
NetIQ® Identity Manager

User Application: Administration Guide

March 2016

Legal Notice

THIS DOCUMENT AND THE SOFTWARE DESCRIBED IN THIS DOCUMENT ARE FURNISHED UNDER AND ARE SUBJECT TO THE TERMS OF A LICENSE AGREEMENT OR A NON-DISCLOSURE AGREEMENT. EXCEPT AS EXPRESSLY SET FORTH IN SUCH LICENSE AGREEMENT OR NON-DISCLOSURE AGREEMENT, NETIQ CORPORATION PROVIDES THIS DOCUMENT AND THE SOFTWARE DESCRIBED IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. SOME STATES DO NOT ALLOW DISCLAIMERS OF EXPRESS OR IMPLIED WARRANTIES IN CERTAIN TRANSACTIONS; THEREFORE, THIS STATEMENT MAY NOT APPLY TO YOU.

For purposes of clarity, any module, adapter or other similar material ("Module") is licensed under the terms and conditions of the End User License Agreement for the applicable version of the NetIQ product or software to which it relates or interoperates with, and by accessing, copying or using a Module you agree to be bound by such terms. If you do not agree to the terms of the End User License Agreement you are not authorized to use, access or copy a Module and you must destroy all copies of the Module and contact NetIQ for further instructions.

This document and the software described in this document may not be lent, sold, or given away without the prior written permission of NetIQ Corporation, except as otherwise permitted by law. Except as expressly set forth in such license agreement or non-disclosure agreement, no part of this document or the software described in this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, or otherwise, without the prior written consent of NetIQ Corporation. Some companies, names, and data in this document are used for illustration purposes and may not represent real companies, individuals, or data.

This document could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein. These changes may be incorporated in new editions of this document. NetIQ Corporation may make improvements in or changes to the software described in this document at any time.

U.S. Government Restricted Rights: If the software and documentation are being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), in accordance with 48 C.F.R. 227.7202-4 (for Department of Defense (DOD) acquisitions) and 48 C.F.R. 2.101 and 12.212 (for non-DOD acquisitions), the government's rights in the software and documentation, including its rights to use, modify, reproduce, release, perform, display or disclose the software or documentation, will be subject in all respects to the commercial license rights and restrictions provided in the license agreement.

© 2016 NetIQ Corporation. All Rights Reserved.

For information about NetIQ trademarks, see <https://www.netiq.com/company/legal/>.

Contents

About this Book and the Library	13
About NetIQ Corporation	15
Part I Overview	17
1 Introduction to the User Application	19
1.1 About the User Application	19
1.1.1 About Identity Self-Service	20
1.1.2 About Roles-Based Provisioning	20
1.1.3 About Resource-Based Provisioning	21
1.1.4 About Workflow-Based Provisioning	23
1.1.5 About Compliance	24
1.2 User Application Architecture	25
1.2.1 User Interface	25
1.2.2 Directory Abstraction Layer	26
1.2.3 Workflow Engine	26
1.2.4 SOAP Endpoints	26
1.2.5 Application Server (J2EE-Compliant)	27
1.2.6 Database	27
1.2.7 User Application Driver	28
1.2.8 Role and Resource Service Driver	29
1.2.9 Designer for Identity Manager	30
1.2.10 iManager	30
1.2.11 Identity Manager Engine	30
1.2.12 Identity Vault	30
1.3 User Application User Types	30
1.3.1 Administrative Users	31
1.3.2 Domain Administrators, Domain Managers, and Team Managers	32
1.3.3 Designers	35
1.3.4 Business Users	35
1.4 Design and Configuration Tools	36
1.5 What's Next	37
Part II Configuring the User Application Environment	39
2 Designing the Production Environment	41
2.1 Topology	41
2.1.1 Design Constraints	41
2.1.2 High Availability Design	42
2.2 Security	43
2.2.1 Security Overview	43
2.2.2 Enabling SSL	44
2.2.3 Turning on SOAP Security	47
2.2.4 Mutual Authentication	48
2.2.5 Third-Party Authentication and Single Sign-On	48
2.2.6 Encryption of Sensitive User Application Data	48
2.2.7 Preventing XSS Attacks	49
2.2.8 Modifying the Trustee Rights for User Preferences	50
2.2.9 Modifying the Trustee Rights for a Provisioning Request Definition	51

2.2.10	Disabling the JMX Management Console on JBoss	51
2.2.11	Updating a Password for a Database User on Tomcat	52
2.3	Digital Signature Configuration	52
2.3.1	Setting Up the User Certificates	52
2.3.2	Configuring the Application Server	53
2.3.3	Configuring Logging	54
2.3.4	Configuring the User Application	54
2.3.5	Configuring the Provisioning Request Definitions	55
2.4	Performance Tuning	56
2.4.1	Logging	56
2.4.2	Identity Vault	57
2.4.3	JVM	58
2.4.4	Session Time-out Value	58
2.4.5	Tuning JBoss	59
2.4.6	Using Secure Sockets for User Application Connections to the Identity Vault	59
2.5	Clustering	60
2.6	Localizing and Customizing User Application Strings and Name Formats	61
2.6.1	Localizing User Application Strings	61
2.6.2	Customizing Text	61
2.6.3	Ensuring that Characters Display Properly in Role Report PDF Files	64
2.6.4	Ensuring that Dates Display Correctly in Norwegian	66
2.7	Configuring the Roles and Resources Tab	66
2.7.1	Role Service Driver Configuration	66
2.7.2	User Application Configuration	68
2.7.3	Security Roles	68
2.7.4	View Request Status Search Limit	69
2.7.5	Provisioning Display Settings	69
2.7.6	E-Mail Notification	69
2.7.7	Enabling Drivers for Resource Mappings	69
2.7.8	Creating a List in the RBPM Database	70
2.8	Configuring the Compliance Tab	71
2.8.1	Security Roles	71
2.8.2	E-Mail Notification	71
2.9	Configuring the Work Dashboard Tab	72
2.10	Recreating the Database After Installation	72
2.11	Changing the Default Administrator Assignments After Installation	72
2.11.1	Granting or Removing Assignments in the User Application	73
2.11.2	Changing the Assignments in Configupdate Without Removing the Existing Users	73
2.11.3	Changing the Assignments in Configupdate and Removing the Existing Users	74

3 Setting Up Logging

75

3.1	How Logging Services Help	75
3.2	What Can Be Logged	82
3.3	How Logging Works	82
3.3.1	Terminology	83
3.3.2	Components for Logging	83
3.3.3	How Logging Works	83
3.4	Types of Log Files	85
3.4.1	Difference Among Catalina, Application, and Localhost Log Files	85
3.4.2	Additional Log Files	86
3.5	Understanding the Log Format	87
3.5.1	Message Fields	87
3.5.2	Message Severity	88
3.6	Configuring Logging	89
3.6.1	Understanding Logging Configuration	90
3.6.2	Understanding the Log Level Settings	90
3.6.3	Specifying the Severity Level for Commons Logging API Loggers	91

3.6.4	Configuring Logging in User Application Administration Tab	91
3.6.5	Editing the log4j Files	92
3.6.6	Managing Log File Size	93
3.7	Configuring Logging in a Cluster	93
3.7.1	Tomcat Logging	94
3.7.2	JBoss Logging	94
3.7.3	User Application Logging	95
3.8	Logging to a Sentinel Server	95
3.8.1	Configuring the Platform Agent	95
3.8.2	Enabling Sentinel Logging	96
3.9	Logging to OpenXDAS	97
3.9.1	Using OpenXDAS with Sentinel	97
3.9.2	Enabling OpenXDAS Logging in the User Application	98
3.10	Using Log Files for Troubleshooting	98
3.11	Log Events	99

Part III Administering the User Application 105

4 Using the Administration Tab 107

4.1	About the Administration Tab	107
4.2	Who Can Use the Administration Tab	107
4.3	Accessing the Administration Tab	107
4.4	Administration Tab Actions You Can Perform	108

5 Application Configuration 109

5.1	Portal Configuration Tasks	109
5.1.1	Caching Management	109
5.1.2	Driver Status	118
5.1.3	Identity Vault Settings	118
5.1.4	Logging Configuration	119
5.1.5	Portal Settings	123
5.1.6	Theme Administration	123
5.1.7	Assigning the User Application Administrator	128
5.2	Working with the Import and Export Tools	129
5.2.1	Requirements	129
5.2.2	Restrictions	130
5.2.3	Exporting Portal Data	130
5.2.4	Importing Portal Data	130
5.3	Password Management Configuration	132
5.3.1	About Password Management Features	132
5.3.2	Configuring Challenge Response	134
5.3.3	Configuring Login	135
5.3.4	Configuring Password Sync Status	138
5.3.5	Configuring Password Hint Change	141
5.3.6	Configuring Change Password	142

6 Page Administration 145

6.1	About Page Administration	145
6.1.1	About Container Pages	145
6.1.2	About Shared Pages	147
6.1.3	An Exception to Page Usage	147
6.2	Creating and Maintaining Container Pages	147
6.2.1	Creating Container Pages	148
6.2.2	Adding Content to a Container Page	150

6.2.3	Deleting Content from a Container Page	150
6.2.4	Modifying the Layout of a Container Page	151
6.2.5	Arranging Content on the Container Page	151
6.2.6	Displaying a Container Page	152
6.3	Creating and Maintaining Shared Pages	152
6.3.1	Creating Shared Pages	153
6.3.2	Adding Content to a Shared Page	155
6.3.3	Deleting Content from a Shared Page	156
6.3.4	Modifying the Layout of a Shared Page	156
6.3.5	Arranging Content on the Shared Page	156
6.3.6	Displaying a Shared Page	157
6.4	Setting Default Pages for Groups	158
6.5	Selecting a Default Shared Page for a Container Page	158

7 RBPM Provisioning and Security Configuration 161

7.1	About RBPM Provisioning and Security Configuration	161
7.2	Provisioning Configuration	161
7.2.1	Configuring Delegation and Proxy Settings	161
7.2.2	Configuring the Digital Signature Service	163
7.2.3	Configuring the Provisioning UI Display Settings	165
7.2.4	Configuring the Workflow Engine and Cluster Settings	168
7.3	Administrator Assignments	171
7.3.1	Viewing Administrator Assignments	172
7.3.2	Creating New Assignments	172
7.3.3	Editing an Existing Assignment	178
7.3.4	Deleting Assignments	178
7.3.5	Refreshing the Assignment List	178
7.4	Team Configuration	179
7.4.1	Viewing Team Configurations	179
7.4.2	Creating New Teams	180
7.4.3	Editing an Existing Team	183
7.4.4	Deleting Teams	183
7.4.5	Refreshing the Team List	183
7.5	Navigation Access Permissions	183

Part IV Configuring and Managing Provisioning Workflows 185

8 Configuring the User Application Driver to Start Workflows 187

8.1	About the User Application Driver	187
8.2	Setting Up Workflows to Start Automatically	188
8.2.1	About Policies	188
8.2.2	Using the Policy Builder	188
8.2.3	Using the Schema Mapping Policy Editor	192

9 Managing Provisioning Request Definitions 201

9.1	About the Provisioning Request Configuration Plug-in	201
9.2	Working with the Installed Templates	202
9.3	Configuring a Provisioning Request Definition	204
9.3.1	Selecting the Driver	205
9.3.2	Deleting a Provisioning Request	205
9.3.3	Filtering the List of Requests	206
9.3.4	Changing the Status of an Existing Provisioning Request	206
9.3.5	Defining Rights on an Existing Provisioning Request	207

