need to reconfigure the Distributed Authentication UI server, run the amconfig script as follows: 1. Copy the amsamplesilent file and set the configuration variables in the new file. For example, you might name the new file as DistAuth_config. On Windows systems, copy the AMConfigurator.properties file to AMConfigurator-distauth.properties. For the variables that you need to set, see Distributed Authentication UI Server Configuration Variables. 2. Run the amconfig script using the new configuration file. For example, on a Solaris system with Access Manager installed in the default directory: # cd /opt/SUNWam/bin # ./amconfig -s ./DistAuth_config On Windows systems, in the amconfig.bat file, change

AMConfigurator.properties to AMConfigurator-distauth.properties, and then run the edited amconfig.bat file. # Restart the web container on the Distributed Authentication UI server.

▼ To Reconfigure the Distributed Authentication User Interface

- 1 **Make a copy of amsamplesilent located in XXXXXXX.**

On Windows systems, copy the AMConfigurator.properties file to AMConfigurator-distauth.properties.

- 2 **Modify the configuration variables in this new file based on the information in [Table 3-1](#).**

- 3 **Run the amconfig script using the new configuration file as input.**

For example, on a Solaris system with Federation Manager installed in the default directory: #
cd /opt/SUNWam/bin # ./amconfig -s ./DistAuth_config On Windows systems, in the amconfig.bat file, change AMConfigurator.properties to AMConfigurator-distauth.properties, and then run the edited amconfig.bat file.

Customizing the Distributed Authentication User Interface XXXXX

Once the Distributed Authentication User Interface is installed and configured, you can modify the JavaServer Pages (JSP) and configuration properties files to reflect branding and specific functionality for any of the following:

Organization/SubOrganization	This is the organization or sub-organization of the request.
Locale	Locale of the request.
Client Path	Client Type information of the request.
Service Name (serviceName)	Service name for service-based authentication.

▼ To Customize the Distributed Authentication User Interface XXXXX

- 1 **Explode the Distributed Authentication User Interface WAR.**
- 2 **At the command line, go to the directory where the default JSP templates are stored.**

Example:

```
cd DistributedAuth-base/config/auth
```

where *DistributedAuth-base* is the directory where the Distributed Authentication User Interface package is exploded.

3 Create a new directory using the appropriate directory path based on the level of customization.

Use the following form:

```
org_locale/orgPath/filePath  
    org/orgPath/filePath  
    default_locale/orgPath/filePath  
    default/orgPath/filePath
```

where:

```
orgPath = subOrg1/subOrg2  
    filePath = clientPath + serviceName  
    clientPath = clientType/sub-clientType
```

The following are optional: Sub-org, Locale, Client Path, and Service Name. In the following example, orgPath and filePath are optional.

For example, given the following:

```
org = iplanet  
locale = en  
subOrg = solaris  
clientPath = html/nokia/  
serviceName = paycheck
```

the appropriate directory paths for the above are:

```
iplanet_en/solaris/html/nokia/paycheck  
iplanet/solaris/html/nokia/paycheck  
default_en/solaris/html/nokia/paycheck  
default/solaris/html/nokia/paycheck
```

4 Copy all the JSP templates and authentication module configuration properties XML files from the default directory to the new directory.

```
cp DistributedAuth-base/config/auth/default/*.jsp  
    DistributedAuth-base/config/auth/new_directory_path  
  
cp DistributedAuth-base/config/auth/default/*.xml  
    DistributedAuth-base/config/auth/new_directory_path
```

- 5 **(Optional) Modify the files in the new directory to suit your needs.**
 - For information about customizing the `.jsp` files, see **Broken Link (Target ID: ADUEX)**.
 - For information about customizing the `.xml` files, “[Modifying Login Screens](#)” on page 82.
- 6 **Create a new `.WAR` file named `amauthdistui_deploy.war` from *DistributedAuth-base*.**
- 7 **Deploy `amauthdistui_deploy.war`.**

The web container administrator deploys the file in the remote web container.

Accessing the Distributed Authentication User Interface

There are two ways in which the Distributed Authentication User Interface can be accessed:

- “[Using a Policy Agent](#)” on page 104
- “[Using a Web Browser](#)” on page 104

Using a Policy Agent

To access the Distributed Authentication User Interface using a policy agent version 2.2, you must configure the Distributed Authentication User Interface login URL as a value for `com.sun.am.policy.am.login.url` in `AMAgent.properties`. See Sun Java System Access Manager Policy Agent 2.2 User's Guide for more information.

Using a Web Browser

To access the Distributed Authentication User Interface, use the following URL in your browser:

DA_server_protocol://DA_server_host:DA_server_port/DA_deploy_URI/UI/Login

Where:

<i>DA_server_protocol</i>	Protocol (<code>http</code> or <code>https</code>) used by the web container in which the Distributed Authentication User Interface is deployed.
<i>DA_server_host</i>	Fully qualified host name of the machine on which the Distributed Authentication User Interface is deployed.