10 Managing Provisioning Workflows	209
10.1 About the Workflow Administration Plug-in	209
10.2 Managing Workflows	209
10.2.1 Connecting to a Workflow Server	209
10.2.2 Restricting Access to Workflows	210
10.2.3 Finding Workflows that Match Search Criteria	210
10.2.4 Controlling the Active Workflows Display	211
10.2.5 Terminating a Workflow Instance	212
10.2.6 Viewing Details about a Workflow Instance	213
10.2.7 Reassigning a Workflow Instance	213
10.2.8 Managing Workflow Processes in a Cluster	214
10.3 Configuring the Email Server	216
10.4 Working with Email Templates	216
10.4.1 Default Content and Format	218
10.4.2 Editing Email Templates	229
10.4.3 Modifying Default Values for the Template	229
10.4.4 Adding Localized Email Templates	230
10.5 Allowing a Named Password to be Retrieved over LDAP	230

Part V Web Service Reference **233**

11 Provisioning Web Service **235**

11.1 About the Provisioning Web Service	235
11.1.1 Provisioning Web Service Overview	235
11.1.2 Removing Administrator Credential Restrictions	236
11.1.3 Provisioning Web Service Method Categories	236
11.2 Developing Clients for the Provisioning Web Service	237
11.2.1 Web Access to the Provisioning Web Service	237
11.2.2 A Java Client for the Provisioning Web Service	239
11.2.3 Developing a Mono Client	244
11.2.4 Sample Ant File	246
11.2.5 Sample Log4J File	247
11.3 Provisioning Web Service API	247
11.3.1 Processes	247
11.3.2 Provisioning	257
11.3.3 Work Entries	269
11.3.4 Comments	285
11.3.5 Configuration	291
11.3.6 Miscellaneous	295
11.3.7 Cluster	298

12 Metrics Web Service **303**

12.1 About the Metrics Web Service	303
12.1.1 Web Service Semantics	304
12.1.2 Accessing the Test Page	304
12.1.3 Web Service Methods Grouped by Security Permissions	304
12.1.4 Specifying Filters	307
12.1.5 Generating the Stub Classes	309
12.1.6 Obtaining the Remote Interface	309
12.1.7 Metrics Configuration Settings	310
12.2 Metrics Web Service API	312
12.2.1 Team Manager Methods	312
12.2.2 Provisioning Application Administrator Methods	314
12.2.3 Utility Methods	316
12.3 Metrics Web Service Examples	317

12.3.1	General Examples	317
12.3.2	Other Examples	318
13	Notification Web Service	321
13.1	About the Notification Web Service	321
13.1.1	Accessing the Test Page	321
13.1.2	Accessing the WSDL	321
13.1.3	Generating the Stub Classes	322
13.2	Notification Web Service API	322
13.2.1	iRemoteNotification	322
13.2.2	BuiltInTokens	322
13.2.3	Entry	324
13.2.4	EntryArray	325
13.2.5	NotificationMap	326
13.2.6	NotificationService	326
13.2.7	StringArray	327
13.2.8	VersionVO	327
13.3	Notification Example	328
14	Directory Abstraction Layer (VDX) Web Service	331
14.1	About the Directory Abstraction Layer (VDX) Web Service	331
14.1.1	Accessing the Test Page	331
14.1.2	Accessing the WSDL	331
14.1.3	Generating the Stub Classes	332
14.1.4	Removing Administrator Credential Restrictions	332
14.2	VDX Web Service API	333
14.2.1	IRemoteVdx	333
14.2.2	Attribute	335
14.2.3	AttributeArray	337
14.2.4	AttributeType	337
14.2.5	BooleanArray	338
14.2.6	ByteArrayArray	338
14.2.7	DateArray	339
14.2.8	EntryAttributeMap	340
14.2.9	Entry	340
14.2.10	EntryArray	341
14.2.11	IntegerArray	342
14.2.12	StringArray	342
14.2.13	StringEntry	343
14.2.14	StringEntryArray	344
14.2.15	StringMap	344
14.2.16	VdxService	345
14.2.17	VersionVO	345
14.3	VDX Example	345
15	Role Web Service	357
15.1	About the Role Web Service	357
15.1.1	Accessing the Test Page	357
15.1.2	Accessing the WSDL	360
15.1.3	Generating the Stub Classes	360
15.1.4	Removing Administrator Credential Restrictions	360
15.2	Role API	361
15.2.1	IRemoteRole	361
15.2.2	Approver	375
15.2.3	ApproverArray	376

15.2.4	Category	376
15.2.5	CategoryArray	377
15.2.6	CategoryKey	378
15.2.7	CategoryKeyArray	378
15.2.8	Configuration	379
15.2.9	Container	382
15.2.10	DNString	383
15.2.11	DNStringArray	384
15.2.12	Entitlement	384
15.2.13	EntitlementArray	385
15.2.14	Group	386
15.2.15	IdentityType	387
15.2.16	IdentityTypeDnMap	389
15.2.17	IdentityTypeDnMapArray	390
15.2.18	LocalizedValue	390
15.2.19	LongArray	391
15.2.20	NrfServiceException	392
15.2.21	RequestCategoryType	392
15.2.22	RequestStatus	394
15.2.23	ResourceAssociation	396
15.2.24	Role	398
15.2.25	RoleAssignment	403
15.2.26	RoleAssignmentArray	405
15.2.27	RoleAssignmentActionType	405
15.2.28	RoleAssignmentRequest	407
15.2.29	RoleAssignmentRequestStatus	410
15.2.30	RoleAssignmentType	413
15.2.31	RoleAssignmentTypeInfo	415
15.2.32	RoleInfo	417
15.2.33	RoleInfoArray	418
15.2.34	RoleLevel	419
15.2.35	RoleLevelArray	420
15.2.36	RoleRequest	421
15.2.37	RoleServiceDelegate	424
15.2.38	RoleServiceSkeletonImpl	428
15.2.39	Sod	433
15.2.40	SodArray	435
15.2.41	SodApprovalType	436
15.2.42	SodJustification	437
15.2.43	SodJustificationArray	438
15.2.44	User	439
15.2.45	VersionVO	443
15.3	Role Web Service Examples	444
15.3.1	Retrieving Roles for a Group	444
15.3.2	Retrieving Role Assignment Request Status	445
15.3.3	Retrieving Type Information for a Role Assignment	446
15.3.4	Retrieving Role Categories	447
15.3.5	Retrieving Role Levels	448
15.3.6	Verifying Whether a User Is In a Role	449
16 Resource Web Service		451
16.1	About the Resource Web Service	451
16.1.1	Accessing the Test Page	451
16.1.2	Accessing the WSDL	452
16.1.3	Generating the Stub Classes	453
16.1.4	Removing Administrator Credential Restrictions	453
16.2	Resource Web Service Interface	454
16.2.1	IRemoteResource	454
16.2.2	CodeMapRefreshStatus	466

16.2.3	CodeMapValueStatus	468
16.2.4	EntitlementRefreshInfo	469
16.2.5	ProvisioningCodeMap	470
16.2.6	Resource	473
16.2.7	ResourceAssignment.	476
16.2.8	ResourceRequestParam	478
16.2.9	ResourceAssignmentRequestStatus	479
16.3	Resource Web Service Examples	482
16.3.1	Code Map Synchronization Code Samples	482
17 Forgot Password Web Service		485
17.1	About the Forgot Password Web Service	485
17.1.1	Accessing the Service	485
17.1.2	Accessing the WSDL	485
17.1.3	Generating the Stub Classes	486
17.2	Password Management Web Service Interface	486
17.2.1	processForgotConf	486
17.2.2	processUser.	486
17.2.3	processChaRes	487
17.2.4	processChgPwd.	488
17.3	ForgotPasswordWSBean	488
Part VI REST Services		491
18 Introduction to Resource Information Services		493
18.1	About RIS	493
18.1.1	How it Works	493
18.1.2	Configuring the RIS WAR	494
18.2	Security	496
18.2.1	Architecture	497
18.2.2	Authorization REST Service	504
18.2.3	Configuration Parameters	506
18.3	WADL Document	508
19 Identities Service		513
19.1	About the Identities Service	513
19.2	Accessing and Using the Identities Service	513
19.2.1	Available Resources	513
19.2.2	Complete URI Syntax	513
19.2.3	JSON Representations Received by the Client	515
19.2.4	Event Status Codes	518
20 Password Management Services		519
20.1	About the Password Management Service	519
20.2	Accessing and Using the Password Management Service	519
20.2.1	Available Resources	519
20.2.2	Complete URI Syntax	519
20.2.3	JSON Representations Received by the Client	521
21 Resources Service		527
21.1	About the Resources Service.	527
21.2	Accessing and Using the Resources Service	527

21.2.1	Available Resources	527
21.2.2	Complete URI Syntax	528
21.2.3	JSON Representations	529
22	Roles Service	533
22.1	About the Role Service	533
22.2	Accessing and Using the Role Service	533
22.2.1	Available Resources	533
22.2.2	Complete URI Syntax	534
22.2.3	JSON Representations	536
22.2.4	Event Status Codes	541
23	Work Items Service	543
23.1	About the Work Items Service	543
23.2	Accessing and Using the Work Items Service	543
23.2.1	Available Resources	543
23.2.2	Complete URI Syntax	544
23.2.3	JSON Representations Received by the Client	546
23.2.4	JSON Representations Sent by the Client	549
23.2.5	Event Status Codes	549
23.2.6	JSON Schema	550
24	Workflow Process and Definition Service	555
24.1	About the Workflow Process and Definition Service	555
24.2	Accessing and Using the Workflow Process and Definition Service	555
24.2.1	Available Resources	555
24.2.2	Complete URI Syntax	556
24.2.3	JSON Representations Received by the Client	559
24.2.4	JSON Representations Sent by the Client	563
24.2.5	Event Status Codes	563
24.2.6	JSON Schema	564
24.2.7	Testing the Client with the CURL Command	569
Part VII	Appendixes	571
A	Configuring the Identity Manager Approvals App	573
A.1	Product Requirements	573
A.2	Setting Up the Approvals App	574
A.2.1	Understanding Approvals App Settings	575
A.2.2	Customizing and Using the Default Approvals App Provisioning Request Definition	577
A.2.3	Creating and Deploying a Custom Configuration Link	581
A.2.4	Creating and Deploying a Custom Configuration QR Code	581
A.3	Optimizing Designer Forms for the Approvals App	582
A.4	Understanding Language Support in the Approvals App	582
B	Schema Extensions for the User Application	585
B.1	Attribute Schema Extensions	585
B.2	Objectclass Schema Extensions	588
B.3	Resource Definition Object (nrfResource)	589
B.4	Resource Request Object (nrfResourceRequest)	590
B.4.1	Resource Request Status Codes (nrfStatus)	591

B.5	Role-Resource Configuration (nrfConfiguration)	592
B.6	Resource Binding to Users (nrflidentity)	592
B.7	Resource Containers	592

C JavaScript Search API 593

C.1	Launching a Basic Search using the SearchListPortlet	593
C.1.1	Passing Request Parameters	593
C.1.2	Using a JSON-formatted String to Represent a Query	594
C.2	Creating a New Query using the JavaScript API	596
C.2.1	JavaScript API	597
C.3	Performing an Advanced Search Using a JSON-formatted Query.	599
C.4	Retrieving all Saved Queries for the Current User.	600
C.5	Running an Existing Saved Query	600
C.6	Performing a Search on All Searchable Attributes	600

D Trouble Shooting 603

D.1	Permgcn Space Error	603
D.2	Email Notification Templates	603
D.3	Org Chart and Guest Access	603
D.4	Provisioning Notification	603
D.5	javax.naming.SizeLimitExceededException	604
D.6	Linux Open Files Error	605

25 Troubleshooting 607

25.1	Using Log Files for Troubleshooting.	607
25.1.1	Virtual Data Access Logging	607
25.1.2	When a Code Map Refresh Is Triggered	610
25.1.3	When Multiple Users Try to Authenticate From Different Interfaces	611
25.1.4	When an E-Mail Approval Notification is Not Delivered	612
25.1.5	When a Role Is Requested	613
25.1.6	When a Role Is Listed in Role Catalog	615
25.1.7	Checking the Status of Database Schema Validation	618
25.1.8	Determining if Liquibase Changeset Has Executed	618
25.1.9	When Assigning a Resource to a User That Does Not Exist	620
25.1.10	When Checking the Workflow Engine Heartbeat.	620
25.2	Troubleshooting Open XDAS Logging	621
25.3	Troubleshooting Self Service Password Reset Issues.	622
25.3.1	Unable to Unlock Account through SSPR	622
25.3.2	SSPR Reports Error 5027 When Attempting to Access Configuration Manager through Internet Explorer	622
25.3.3	SSPR Reports Out of Order Page Request Error	622
25.3.4	Pressing Enter Button in SSPR's People Search Displays Locale Screen on Microsoft Internet Explorer 11 Browser.	622
25.4	Entitlement Configuration Error During Codemap Refresh	623
25.5	Mismatch of Certificates Used by Identity Manager Engine and User Application Causes Code (-9205) Error in vnd.nds.stream	623

About this Book and the Library

The *User Application: Administration Guide* describes how to administer the NetIQ Identity Manager User Application.

Intended Audience

This book provides information for individuals responsible for understanding administration concepts and implementing a secure, distributed administration model.

Other Information in the Library

For more information about the library for Identity Manager, see the [Identity Manager documentation Web site \(https://www.netiq.com/documentation/idm45/\)](https://www.netiq.com/documentation/idm45/).

About NetIQ Corporation

We are a global, enterprise software company, with a focus on the three persistent challenges in your environment: Change, complexity and risk—and how we can help you control them.

Our Viewpoint

Adapting to change and managing complexity and risk are nothing new

In fact, of all the challenges you face, these are perhaps the most prominent variables that deny you the control you need to securely measure, monitor, and manage your physical, virtual, and cloud computing environments.

Enabling critical business services, better and faster

We believe that providing as much control as possible to IT organizations is the only way to enable timelier and cost effective delivery of services. Persistent pressures like change and complexity will only continue to increase as organizations continue to change and the technologies needed to manage them become inherently more complex.

Our Philosophy

Selling intelligent solutions, not just software

In order to provide reliable control, we first make sure we understand the real-world scenarios in which IT organizations like yours operate—day in and day out. That's the only way we can develop practical, intelligent IT solutions that successfully yield proven, measurable results. And that's so much more rewarding than simply selling software.

Driving your success is our passion

We place your success at the heart of how we do business. From product inception to deployment, we understand that you need IT solutions that work well and integrate seamlessly with your existing investments; you need ongoing support and training post-deployment; and you need someone that is truly easy to work with—for a change. Ultimately, when you succeed, we all succeed.

Our Solutions

- ◆ Identity & Access Governance
- ◆ Access Management
- ◆ Security Management
- ◆ Systems & Application Management
- ◆ Workload Management
- ◆ Service Management

Contacting Sales Support

For questions about products, pricing, and capabilities, contact your local partner. If you cannot contact your partner, contact our Sales Support team.

Worldwide:	www.netiq.com/about_netiq/officelocations.asp
United States and Canada:	1-888-323-6768
Email:	info@netiq.com
Web Site:	www.netiq.com

Contacting Technical Support

For specific product issues, contact our Technical Support team.

Worldwide:	www.netiq.com/support/contactinfo.asp
North and South America:	1-713-418-5555
Europe, Middle East, and Africa:	+353 (0) 91-782 677
Email:	support@netiq.com
Web Site:	www.netiq.com/support

Contacting Documentation Support

Our goal is to provide documentation that meets your needs. The documentation for this product is available on the NetIQ Web site in HTML and PDF formats on a page that does not require you to log on. If you have suggestions for documentation improvements, click **comment on this topic** at the bottom of any page in the HTML version of the documentation posted at www.netiq.com/documentation. You can also email Documentation-Feedback@netiq.com. We value your input and look forward to hearing from you.

Contacting the Online User Community

NetIQ Communities, the NetIQ online community, is a collaborative network connecting you to your peers and NetIQ experts. By providing more immediate information, useful links to helpful resources, and access to NetIQ experts, NetIQ Communities helps ensure you are mastering the knowledge you need to realize the full potential of IT investments upon which you rely. For more information, visit community.netiq.com.

Overview

This section introduces you to the Identity Manager User Application, and helps you plan for its use in your organization.

1 Introduction to the User Application

This section introduces the Identity Manager User Application.

1.1 About the User Application

The Identity Manager User Application is a browser-based Web application that allows you to perform the following identity self-service tasks.

- ◆ Conveniently perform roles-based provisioning actions

You can map role assignments to resources within your organization, such as user accounts, computers, and databases.

For details on setting up the Roles and Resources tab, see [Section 2.7, “Configuring the Roles and Resources Tab,” on page 66](#).

- ◆ Ensure that your organization has a method for verifying that personnel are fully aware of organizational policies and are taking steps to comply with these policies.

For details on setting up the Compliance tab, see [Section 2.8, “Configuring the Compliance Tab,” on page 71](#).

- ◆ Ensure that access to corporate resources complies with organizational policies and that provisioning occurs within the context of the corporate security policy.

You can grant users access to identity data within the guidelines of corporate security policies.

For more information, see [Section 2.2, “Security,” on page 43](#).

- ◆ Reduce the administrative burden of entering, updating, and deleting user information across all systems in the enterprise.

You can create customized workflows to provide a Web-based interface for users to manipulate distributed identity data triggering workflows as necessary.

For more information, see [Part IV, “Configuring and Managing Provisioning Workflows,” on page 185](#).

- ◆ Support complex workflows and manage manual and automated provisioning of identities, services, resources, and assets.

You can implement manual provisioning by creating workflows that route provisioning requests to one or more authorities. For automated provisioning, you can configure the User Application to start workflows automatically in response to events occurring in the Identity Vault.

For more information, see [Part IV, “Configuring and Managing Provisioning Workflows,” on page 185](#).

IMPORTANT: The User Application is an application and not a framework. The areas within the User Application that are supported to be modified are outlined within the product documentation. Modifications to areas not outlined within the product documentation are not supported.

1.1.1 About Identity Self-Service

Identity is the foundation of the User Application. The application uses identity as the basis for authorizing users' access to systems, applications, and databases. Each user's unique identifier—and each user's roles—have specific access rights to identity data. For example, users who are identified as managers can access salary information about their direct reports, but not about other employees in their organization.

The **Identity Self-Service** tab within the application allows your organization to be more responsive by giving users a convenient way to view and work with identity information at any time. For example, users can use the **Identity Self-Service** tab to:

- ◆ Manage their own user accounts directly
- ◆ Look up other users and groups in the organization on demand
- ◆ Visualize how those users and groups are related
- ◆ List applications with which they are associated

The User Application Administrator sets up the contents of the **Identity Self-Service** tab, which determines what business users can see and do. A user's job requirements and level of authority also affect what is available.

NOTE: In Identity Manager 4.5 and later, if you log on as a business user, the **Identity Self-Service** tab is the only tab you will see in the User Application. If you log on as a User Application Administrator, you see the **Administration** tab as well.

1.1.2 About Roles-Based Provisioning

The purpose of the **Roles and Resources** tab within the User Application is to give you a convenient way to manage role definitions and role assignments within your organization. You can map role assignments to resources within a company, such as user accounts, computers, and databases. For example, you might use the **Roles and Resources** tab to:

- ◆ Make role requests for yourself or other users within your organization
- ◆ Create roles and role relationships within the roles hierarchy
- ◆ Create separation of duties (SoD) constraints to manage potential conflicts between role assignments
- ◆ Look at reports that provide details about the current state of the Role Catalog and the roles currently assigned to users, groups, and containers

When a role assignment request requires permission from one or more individuals in an organization, the request starts a workflow. The workflow coordinates the approvals needed to fulfill the request. Some role assignment requests require approval from a single individual; others require approval from several individuals. In some instances, a request can be fulfilled without any approvals.

When a role assignment request results in a potential SoD conflict, the initiator has the option to override the SoD constraint and provide a justification for making an exception. In some cases, a SoD conflict can cause a workflow to start. The workflow coordinates the approvals needed to allow the SoD exception to take effect.

Your workflow designer and system administrator are responsible for setting up the contents of the **Roles and Resources** tab for you and the others in your organization. The flow of control for a roles-based workflow or SoD workflow, as well as the appearance of forms, can vary depending on how the workflow designer defined the workflow's approval definition in the Designer for Identity Manager. In addition, your job requirements and level of authority determine what you can see and do.

For details on setting up the Role Subsystem, see [Section 2.7, "Configuring the Roles and Resources Tab," on page 66](#). For details on using the **Roles and Resources** tab, see the discussion of the **Roles and Resources** tab in the *Identity Manager User Application: User Guide* (<http://www.netiq.com/documentation/idm45/pdfdoc/ugpro/ugpro.pdf>).

NOTE: The ability to define custom roles is only available with Identity Manager 4.5 and later.

1.1.3 About Resource-Based Provisioning

The purpose of the resource functionality within the User Application is to give you a convenient way to manage resource definitions and resource assignments within your organization. You can map resource assignments to users or to roles within a company. For example, you might use resources to:

- ◆ Make resource requests for yourself or other users within your organization
- ◆ Create resources and map them to entitlements

When a resource assignment request requires permission from one or more individuals in an organization, the request starts a workflow. The workflow coordinates the approvals needed to fulfill the request. Some resource assignment requests require approval from a single individual; others require approval from several individuals. In some instances, a request can be fulfilled without any approvals.

The following business rules govern the behavior of resources within the User Application:

- ◆ Resources can only be assigned to a user. The resource can be granted to users in a container or group based on implicit role assignment. But the resource assignment will only be associated with a user.
- ◆ Resources can be assigned in any of the following ways:
 - ◆ Directly by a user through UI mechanisms
 - ◆ Through a provisioning request
 - ◆ Through a role request assignment
 - ◆ Through a Rest or SOAP interface
- ◆ The same resource can be granted to a user multiple times (if this capability has been enabled in the resource definition).
- ◆ A resource definition can have no more than one entitlement bound to it.
- ◆ A resource definition can have one or more same-entitlement references bound to it. This capability provides support for entitlements where the entitlement parameters represent provisionable accounts or permissions on the connected system.
- ◆ Entitlement and decision support parameters can be specified at design time (static) or at request time (dynamic).

Your workflow designer and system administrator are responsible for setting up the User Application for you and the others in your organization. The flow of control for a resource-based workflow, as well as the appearance of forms, can vary depending on how the workflow designer defined the workflow's approval definition in the Designer for Identity Manager. In addition, your job requirements and level of authority determine what you can see and do.

Resources

A **resource** is any digital entity such as a user account, computer, or database that a business user needs to be able to access. The User Application provides a convenient way for end users to request the resources they need. It also provides tools that administrators can use to define resources.

Each resource is mapped to an entitlement. A resource definition can have no more than one entitlement bound to it. A resource definition can be bound to the same entitlement more than once, with different entitlement parameters for each resource.

Resource Requests

Resources can be assigned to users only. They cannot be assigned to groups or containers. However, if a role is assigned to a group or container, the users in the group or container may automatically be granted access to the resources associated with the role.

Resource requests may require approvals. The approval process for a resource may be handled by one of the following:

- ♦ a provisioning request definition
- ♦ an external system, by setting the status code on the resource request

If a resource grant request is initiated by a role assignment, it is possible that the request will not be granted, even though the role is provisioned. The most likely reason for this would be that the necessary approvals were not provided.

A resource request can grant a resource to a user or revoke a resource from a user.