<i>DA_server_port</i>	Port of the machine on which the Distributed Authentication User Interface is deployed.
<i>DA_deploy_URI</i>	Deployment URI prefix for the Distributed Authentication User Interface. The default value is /amdistauth.

An example URL to access the Distributed Authentication User Interface might be:

`https://daserver.example.com:80/amdistauth/UI/Login`

◆ ◆ ◆ CHAPTER 4

Inside the Authentication Service

The Authentication Service contains many core components. The following authentication processes are invoked.

Client Detection

An initial step in the authentication process is to identify the type of client making the HTTP(S) request. This is referred to as *client detection*. Since the Authentication Service has the capability to process requests from multiple client types, the client characteristics embedded in the initial HTTP(S) request are used to retrieve the appropriate authentication pages. Once the user has been successfully authenticated, the client type is added to the user's session token. For example, when a Netscape browser is used to generate a request, Federation Manager displays an HTML-based Authentication Service User Interface page.

The Client Detection Service can process user requests from clients based in HTML, WML or other protocols. The client detection API are used to determine the protocol of the requesting client browser and retrieve the correctly formatted pages for the particular client type.



Caution – The Federation Manager Console can only be accessed using an HTML client.

Client Detection Process

Since any user requesting access to Federation Manager must first be successfully authenticated, client detection is done within the Authentication Service. When a user's request is passed to Federation Manager, it is directed to the Authentication Service which identifies the browser type. The Client Detection Service uses the value of the User-Agent field stored in the HTTP request.

Note – The value of the `User-Agent` field is the client initiating the request. The format of this field is loosely defined as *software-name/version*. See the [World Wide Web Consortium web site](#) for more information.

The `User-Agent` information is matched to browser type data defined in `amClientData.xml` located in `XXXXXX`. Based on this match, correctly formatted browser pages are sent back to the client for authentication (for example, HTML or WML pages). Once the user is authenticated, the client type is added to the session token (as the value of the key `clientType`) where it can be retrieved and used by other Federation Manager services.



Caution – If there is no matching client data, the default type is returned. Note ? The `userAgent` must be a part of the client data configured for all browser type clients. It can be a partial string or the exact product token.

Client Data

In order to detect different client types, Federation Manager needs to recognize their identifying characteristics. These characteristics are defined as a schema in the `amClientData.xml` service file. The configured Federation Manager client data available for HTML-based browsers is defined as sub-configurations of the overall schema: `genericHTML` and its parent `HTML`.

Note – Parent profiles (or styles, as they are referred to in the Federation Manager console) are defined with properties that are common to its configured child devices. This allows for the dynamic inheritance of the parent properties to the child devices making the device profiles easier to manage.

HTML

HTML is a base style containing properties common to HTML-based browsers. It might have several branches including web-based HTML (or `genericHTML`), `cHTML` (Compact HTML) and others. All configured devices for this style would inherit the following properties:

TABLE4-1

Header	Header
<code>parentId</code>	Identifies the base profile. The default value is <code>HTML</code> .
<code>clientType</code>	Arbitrary string which uniquely identifies the client. The default value is <code>HTML</code> .

TABLE 4-1 (Continued)

Header	Header
filePath	Used to locate the client type files (templates and JSP files). The default value is html.
contentType	Defines the content type of the HTTP request. The default value is text/html.
genericHTML	Client that will be treated as HTML. The default value is true. This attribute does not refer to the similarly named generic HTML style.
cookieSupport	Defines whether cookies are supported by the client browser. The default value is true which sets a cookie in the response header. The other two values could be False which sets the cookie in the URL and Null which allows for dynamic cookie detection. In the first request, the cookie is set in both the response header and the URL; the actual mode is then detected and set from the subsequent request. Note – Although the Client Detection Service supports a cookieless mode, the Federation Manager console does not. Therefore, enabling this function will not allow login to the console. This feature is provided for wireless applications and others that will support it.
CcppAccept-Charset	Defines the character encoding used by Federation Manager to send a response to the browser. The default value is UTF-8.

genericHTML

genericHTML refers to an HTML browser such as Netscape Navigator™, Microsoft™ Internet Explorer, or Mozilla. As a configured device, inherits properties from the HTML style as well as defining its own properties. genericHTML properties include the following; parentId Identifies the base profile for the configured device. The default value is HTML. clientType An arbitrary string which uniquely identifies the client. The default value is genericHTML. userAgent Search filter used to compare/match the user agent defined in the HTTP header. The default value is Mozilla/4.0. CcppAccept-Charset Defines the character encoding set supported by the browser. The default values are : UTF-8;ISO-8859-1;ISO-8859-2; ISO-8859-3;ISO-8859-4;ISO-8859-5; ISO-8859-6;ISO-8859-7;ISO-8859-8; ISO-8859-9;ISO-8859-10;ISO-8859-14; ISO-8859-15;Shift_JIS;EUC-JP; ISO-2022-JP;GB18030;GB2312;BIG5; EUC-KR;ISO-2022-KR;TIS-620;KOI8-R The character set can be configured for any given locale by adding charset_locale=codeset where the code set name is based on the Internet Assigned Numbers Authority (IANA) standard.