Role and Resource Service Driver

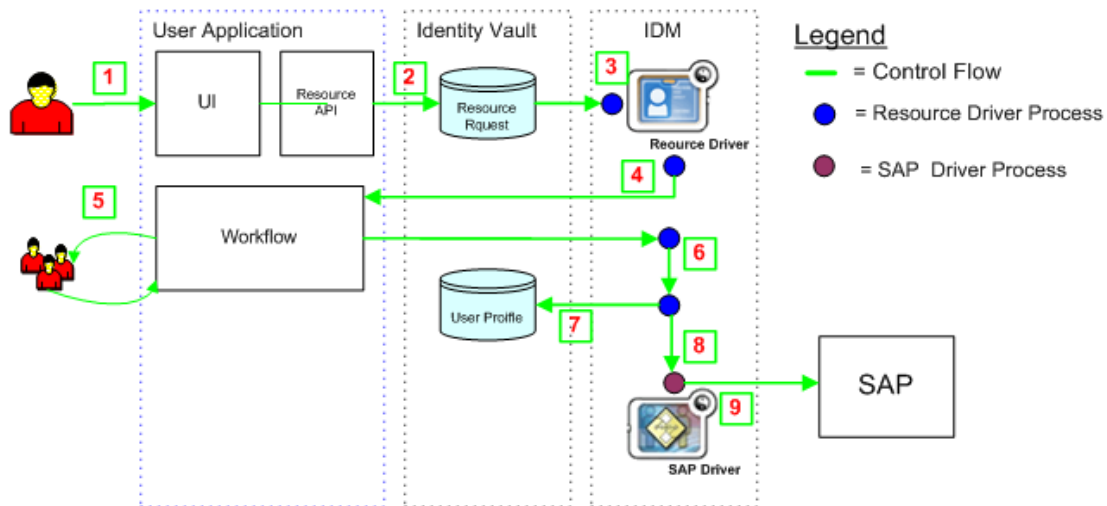
The User Application uses the Role and Resource Service Driver to manage back-end processing of resources, such as:

- ♦ managing all resource requests
- ♦ starting workflows for resource requests
- ♦ initiating the provisioning process for resource requests

Resource Request Process Flow

The following example shows the process flow for a resource assignment request. In this example, a user requests a resource that grants access to an SAP profile:

Figure 1-1 Process Flow for a Resource Request



The steps in the process are described below:

1. A user requests a resource within the User Application.
2. A User Request object is created in the Identity Vault.
3. The Role and Resource Service Driver processes the new request.
4. The Role and Resource Service Driver starts a workflow, and changes the request status.
5. The approval process is performed within the User Application. Upon completion of the approval process, the workflow activity changes the request status.
6. The Role and Resource Driver picks up the change in the status and begins to provision the resource if all of the necessary approvals have been provided.
7. The User Object attributes are updated to include the resource binding and approval information.
8. An entitlement request is made for the SAP Profile.
9. The SAP Driver processes the entitlement and creates the profile in SAP.

1.1.4 About Workflow-Based Provisioning

Workflow-based provisioning allows you to initiate workflow processes to manage the approval and revocation of user access to your organization's secure systems.

The **Work Dashboard** tab in the User Application allows users to make workflow provisioning requests. A **provisioning request** is a user or system action intended to initiate a process, and can be in the following ways:

- ♦ Directly by the user through the **Work Dashboard** tab
- ♦ Indirectly in response to events occurring in the Identity Vault

When a provisioning request requires permission from one or more individuals in an organization, the request starts one or more workflows. The workflows coordinate the approvals needed to fulfill the request. Some provisioning requests require approval from a single individual; others require approval from several individuals. In some instances, a request can be fulfilled without any approvals.

By default, the **Work Dashboard** tab does not display any provisioning requests. To configure a provisioning request, a designer familiar with your business needs creates a *provisioning request definition*, which binds the resource to a workflow.

The designer can configure workflows that proceed in one of the following ways:

- ◆ *Sequential* fashion, with each approval step being performed in order
- ◆ *Parallel* fashion, which allows more than one user to act on a workflow task concurrently

Identity Manager provides a set of Eclipse-based tools for designing the data and the flow of control within the workflows. In addition, Identity Manager provides a set of Web-based tools that allow users to view existing provisioning requests and manage workflows that are in process. For more information, see [Section 1.4, “Design and Configuration Tools,” on page 36](#).

The Provisioning Administrator is responsible for managing the workflow-based provisioning features of the User Application. For more information, see [Section 1.3, “User Application User Types,” on page 30](#).

1.1.5 About Compliance

Compliance is the process of ensuring that an organization conforms to relevant business laws and regulations. One of the key elements of compliance is attestation, which provides a method for organizations to verify that personnel are fully aware of organizational policies and are taking steps to comply with these policies. By requesting that employees or administrators regularly attest to the accuracy of data, management ensures that personnel information such as user profiles, role assignments, and approved SoD exceptions are up-to-date and in compliance.

To allow individuals within an organization to verify the accuracy of corporate data, a user makes an *attestation request*, which initiates one or more workflow processes. The *workflow processes* give the *attesters* an opportunity to attest to the correctness of the data. A separate workflow process is initiated for each attester. An attester is assigned a workflow task in the **My Tasks** list on the **Requests & Approvals** tab. To complete the workflow process, the attester opens the task, reviews the data, and attests that it is correct or incorrect.

The Roles Based Provisioning Module supports four types of attestation:

- ◆ User profile
- ◆ SoD violations
- ◆ Role assignment
- ◆ User assignment

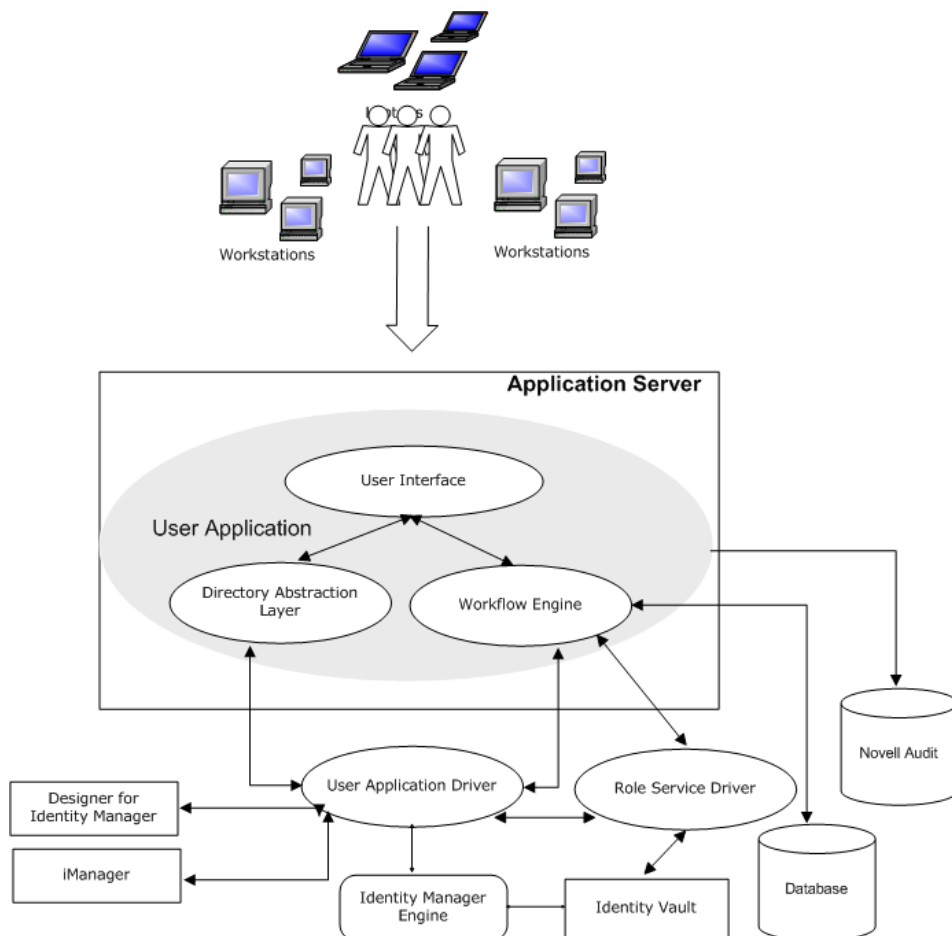
For details on setting up the **Compliance** tab, see [Section 2.8, “Configuring the Compliance Tab,” on page 71](#). For details on using the **Compliance** tab, see the discussion of the **Compliance** tab in the *Identity Manager User Application: User Guide* (<http://www.netiq.com/documentation/idm45/pdfdoc/ugpro/ugpro.pdf>).

NOTE: For compliance and attestation processes, use NetIQ Access Review instead of the User Application. Access Review enables administrators and managers to easily collect all user and access information in one central location and certify that each user has only the level of access that they need to do their job. Following the principle of least privilege, Access Review helps you ensure that your users have focused access to those applications and resources that they use and cannot access resources that they do not need to access. You can review all permissions assigned to your employees, either individually or as a group, and decide whether those permission assignments are appropriate. For more information, see the [NetIQ Access Review documentation \(https://www.netiq.com/documentation/access-review-10/\)](https://www.netiq.com/documentation/access-review-10/).

1.2 User Application Architecture

The Identity Manager User Application relies on a number of independent components acting together. The core components are shown in [Figure 1-2](#).

Figure 1-2 User Application Core Components



1.2.1 User Interface

The User Application is a browser-based Java application. The User Application framework provides container services, such as managing window state, persistence, caching, theming, logging, and acts as a security gatekeeper. The application server, on which the User Application runs, provides various services to the application as a whole, such as scalability through clustering, database access via JDBC, and support for certificate-based security.

NOTE: Support for the portal functionality in the User Application was removed in Identity Manager 4.5.

1.2.2 Directory Abstraction Layer

The directory abstraction layer provides a logical view of the Identity Vault data. You define a set of entities and their related attributes based on the Identity Vault objects that you want users to view, modify, or delete in the User Application. The Directory Abstraction layer:

- ◆ Performs all of the User Application's LDAP queries against the Identity Vault. This isolates presentation-layer logic from the Identity Vault, so that all requests for identity data go through the directory abstraction layer.
- ◆ Checks constraints and access control on data requests made via the User Application.
- ◆ Caches runtime configuration and entity-definition data obtained from the Identity Vault. See [Section 5.1.1, "Caching Management," on page 109](#)

You use the directory abstraction layer editor plug-in (available in Designer for Identity Manager) to define the structure of the directory abstraction layer data definitions. To learn more, see the section on the directory abstraction layer editor in the *Identity Manager User Application: Design Guide*.

1.2.3 Workflow Engine

The Workflow Engine is a set of Java executables responsible for managing and executing steps in an administrator-defined workflow and keeping track of state information (which is persisted in a database). When the necessary approvals have been given, the Provisioning System provisions the resource as requested.

During the course of workflow execution, the Workflow Engine can send one or more email messages to notify users of changes in the state of the workflow. In addition, it can send email messages to notify users when updates have been made to proxy, delegate, and availability settings.

You can edit an email template in the Designer for Identity Manager or in iManager and then use this template for email notifications. At runtime, the Workflow Engine retrieves the template from the directory and replaces tags with dynamic text suitable for the notification.

Additional details about the Workflow Engine, including how to configure and manage provisioning workflows, are in [Part IV, "Configuring and Managing Provisioning Workflows," on page 185](#).

1.2.4 SOAP Endpoints

The User Application provides the following SOAP endpoints to allow third-party software applications to take advantage of User Application services:

Table 1-1 SOAP Endpoints

SOAP Endpoint	Description
Provisioning Web Service	To support third-party access, the provisioning Workflow Engine includes a Web service endpoint. The endpoint offers all provisioning functionality (for example, allowing SOAP clients to start a new approval flow, or list currently executing flows).