◆ ◆ ◆ A P P E N D I X A

Forced Reauthentication

XXXXXXNew for 7.5 need info

Index

A

Active Directory, authentication, 23-24
 amAuthUI.properties, 85
 Anonymous, authentication, 24
 arg parameter, 78-79
 authentication
 authentication level-based, 53-56
 login URLs, 54
 redirection URLs, 54-56
 login URLs
 authentication level-based-based, 54
 module-based, 57
 realm-based, 39-40
 role-based, 42-43
 service-based, 47
 user-based, 50-51
 module-based, 56-59
 login URLs, 57
 redirection URLs, 57-59
 realm-based, 38-42
 login URLs, 39-40
 redirection URLs, 40-41
 redirection URLs
 authentication level-based, 54-56
 module-based, 57-59
 realm-based, 40-41
 role-based, 43-45
 service-based, 47-49
 user-based, 51-52
 role-based, 42-46
 login URLs, 42-43
 redirection URLs, 43-45

authentication (*Continued*)

 service-based, 46-50
 login URLs, 47
 redirection URLs, 47-49
 user-based, 50-53
 login URLs, 50-51
 redirection URLs, 51-52
 user interface, 65-72
 accessing, 72-73
 user interface login URL, 73-81
 arg parameter, 78-79
 authlevel parameter, 77
 domain parameter, 76-77
 goto parameter, 74
 gotoOnFail parameter, 74-75
 IDTokenN parameter, 79-80
 iPSPCookie parameter, 79
 locale parameter, 80-81
 module parameter, 77-78
 org parameter, 76
 realm parameter, 75
 role parameter, 78
 service parameter, 78
 user parameter, 78
 authentication configuration service
 See also role-based authentication
 assign, 21-22
 authentication level-based authentication, 53-56
 login URLs, 54
 redirection URLs, 54-56
 authentication modules, 22-36
 Active Directory, 23-24

authentication modules (*Continued*)

- Anonymous, 24
- Certificate, 24-26
- Data Store, 26
- Federation, 26-27
- HTTP Basic, 27-28
- JDBC, 28
- LDAP, 29
- Membership, 29-30
- MSISDN, 30-31
- Unix, 31
- Windows Desktop SSO, 31-34
- Windows NT, 35-36

Authentication Service

- Active Directory, 23-24
- Anonymous, 24
- authentication configuration service, 21-22
- authentication modules, 22-36
- Certificate, 24-26
- configure, 19-22
- Data Store, 26
- Federation, 26-27
- general properties, 20-21
- HTTP Basic, 27-28
- JDBC, 28
- LDAP, 29
- Membership, 29-30
- MSISDN, 30-31
- Unix, 31
- Windows Desktop SSO, 31-34
- Windows NT, 35-36

Authentication Service User Interface

- button information, 85
- cascading style sheets, 70, 91
- customize, 81-94
 - adding functionality, 90
 - branding, 90-91
 - directories, 91-94
 - images, 91
 - styles, 91
- images, 70-71
- JavaScript, 70
- JavaServer Pages, 68-70
- localization files, 71-72

Authentication Service User Interface (*Continued*)

- Login.jsp, 86-88
- login screens
 - customize, 86-90
- Membership
 - customize, 88-90
- membership.jsp, 88-90
- register.jsp, 88-90
- XML files, 66-68, 82-86
- Authentication Service User Interface login
 - URL, 73-81
- authlevel parameter, 77

B

branding

- Authentication Service User Interface
 - customize, 90-91

buttons

- Authentication Service User Interface
 - customize, 85

C

Callbacks, Authentication Service User Interface, 82-86

cascading style sheets

- Authentication Service User Interface, 70, 91

Certificate, authentication, 24-26

configure, Authentication Service, 19-22

Core authentication, *See* general authentication propertiesCSS, *See* cascading style sheets

customize

- Authentication Service User Interface, 81-94
 - adding functionality, 90
 - branding, 90-91
 - buttons, 85
 - cascading style sheets, 91
 - directories, 91-94
 - images, 91
 - Login.jsp, 86-88
 - login screens, 86-90

customize, Authentication Service User Interface
(*Continued*)

- Membership, 88-90
- membership.jsp, 88-90
- register.jsp, 88-90
- styles, 91
- XML files, 82-86