SOAP Endpoint	Description
Metrics Web Service	The workflow engine also includes a Web Service for gathering workflow metrics. The addition of the Metrics Web Service to the Workflow Engine lets you monitor an approval flow process. In addition, it provides indicators the business manager can use to modify the process for optimal performance.
Notification Web Service	The Provisioning System includes an email notification facility that lets you send email messages to notify users of changes in the state of the provisioning system, as well as tasks that they need to perform. To support third-party access, the notification facility includes a Web service endpoint that lets you send an email message to one or more users.
Directory Abstraction Layer (VDX) Web Service	The directory abstraction layer provides a logical view of the Identity Vault data. To support access by third-party software applications, the directory abstraction layer includes a Web service endpoint called the VDX Web Service. This endpoint lets you access the attributes associated with entities defined in the directory abstraction layer. It also lets you perform ad hoc searches for entities and execute predefined searches called global queries.
Role Web Service	To support access by third-party software applications, the Role subsystem includes a Web service endpoint called the Role Web Service. It supports a wide range of role management and SoD management functions.

1.2.5 Application Server (J2EE-Compliant)

The application server provides the runtime framework in which the User Application, directory abstraction layer, and Workflow Engine execute. The User Application is packaged as a Java Web Application Archive, or WAR file. The WAR is deployed to the application server.

The User Application runs on JBoss, WebSphere, and Tomcat. For a complete list of supported platforms, see the *Roles Based Provisioning Module Installation Guide*.

1.2.6 Database

The User Application relies on a database (see the *Roles Based Provisioning Module Installation Guide* for a list of supported databases) to store the following information:

- ♦ User application configuration data, such as Web page definitions and preference values.
- ♦ Workflow state information. The actual workflow definitions are stored in the User Application driver in the Identity Vault.

1.2.7 User Application Driver

The User Application driver is responsible for:

- ♦ Storing application-specific environment configuration data.
- ♦ Notifying the directory abstraction layer when important data values change in the Identity Vault. This causes the directory abstraction layer to update its cache.

You can configure the User Application driver to:

- ♦ Allow events in the Identity Vault to trigger workflows.
- ♦ Communicate the success or failure of a workflow's provisioning activity back to the User Application database, which allows users to view the final status of their requests.
- ♦ Start workflows automatically in response to changes of attribute values in the Identity Vault.

The User Application driver is both a runtime component and a storage wrapper for directory objects (comprising the User Application's runtime artifacts).

Table 1-2 Artifacts Stored in the User Application Driver

Artifacts	Description
Driver Set Object	Every Identity Manager installation requires that drivers be grouped into driver sets. Only one driver set can be active at a time (on a given directory server). The drivers within that set can be toggled on or off individually without affecting the driver set as a whole. The User Application driver like any other Identity Manager driver, must exist inside a driver set. The driver set is not automatically created by the User Application; you must create one, then create the User Application driver within it.
User Application	The User Application driver object is the container for a variety of artifacts. The User Application driver implements Publisher and Subscriber channel objects and policies. The Publisher channel is not used by the User Application but is available for custom use cases.

Artifacts	Description
App Config Object	<p>The AppConfig object is a container for the following User Application configuration objects.</p> <ul style="list-style-type: none"> ◆ RequestDefs: Container for Provisioning Request Definitions. The definitions stored here (as XML) represent the classes of requests that end users with appropriate rights can instantiate via the User Application. ◆ WorkflowDefs: Container for Workflow objects, including design-time descriptions plus any template or unused flows. ◆ ResourceDefs: Container for Provisioned Resource definitions, including design-time descriptions plus any templates or unused targets. ◆ ServiceDefs: Container for Service Definition objects, which wrap Web Services called by workflows. ◆ DirectoryModel: Directory abstraction layer objects that represent different types of content of the Identity Vault that can be exposed in the User Application. ◆ AppDefs: Container for configuration objects that initialize the runtime environment, such as cache configuration information and email notification properties. ◆ ProxyDefs: Container for proxy definitions. ◆ DelegateeDefs: Container for delegate definitions.

1.2.8 Role and Resource Service Driver

The Roles subsystem uses the Role and Resource Service driver to perform the following functions:

- ◆ Starts an SoD workflow and waits for approvals in situations where a role request requires an SoD workflow.
- ◆ Starts a role assignment workflow and waits for approvals in situations where a role request requires a workflow.
- ◆ Adds users to and remove users from roles. To do this, the Role and Resource Service driver:
 - ◆ Waits for a start date before making assignments
 - ◆ Terminates a role assignment when the end date is reached
- ◆ Adds and removes higher-level and lower-level role relationships.
- ◆ Adds and removes role assignments for groups.
- ◆ Adds and removes role assignments for containers.
- ◆ Maintains all role membership information for indirect role assignments, including:
 - ◆ Role assignments acquired through role relationships
 - ◆ Role assignments that result from membership in groups
 - ◆ Role assignments that result from membership in containers
- ◆ Grants and revokes entitlements to and from users according to their role memberships.
- ◆ Maintains additional reporting information that is associated with each role assignment.
- ◆ Maintains additional reporting information on objects in eDirectory, such as:
 - ◆ Approval information

- ◆ Where indirect assignments come from
- ◆ Where entitlements come from
- ◆ Logs events to an auditing service.
- ◆ Cleans up processed requests after a user-specified amount of time.
- ◆ Recalculates role assignments based on dynamic and nested groups on a polled basis.

1.2.9 Designer for Identity Manager

Designer for Identity Manager provides a set of plug-ins you can use to define the directory abstraction layer objects and provisioning requests and their associated workflows. For more information, see [Section 1.4, “Design and Configuration Tools,” on page 36](#)

1.2.10 iManager

iManager provides a set of plug-ins you can use to view provisioning requests and manage their associated workflows. For more information, see [Section 1.4, “Design and Configuration Tools,” on page 36](#).

1.2.11 Identity Manager Engine

The Identity Manager engine provides the runtime framework that monitors events in the Identity Vault and connected systems. It enforces policies and routes data to and from the Identity Vault. The Identity Manager User Application is a connected system. Communication between the Identity Vault, the User Application’s directory abstraction layer, and the Workflow Engine occurs through the User Application driver.

1.2.12 Identity Vault

The Identity Vault is the repository for:

- ◆ User data
- ◆ Other identity data
- ◆ The identity Manager driver set
- ◆ The User Application driver

The User Application relies on various Identity Vault objects, so it is necessary to extend the eDirectory schema to accommodate the custom LDAP objects and attributes required by the User Application.

The schema extension occurs automatically as part of the User Application install. The custom objects and attributes are populated with default values after the User Application driver is installed and activated.

1.3 User Application User Types

The User Application users fall into these categories:

- ◆ [Administrative Users](#)
- ◆ [Domain Administrators, Domain Managers, and Team Managers](#)

- ◆ Designers
- ◆ Business Users

1.3.1 Administrative Users

The User Application defines several types of administrative users. The administrative users defined in [Table 1-3](#) are defined at installation.

Table 1-3 User Application Administrative Users

User	Description
Identity Vault Administrator	<p>A user who has rights to configure the Identity Vault. This is a logical role that can be shared with other administrative user types.</p> <p>The Identity Vault Administrator account is a proxy user for the User Application to carry out tasks on the LDAP server that an ordinary user might not have permission to execute, such as creating a new user, group, or container. It represents credentials (username and password) used to bind to the Identity Vault to perform system LDAP operations, so these are the rights that the User Application itself needs to run. The Identity Vault Administrator needs:</p> <ul style="list-style-type: none"> ◆ Supervisor rights to the User Application Driver and all the objects it contains. You can accomplish this by setting the rights at the driver container level and making them inheritable. ◆ Supervisor Entry rights to any of the users that are defined through the directory abstraction layer user entity definition. This should include Write attribute rights to objectClass and any of the attributes associated with the DirXML-EntitlementRecipient, srvprvEntityAux and srvprvUserAux auxiliary classes. ◆ Supervisor rights to the container object cn=DefaultNotificationCollection, cn=Security. This object persists email server settings used for automated provisioning emails. It can contain SecretStore credentials for authenticating to the email server itself.

User	Description
User Application Administrator	<p>A user who has the rights to perform administrative tasks for the User Application. This user can:</p> <ul style="list-style-type: none"> ◆ Use the Administration of the User Application to manage the User Application. ◆ Use iManager to administer workflow tasks (such as enabling, disabling, or terminating in-process workflows) <p>This user does not have any special privileges within the User Application.</p> <p>This user does not need any special directory rights because it controls application level access via the Administration page. Although a User Application Administrator has the ability to manage themes in the Administration page, the User Application uses the LDAP administrator credentials to modify the theme selections in the Identity Vault.</p> <p>Password self-service: One task of the User Application Administrator is to configure password self-service for the User Application. A feature of password self-service is password synchronization status. To enable the User Application Administrator to view the password synchronization status for other users (for troubleshooting or other reasons), NetIQ recommends that you create a PasswordManagement group and assign one or more users to this group. The members of this group are allowed to view the password synchronization status of other users. If you choose to create this group, it must:</p> <ul style="list-style-type: none"> ◆ Be named PasswordManagement. ◆ Be given the privileges to the Identity Vault. The group must have rights to read the user's eDirectory object attribute for users whose password synchronization status they need to view. <p>IMPORTANT: NetIQ recommends that you use NetIQ Self Service Password Reset (SSPR) for configuring passwords. It is the default password management program for Identity Manager. For more information, see "Performing Password Management" in the <i>NetIQ User Application: User Guide</i>.</p>

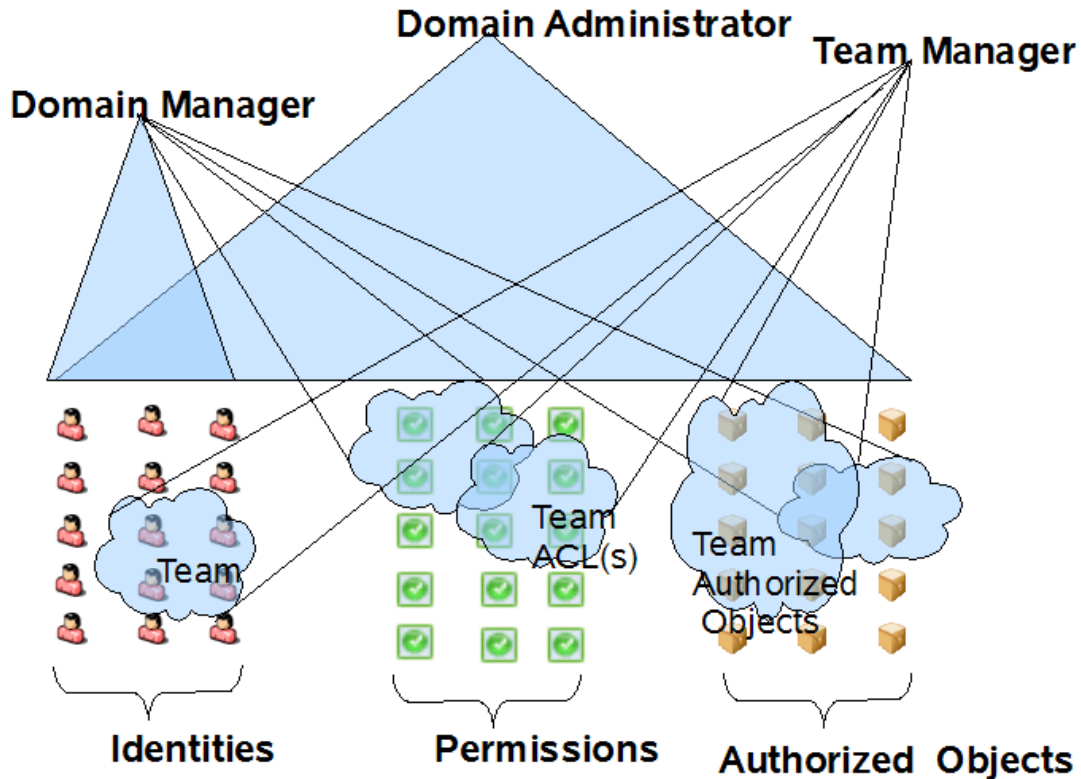
1.3.2 Domain Administrators, Domain Managers, and Team Managers

The Roles Based Provisioning Module uses a security model that recognizes three general categories of administrators and managers:

- ◆ The *Domain Administrator* is an administrator who has the full range of capabilities within a particular domain, and is able to perform all operations on all objects within the domain for all users.
- ◆ The *Domain Manager* is a delegated administrator who has the ability to perform selected operations for a subset of authorized objects within the domain for all users.
- ◆ The *Team Manager* is a business line manager who can perform selected operations for a subset of authorized objects within the domain, but only for a designated set of users (team members).

The following diagram illustrates the security model:

Figure 1-3 Security Model



Domain Administrators and Domain Managers

Domain Administrators and Domain Managers are designated through system role assignments. The Roles Based Provisioning Module allows you to assign users to any of the following system roles:

Table 1-4 System Roles for Domain Administrators and Domain Managers

Role	Description
Compliance Administrator	The Compliance Administrator is a Domain Administrator who can perform all possible actions for all objects within the Compliance domain.
Configuration Administrator	The Configuration Administrator is a Domain Administrator who can perform all possible actions on all objects within the Configuration domain. The Configuration Administrator controls access to navigation items with the Roles Based Provisioning Module. In addition, the Configuration Administrator configures the delegation and proxy service, the digital signature service, the provisioning user interface, and the workflow engine.
Provisioning Administrator	The Provisioning Administrator is a Domain Administrator who can perform all possible actions for all objects within the Provisioning domain.

Role	Description
Provisioning Manager	The Provisioning Manager is a Domain Manager who can perform only allowed actions for a subset of objects within the Provisioning domain.
Report Administrator	The Report Administrator is a Domain Administrator who can define report permissions, which include the ability to use the Identity Manager Reporting tool.
Resource Administrator	The Resource Administrator is a Domain Administrator who can perform all possible actions for all objects within the Resource domain.
Resource Manager	The Resource Manager is a Domain Manager who can perform only allowed actions for a subset of objects within the Resource domain.
Role Administrator	The Role Administrator is a Domain Administrator who can perform all possible actions for all objects (except for the System Roles) within the Role domain.
Role Manager	The Role Manager is a Domain Manager who can perform only allowed actions for a subset of objects within the Role domain.
Security Administrator	<p>The Security Administrator is a Domain Administrator who can perform all possible actions for all objects within the Security domain. The Security domain allows the Security Administrator to configure access permissions for all objects in all domains within the Roles Based Provisioning Module.</p> <p>The Security Administrator can configure teams, and also assign domain administrators, delegated administrators, and other Security Administrators.</p> <p>NOTE: For testing purposes, NetIQ does not lock down the security model in Standard Edition. Therefore, the Security Administrator is able to assign all domain administrators, delegated administrators, and also other Security Administrators. However, the use of these advanced features is not supported in production. In production environments, all administrator assignments are restricted by licensing. NetIQ collects monitoring data in the audit database to ensure that production environments comply. Furthermore, NetIQ recommends that only one user be given the permissions of the Security Administrator.</p>

These roles are initialized at install time, but can also be assigned through the **Administrator Assignments** user interface on the **Administration** tab. For details on assigning users to the system roles, see [Section 7.3, “Administrator Assignments,” on page 171](#).

In Identity Manager 4.5 Advanced Edition, you need to assign the following administrators:

- ◆ User Application Administrator
- ◆ Provisioning Administrator
- ◆ Compliance Administrator
- ◆ Roles Administrator

- ◆ Security Administrator
- ◆ Resources Administrator
- ◆ Configuration Administrator
- ◆ Report Administrator

NOTE: At install time, you can simply specify a User Application Administrator and allow all other assignments to default to this user.

Team Managers

A Team Manager is a user designated as a manager of a team through the **Team Configuration** user interface on the **Administration** tab. For details on configuring teams, see [Section 7.4, “Team Configuration,”](#) on page 179.

1.3.3 Designers

Designers use the Designer for Identity Manager to customize the User Application for your enterprise. Designer is a tool aimed at information technology professionals such as enterprise IT developers, consultants, sales engineers, architects or system designers, and system administrators who have a strong understanding of directories, databases, and their information environment and who act in the role of a designer or architect of identity-based solutions.

To create or edit workflow objects in Designer, the user needs the following rights on the RequestDefs.AppConfig container for the specific User Application driver.

- ◆ [Entry Rights] Supervisor or Create
- ◆ [All Attribute Rights] Supervisor or Write

To initiate a workflow, the user must have Browse [Entry Rights] on the RequestDefs.AppConfig container for the specific User Application driver or individually per request definition object if you are using a delegated model.

1.3.4 Business Users

Business users interact with the User Application’s **Identity Self-Service**, **Work Dashboard**, and **Roles and Resources** tabs. A business user is an *authenticated user*, such as an employee, a manager, or a delegate or proxy for an employee or manager. A *delegate user* is a user to whom one or more specific tasks appropriate to that user’s rights can be delegated, so that the delegate can work on those specific tasks on behalf of someone else. A *proxy user* is user who acts in the role of another user by temporarily assuming that user’s identity. All of the rights of the original user apply to the proxy. Work owned by the original user continues to be owned by that user.

The user’s capabilities within the User Application depend on what features of the User Application Administrator has enabled for them. They can be allowed to:

- ◆ View and edit user information, with appropriate rights
- ◆ Search for users or resources using advanced search criteria, which can be saved for later reuse
- ◆ Recover forgotten passwords

The User Application can be configured so that users can:

- ◆ Request a resource (start one of potentially many predefined workflows)

- ◆ View the status of previous requests
- ◆ Claim tasks and view tasklists (by resource, recipient, or other characteristics)
- ◆ View proxy assignments
- ◆ View delegate assignments
- ◆ Specify one's availability
- ◆ Enter proxy mode to claim tasks on behalf of another
- ◆ View team tasks and request team resources

1.4 Design and Configuration Tools

The various administrators can use the following tools to design and configure the Identity Manager User Application.

Table 1-5 Tools for Designing and Configuring the User Application

Tool	Purpose
Designer for Identity Manager	<p>A powerful, graphical toolset for configuring and deploying Identity Manager. The following plug-ins are designed to help you configure the User Application:</p> <ul style="list-style-type: none"> ◆ Directory Abstraction Layer editor: Lets you define the Identity Vault objects for your User Application. ◆ Provisioning Request Definition editor: Lets you create workflows for provisioning request definitions. Also allows you to customize the forms by which users make and approve requests and email templates. ◆ Provisioning view: Lets you import, export, deploy, and migrate directory abstraction layer and provisioning requests to the User Application driver. ◆ Role editor: Lets you create and configure roles for use within the User Application. ◆ Resource editor: Lets you create and configure resources for use within the User Application.

For more information, see the *Identity Manager User Application: Design Guide*.

Tool	Purpose
iManager	<p>A Web-based administration console. The following plug-ins are designed to help you configure and administer the User Application:</p> <ul style="list-style-type: none"> ◆ Provisioning Request Configuration plug-in: Provides a read-only view of provisioning request definitions created through Designer and allows you to mark them active or inactive. ◆ Workflow Administration plug-in: Provides a browser-based interface that lets you view the status of workflow processes, reassign activities within a workflow, or terminate a workflow in the event that it is stopped and cannot be restarted. ◆ Provisioning Team plug-in: Not supported with this release of the Roles Based Provisioning Module. The Team Configuration user interface on the Administration of the User Application replaces this iManager tool. ◆ Provisioning Team Request plug-in: Not supported with this release of the Roles Based Provisioning Module. The Team Configuration user interface on the Administration of the User Application replaces this iManager tool. <p>For more information, see Part IV, “Configuring and Managing Provisioning Workflows,” on page 185</p>
User Application Administration tab	<p>A Web-based administration console that allows you to configure, manage, and customize the User Application.</p> <p>For more information, see Part III, “Administering the User Application,” on page 105.</p>

1.5 What’s Next

Now that you have learned about the features and architecture of the Identity Manager User Application, you can start to customizing it as needed for your own business needs. Typically you will,

- ◆ Customize the user interface and identity self-service features. See [Table 1-6 on page 37](#).
- ◆ Set up the requests and approval features (if provisioning is installed). See [Table 1-7 on page 38](#).
- ◆ Set up your production environment. See [Table 1-8 on page 38](#).

Table 1-6 Customizing the User Interface and Identity Self-Service Features

To learn about	See
Setting up directory abstraction layer objects	<i>Identity Manager User Application: Design Guide</i>
Adding new pages and setting page security	Chapter 6, “Page Administration,” on page 145
Changing the User Application’s theme or branding	Section 5.1.6, “Theme Administration,” on page 123

To learn about	See
Localizing the User Application user interface	Section 2.6, "Localizing and Customizing User Application Strings and Name Formats," on page 61
Enabling password self-service	Section 5.3, "Password Management Configuration," on page 132

Table 1-7 *Setting Up the Provisioning, Roles, and Resources Features*

To learn about	See
Creating provisioning requests	Identity Manager User Application: Design Guide and Chapter 9, "Managing Provisioning Request Definitions," on page 201
Customizing request and approval forms	Identity Manager User Application: Design Guide
Defining administrator assignments	Section 7.3, "Administrator Assignments," on page 171
Defining teams	Section 7.4, "Team Configuration," on page 179
Defining navigation access permissions	Section 7.5, "Navigation Access Permissions," on page 183
Defining email templates	Identity Manager User Application: Design Guide and Section 10.4, "Working with Email Templates," on page 216

Table 1-8 *Setting Up the User Application Production Environment*

To learn about	See
Your production environment topology	Section 2.1, "Topology," on page 41
Setting up security	Section 2.2, "Security," on page 43
Performance tuning strategies	Section 2.4, "Performance Tuning," on page 56
Setting up a cluster	Section 2.5, "Clustering," on page 60
Setting up logging	Chapter 3, "Setting Up Logging," on page 75

|| **Configuring the User Application Environment**

These sections describe how to configure various aspects of the Identity Manager User Application environment to meet the needs of your organization.

2 Designing the Production Environment

This section discusses issues relating to setting up a production environment. It provides guidance on a number of considerations that come into play when making the transition from a sandbox, test, or other pre-production environment to a production environment.

2.1 Topology

Each major subsystem can have many instances and many ways of connecting. Not every possible layout is supported.

2.1.1 Design Constraints

Audit Server: This application is responsible for capturing event information (and possibly a good deal of other information) from the User Application environment at runtime. It might also be doing double duty as a persistence store for other applications in your company. For a variety of reasons, you must never put other major pieces of the Identity Manager system (for example, the application server or the Identity Vault) on the same machine as the Audit server.

Identity Vault: This is a heavily trafficked component with a need for good performance and good scalability. You must put the Identity Vault on a dedicated machine. You should never put another high-traffic system, such as an application server with a deployment of the User Application, on the same machine as the Identity Vault.

Database: If this instance of a supported database is also your auditing database, it is probably on a dedicated machine. The User Application uses this component in the following ways:

- ♦ As a persistence store for portal configuration data
- ♦ As the persistence store for state information on in-process workflows

Application Server: For performance and capacity reasons, you must run this piece on a dedicated machine.

These considerations require at a minimum a three-machine configuration.