Distributed Authentication User Interface, 102-104

D

Data Store, authentication, 26

directories

- Authentication Service User Interface
customize, 91-94

Directory Server documentation, 14

Distributed Authentication User Interface,
customize, 102-104

documentation

- Federation Manager, 15-16
- related, 16

domain parameter, 76-77

F

Federation, authentication, 26-27

Federation Manager documentation set, 15-16

functionality

- Authentication Service User Interface
customize, 90

G

general authentication properties, 20-21

goto parameter, 74

gotoOnFail parameter, 74-75

H

HTTP Basic, authentication, 27-28

I

IDTokenN parameter, 79-80

images

- Authentication Service User Interface, 70-71
customize, 91

instance directory, 81-94

iPSPCookie parameter, 79

J

Jato, 98

JavaScript, Authentication Service User Interface, 70

JavaServer Pages, Authentication Service User
Interface, 68-70

JDBC, authentication, 28

L

LDAP, authentication, 29

Liberty Alliance Project specifications, 14

locale parameter, 80-81

localization files, Authentication Service User
Interface, 71-72

Login.jsp

- Authentication Service User Interface
customize, 86-88

login screens

- Authentication Service User Interface
customize, 86-90
- customize, 82-86

login URLs

- authentication level-based authentication, 54
- module-based authentication, 57
- realm-based authentication, 39-40
- role-based authentication, 42-43
- service-based authentication, 47
- user-based authentication, 50-51

M

Membership

- authentication, 29-30

Membership (*Continued*)

- Authentication Service User Interface, 88-90
- membership.jsp
 - Authentication Service User Interface
 - customize, 88-90
- module-based authentication, 56-59
 - login URLs, 57
 - redirection URLs, 57-59
- module parameter, 77-78
- MSISDN, authentication, 30-31

O

- org parameter, 76

P

- parameters
 - arg parameter, 78-79
 - Authentication Service User Interface login URL, 73-81
 - authlevel parameter, 77
 - domain parameter, 76-77
 - goto parameter, 74
 - gotoOnFail parameter, 74-75
 - IDTokenN parameter, 79-80
 - iPSPCookie parameter, 79
 - locale parameter, 80-81
 - module parameter, 77-78
 - org parameter, 76
 - realm parameter, 75
 - role parameter, 78
 - service parameter, 78
 - user parameter, 78
- patches, Solaris, 18

R

- realm-based authentication, 38-42
 - login URLs, 39-40
 - redirection URLs, 40-41
- realm parameter, 75

redirection URLs

- authentication level-based authentication, 54-56
- module-based authentication, 57-59
- realm-based authentication, 40-41
- role-based authentication, 43-45
- service-based authentication, 47-49
- user-based authentication, 51-52
- register.jsp
 - Authentication Service User Interface
 - customize, 88-90
- role-based authentication, 42-46
 - See also* authentication configuration service
 - login URLs, 42-43
 - redirection URLs, 43-45
- role parameter, 78

S

- service-based authentication, 46-50
 - login URLs, 47
 - redirection URLs, 47-49
- service parameter, 78
- SJS product documentation, 16
- Solaris
 - patches, 18
 - support, 18
- staging directory, 81-94
- styles
 - Authentication Service User Interface
 - customize, 91
- Sun Java Enterprise System documentation, 14
- support, Solaris, 18

U

- Unix, authentication, 31
- User Alias List attribute, 50
- user-based authentication, 50-53
 - login URLs, 50-51
 - redirection URLs, 51-52
- user interface, 65-72
 - access, authentication, 72-73
 - accessing, 72-73

user interface (*Continued*)

- authentication, 65-72
- authentication login URL, 73-81
 - arg parameter, 78-79
 - authlevel parameter, 77
 - domain parameter, 76-77
 - goto parameter, 74
 - gotoOnFail parameter, 74-75
 - IDTokenN parameter, 79-80
 - iPSPCookie parameter, 79
 - locale parameter, 80-81
 - module parameter, 77-78
 - org parameter, 76
 - realm parameter, 75
 - role parameter, 78
 - service parameter, 78
 - user parameter, 78

user interface login URL

- arg parameter, 78-79
- authlevel parameter, 77
- domain parameter, 76-77
- goto parameter, 74
- gotoOnFail parameter, 74-75
- IDTokenN parameter, 79-80
- iPSPCookie parameter, 79
- locale parameter, 80-81
- module parameter, 77-78
- org parameter, 76
- realm parameter, 75
- role parameter, 78
- service parameter, 78
- user parameter, 78

user parameter, 78

XML files, Authentication Service User
Interface, 66-68

Z

Zero Page Login, 79-80

W

Windows Desktop SSO, authentication, 31-34

Windows NT, authentication, 35-36

X

XML, Authentication Service User Interface, 82-86