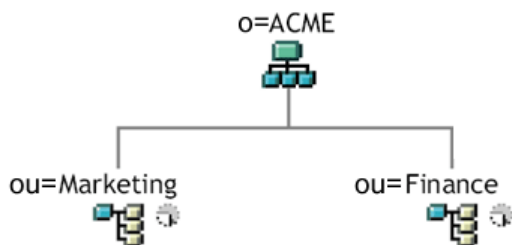
Additional Constraints The following additional architectural constraints apply to any User Application configuration:

- ♦ No User Application instance can service (search, query, add users to, and so forth) more than one user container. Also, a user container association with an application is meant to be permanent.
- ♦ No User Application driver can be associated with more than one User Application, except when the User Applications are installed on sister nodes of the same JBoss or Websphere cluster. In other words, a one-to-many mapping of drivers to User Applications is not supported.

The first constraint enforces a high degree of encapsulation in User Application design.

Suppose you have the following organizational structure:

Figure 2-1 Sample Organizational Structure



During installation of the User Application, you are asked to specify the top-level user container that your installation looks for in the Identity Vault. In this case, you could specify `ou=Marketing,o=ACME` or (alternatively) `ou=Finance,o=ACME`. You cannot specify both. All User Application searches and queries (and administrator log-ins) are scoped to whichever container you specify.

NOTE: In theory, you could specify a scope of `o=ACME` in order to encompass Marketing and Finance. But in a large organization, with potentially many `ou` containers (rather than just two relating to Marketing and Finance), this is not likely to be practical.

It is possible, of course, to create two independent installations of the User Application (sharing no resources in common), one for Marketing and another for Finance. Each installation would have its own database, its own appropriately configured User Application driver, and each User Application would be administered separately, possibly having unique themes.

If you truly need to place Marketing and Finance within the same scope for one User Application installation, there are two possible tactics to consider. One is to insert a new container object (for example, `ou=MarketingAndFinance`) in the hierarchy, above the two sibling nodes; then point to the new container as the scope root. Another tactic is to create a filtered replica (a special type of eDirectory tree) that combines the needed parts of the original ACME tree, and point the User Application at the replica's `root` container. For more information about filtered replicas, see the [eDirectory Administration Guide \(https://www.netiq.com/documentation/edir88/edir88/data/a2iii88.html\)](https://www.netiq.com/documentation/edir88/edir88/data/a2iii88.html).

If you have questions about a particular system layout, contact your NetIQ representative for assistance or advice.

2.1.2 High Availability Design

Clustering for high availability and capacity is discussed in [Section 2.5, “Clustering,” on page 60](#). For now, you should know that:

- ♦ High availability of the User Application is available through clustering. You can set up a cluster so that each node runs one User Application instance. The instances are all coequals (peers).
- ♦ Automatic failover is supported. An interrupted workflow can resume after the loss of a cluster node.

See [Section 2.5, “Clustering,” on page 60](#) for more information.

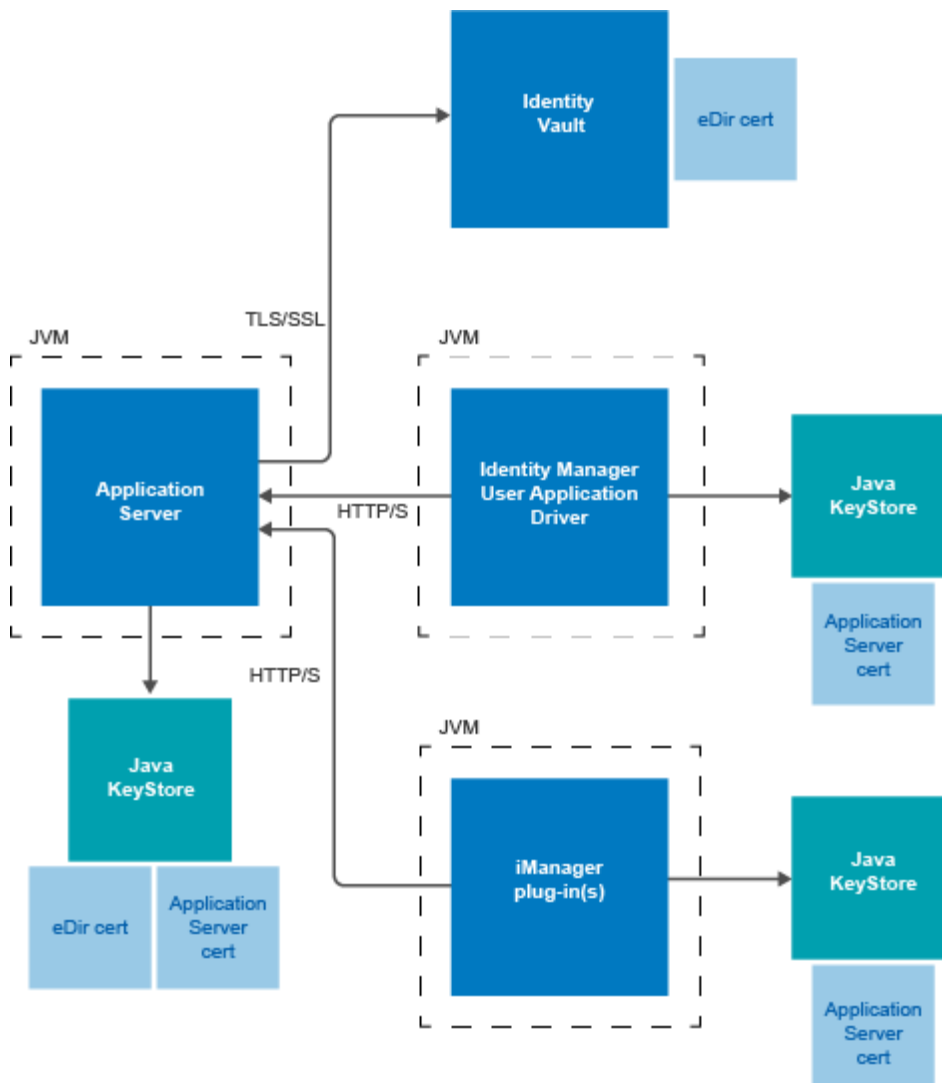
2.2 Security

2.2.1 Security Overview

Moving from pre-production to production usually involves hardening the security aspects of the system. In sandbox testing, you might use regular HTTP to connect the User Application driver to the application server, or you might use a self-signed certificate (as a temporary measure) for driver/app-server communication. In production, on the other hand, you probably use secure connections, with server authentication based on your company's Verisign* (or other trusted provider) certificate.

It is typical for X.509 certificates to be used in a variety of places in the Identity Manager User Application environment, as shown in the following diagram.

Figure 2-2 Identity Manager User Application Environment



All communication between the User Application and the Identity Vault is secure, using Transport Layer Security, by default. The installation of the Identity Vault (eDirectory) certificate into the Tomcat application server keystore is done automatically at install time. Unless you specify otherwise, the User Application installer places a copy of the eDirectory certificate in the JRE's default *cacerts* store.

The installation and configuration process also handles the certificate for a JBoss application server. However, you must manually install the certificate into the WebSphere application server by using the IBM tools. For more information, see the [NetIQ Identity Manager Setup Guide](#).

The server certificate needs to be in several places, if communications are to be secure, as shown in the diagram. Different setup steps might be needed depending on whether you intend to use a self-signed certificate in the various places in the diagram shown with a *JBoss cert* box, or you intend to use a certificate issued by a trusted certificate authority (CA) such as Verisign.

2.2.2 Enabling SSL

The User Application uses HTML forms for authentication. As a result, user credentials are exposed during login. We strongly recommend that you enable SSL to protect sensitive information.

The procedure for enabling SSL varies depending on whether you are working in a test or production environment, as described below.

Enabling SSL in a Test Environment

If you are in a test environment, you might want to use a self-signed certificate. The procedure below explains how to do this.

To enable SSL in a test environment:

- 1 Export the Certificate Authority from your eDirectory server using iManager:
 - 1a Go to iManager.
 - 1b Login with the eDirectory administrator's username and password.
 - 1c Go to **Administration > Modify Object**.
 - 1d Browse to the CA object in the Security container called *TreeName* CA.Security. For example, IDMTTESTTREE CA.Security.
 - 1e Click **OK**.
 - 1f Click **Certificates > Self Signed Certificate**.
 - 1g Select the self-signed certificate you want to use.
 - 1h Click the **Export** button.
 - 1i Clear **Export private key**.
 - 1j Click **Export format** and select **DER**.
 - 1k Click **Next**.
 - 1l Click **Save the exported certificate**.
 - 1m Click **Save File**. iManager saves the file as `cert.der`.
 - 1n Click **Close**.
 - 1o Move the saved file to a location where you want to store the exported certificate.
- 2 Create a keystore:

In a command prompt, cd to your `.../tomcat/conf` directory and create the keystore.

NOTE: The `tomcat/conf` path is the default path for a User Application installed on Tomcat. The path can vary, depending on how you installed the User Application and Tomcat.

To create the keystore, use the following command:

```
keytool -genkey -alias [keystore name] -keyalg RSA -keystore [your keystore name.keystore] -validity 3650
```

You will be prompted for your password, first and last name, and possibly other pieces of information.

Here are a few important points to keep in mind as you answer the prompts:

- ◆ When asked for your first and last name, you should supply the fully qualified name of the server. For example:

```
MyJBossServer.NetIQ.com
```

- ◆ Be sure your spelling is correct. If you spell any words incorrectly, you will see errors when you generate your signed certificate from the signing authority.
- ◆ Save a copy of the information you provided in a simple text file. This will help to ensure that you supply the same information when you apply to the signing authority and when you import your certificate.

- 3 In your JBoss `conf` directory, create a simple text file to store your keystore `.csr` file. Once this file is created, return to a command prompt and create the `.csr` with the following command:

```
keytool -certreq -v -alias [Keystore name you used when you created your keystore] -file [your.csr] -keypass [password you created in keystore] -keystore [your.keystore] -storepass [your password]
```

- 4 Issue a certificate using iManager:

4a Go to **Certificate Server > Issue Certificate**.

4b Browse to the `.csr` file created earlier.

Click **Next**. Then click **Next** again.

4c Select Unspecified as the certificate type.

Click **Next**. Then click **Next** again.

Enabling SSL in a Production Environment

To install a signed certificate into the JBoss Application Server with the User Application:

- 1 Create a keystore using the `keytool` utility included in the JRE.

In a command prompt, `cd` to your `.../tomcat/conf` directory and create the keystore.

NOTE: The `tomcat/conf` path is the default path for a User Application installed on Tomcat. The path can vary, depending on how you installed the User Application and Tomcat.

```
keytool -genkey -alias [keystore name] -keyalg RSA -keystore [your keystore name.keystore] -validity 3650
```

You will be prompted for your password, first and last name, and possibly other pieces of information.

Here are a few important points to keep in mind as you answer the prompts:

- ◆ When asked for your first and last name, you should supply the fully qualified name of the server (for example, `MyJBossServer.NetIQ.com`).

- ♦ Be sure your spelling is correct. If you spell any words incorrectly, you will see errors when you generate your signed certificate from the signing authority.
 - ♦ Save a copy of the information you provided in a simple text file. This will help to ensure that you supply the same information when you apply to the signing authority and when you import your certificate.
- 2 In your JBoss `conf` directory, create a simple text file to store your keystore `.csr` file. Once this file is created, return to a command prompt and create the `.csr` with the following command:

```
keytool -certreq -v -alias [Keystore name you used when you created your keystore] -file [your.csr] -keypass [password you created in keystore] -keystore [your.keystore] -storepass [your password]
```

- 3 Submit your `.csr` file to your Certificate Authority (CA), such as VeriSign or Entrust.

Once you have received your signed certificate from your CA, you need to import it into your Tomcat Server.

To import your signed certificate:

- 1 Place a copy of your certificate in your Tomcat `conf` directory.
Be sure to create a backup copy of this certificate and store it in a safe location.
- 2 Convert the root CA to DER format:
 - 2a Double-click on your certificate stored in the Tomcat `conf` directory.
This will open a pop-up Certificate dialog screen.
 - 2b Click on the **Certificate Path** tab.
 - 2c Highlight the root certificate (the certificated issue by the signing authority, such as Entrust or Verisign).
 - 2d Click on **View Certificate**. This will open a new Certificate dialog for the root certificate.
 - 2e Click on the **Details** tab.
 - 2f Click **copy to file**. This will open the Export Certificate Wizard.
 - 2g Click **next** when the Export Certificate Wizard opens.
 - 2h Select **DER encoded binary for X.509 (.CER)** and click **next**.
 - 2i Create a new file to store the newly formatted certificate and store it in your Tomcat `conf` directory.
Then click **Finish**.
- 3 Convert the signed certificate into DER format:
 - 3a Double click on your certificate, which should be stored in the Tomcat `conf` directory.
This will open a pop-up Certificate dialog screen.
 - 3b Click on the **Details** tab.
 - 3c Click **copy to file**. This will open the Export Certificate Wizard.
 - 3d Click **next** when the Export Certificate Wizard opens.
 - 3e Select **DER encoded binary for X.509 (.CER)** and click **next**.
 - 3f Create a new file to store the newly formatted certificate and store it in your Tomcat `conf` directory.
Then click **Finish**.
- 4 Open a command prompt and `cd` to your Tomcat `conf` directory.
- 5 Import your Root CA:

```
keytool -import -trustcacerts -alias root -keystore your.keystore -file
yourRootCA.cer
```

Be sure to specify **root** as your alias in this step.

If all goes well, you should see a **Certificate was added to keystore** message.

6 Import your signed certificate.

```
keytool -import -alias [alias you used when creating the .csr] -keystore
[your.keystore] -file [your DER converted Signed Cert.cer]
```

If all goes well, you should see a **Certificate reply was installed in keystore** message.

7 To verify that the signed certificate was imported correctly, you can run the following command in a command prompt from your Tomcat `conf` directory.

```
keytool -list -v -alias idm -keystore idm.keystore
```

You should see your self signed and signed certificates listed in the output.

8 Enable SSL in Tomcat.

Locate `server.xml` under `.../tomcat/conf` directory and open that file in a text editor. Enable SSL by uncommenting or adding the following section:

```
maxThreads="150" strategy="ms" maxHttpHeaderSize="8192"
emptySessionPath="true"
scheme="https" secure="true" clientAuth="false"
keystoreFile="${tomcat.server.home.dir}/conf/tomcat.jks"
keystorePass="changeit" sslProtocol="TLS" />
```

NOTE: Remember to point `keystoreFile` to the keystore you created. For example: `${tomcat.server.home.dir}/conf/server.keystore`. Also, remember to change the `keystorePass="changeit"` to your keystore password.

You may also need to add `SSLEnabled="true" protocol="HTTP/1.1"`, as shown below:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
    maxThreads="150" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS"
    keystoreFile="/opt/certs/chap8.keystore"
    keystorePass="changeit" />
```

9 Restart your Tomcat server and test.

2.2.3 Turning on SOAP Security

- 1 In `IDMProv.war`, find the `web.xml` file and open it in a text editor.
- 2 At the bottom of the file, uncomment the following section:

```

<security-constraint>
  <web-resource-collection>
    <web-resource-name>IDMProv</web-resource-name>
    <description>IDM Provisioning Edition</description>
    <url-pattern>/*</url-pattern>
    <http-method>POST</http-method>
    <http-method>GET</http-method>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>

```

3 Save the file and archive, then restart Tomcat.

2.2.4 Mutual Authentication

The Identity Manager User Application does not support client certificate-based authentication out of the box. That functionality can be obtained, however, by using NetIQ Access Manager. See your NetIQ representative for more information. See also [Section 2.2.5, “Third-Party Authentication and Single Sign-On,” on page 48](#).

2.2.5 Third-Party Authentication and Single Sign-On

You can configure Identity Manager to work with NetIQ Access Manager using SAML 2.0 authentication. This capability enables using a non-password-based technology to log in to the identity applications through Access Manager. For example, users can log in through a user (client) certificate, such as from a smart card.

Access Manager interacts with One SSO Provider (OSP) in Identity Manager to map the user to a DN in the Identity Vault. When a user logs in to the identity applications through Access Manager, Access Manager can inject a SAML assertion (with the user’s DN as the identifier) into an HTTP header and forwards the request to the identity applications. The identity applications uses the SAML assertion to establish the LDAP connection with the Identity Vault. For information on configuring Access Manager to support this capability, refer to the Access Manager documentation.

Accessory portlets that allow single sign-on authentication based on passwords do not support single sign-on when SAML assertions are used for identity application authentication.

For more information about configuring Identity Manager to work with Access Manager, see [“Using SAML Authentication with NetIQ Access Manager for Single Sign-on”](#).

2.2.6 Encryption of Sensitive User Application Data

Any sensitive information associated with the User Application that is stored persistently is encrypted by using the symmetric algorithm AES-128. The master key itself is protected by password-based cryptography using PBESWithSHA1AndDESede. The password is never persisted or stored out of memory.

Information that is encrypted includes (but is not limited to):

- ♦ LDAP administrator user password
- ♦ LDAP guest user password
- ♦ DSS trusted CA keystore password

- ◆ DSS signature key keystore password
- ◆ DSS signature key entry password

However, in a cluster environment, if session failover is enabled, some sensitive data (for example, a login-password for single sign-on) in the user session can be transferred on the network during session replication. This can expose sensitive data to network sniffers. To protect this sensitive data, do one of the following:

- ◆ Enable encryption for JGroups. Perform the following steps to enable encryption for JGroups:
 1. Log in to the User Application GUI.
 2. Click **Application > Application Configuration > Cluster and Cache Configuration**.
 3. Under **Cluster Configuration**, navigate to **Cluster Properties** and add Global values from Default to JGroups encryption.

The default cache setting properties are:

```
UDP(mcast_addr=228.8.8.8;mcast_port=45654):PING(timeout=5000):FD(timeout=10000;max_tries=5):VERIFY_SUSPECT:pbcast.NAKACK:UNICAST:pbcast.STABLE:FRAG:pbcast.GMS
```

- ◆ For UDP, change the default cache setting properties to:

```
UDP(mcast_addr=228.8.8.8;mcast_port=45654;tos=8;ucast_rcv_buf_size=2000000;ucast_send_buf_size=640000;mcast_rcv_buf_size=25000000;mcast_send_buf_size=640000;loopback=false;discard_incompatible_packets=true;enable_bundling=false;max_bundle_size=64000;max_bundle_timeout=30;use_incoming_packet_handler=true;use_outgoing_packet_handler=false;ip_ttl=2):PING(timeout=5000):FD(timeout=10000;max_tries=5):ENCRYPT(encrypt_entire_message=true;sym_init=128;sym_algorithm=AES/ECB/PKCS5Padding;asym_init=512;asym_algorithm=RSA):VERIFY_SUSPECT:pbcast.NAKACK:UNICAST:pbcast.STABLE:FRAG:pbcast.GMS
```

- ◆ For TCP, change the default cache setting properties to:

```
TCP(bind_addr=<node1_ip_address>;start_port=7815;loopback=true):TCPPING(initial_hosts=<node1_ip_address>[7815],<node2_ip_address>[7815];port_range=3;timeout=3500;num_initial_members=3;up_thread=true;down_thread=true):MERGE2(min_interval=5000;max_interval=10000):FD(shun=true;timeout=2500;max_tries=5;up_thread=true;down_thread=true):ENCRYPT(encrypt_entire_message=true;sym_init=128;sym_algorithm=AES/ECB/PKCS5Padding;asym_init=512;asym_algorithm=RSA):VERIFY_SUSPECT(timeout=1500;down_thread=false;up_thread=false):pbcast.NAKACK(down_thread=true;up_thread=true;gc_lag=100;retransmit_timeout=3000):pbcast.STABLE(desired_avg_gossip=20000;down_thread=false;up_thread=false):pbcast.GMS(join_timeout=5000;join_retry_timeout=2000;shun=false;print_local_addr=true;down_thread=true;up_thread=true):pbcast.STATE_TRANSFER(up_thread=true;down_thread=true)
```

- ◆ Make sure that the cluster is behind a firewall.

2.2.7 Preventing XSS Attacks

The User Application supports the concept of XSS (Cross-Site Scripting) blacklists to allow you to prevent scripting attacks. The XSS blacklists prevent XSS injection in the free text input fields within the Detail portlet, approval flow, and role assignments pages within the application.

The User Application provides default values for two blacklists, one for the Detail Portlet, and one for the workflow system (which handles the approval flow and role assignments pages). However, you can customize the blacklists to suit the requirements of your environment.

To customize the either of the blacklists, you need to enter the words or characters you want to block in the `sys-configuration-xml\data.xml` file. In Tomcat, you can find this file in the `<tomcat_home>/conf` folder. Open the file with a UTF-8 friendly editor.

To modify the blacklist for the Detail portlet, open `<tomcat_home>/conf/ism-configuration.properties` in a UTF-8 editor, and find the `com.netiq.xss.blacklist.detailportlet` property:

```
<property>
  <key>com.netiq.xss.blacklist.detailportlet</key>
  <value>...</value>
</property>
```

The text node of `<value>` is the blacklist for Detail portlet. The blocked words are separated by comma (for example, `blocked_word1,blocked_word2,...`). The default setting is:

```
",&lt;
```

This means that double quote and `<` are disallowed.

To modify the blacklist for the approval flow and role assignments pages, locate the `com.netiq.xss.blacklist.workflow` property.

```
<property>
  <key>com.netiq.xss.blacklist.workflow</key>
  <value>...</value>
</property>
```

The syntax is the same. The default value is:


```
&lt;
```

which means that `<` is disallowed.

If you decide to customize the blacklists, be careful not to remove the default values. If you remove these values, you will make the lists less restricted, and therefore increase the risk of XSS attacks.

2.2.8 Modifying the Trustee Rights for User Preferences

To allow user preferences to be saved, the administrator must ensure that the permissions on the `srvprvUserPrefsPlus` and `srvprvQueryList` attributes are set so that the user is able to write to these attributes. The necessary rights should be set for [This] at the tree root level, since [This] is a special alias to the object itself, causing only the user to have rights to modify its own preferences. To set the proper permissions, the administrator needs to modify the trustees for these attributes in iManager, as shown below:

 **Modify Trustees** ?

Object name: .THOMASVMTREE.
Trustee name: [This]

Property Name	Assigned Rights	Inherit
<input type="checkbox"/> nsimHint	<input type="checkbox"/> Supervisor <input type="checkbox"/> Compare <input type="checkbox"/> Read <input checked="" type="checkbox"/> Write <input checked="" type="checkbox"/> Self <input type="checkbox"/> Dynamic <input type="checkbox"/> Nested	<input checked="" type="checkbox"/>
<input type="checkbox"/> nsimPasswordReminder	<input type="checkbox"/> Supervisor <input type="checkbox"/> Compare <input type="checkbox"/> Read <input checked="" type="checkbox"/> Write <input checked="" type="checkbox"/> Self <input type="checkbox"/> Dynamic <input type="checkbox"/> Nested	<input checked="" type="checkbox"/>
<input type="checkbox"/> srvprvQueryList	<input type="checkbox"/> Supervisor <input checked="" type="checkbox"/> Compare <input checked="" type="checkbox"/> Read <input checked="" type="checkbox"/> Write <input type="checkbox"/> Self <input type="checkbox"/> Dynamic <input type="checkbox"/> Nested	<input checked="" type="checkbox"/>
<input type="checkbox"/> srvprvUserPrefsPlus	<input type="checkbox"/> Supervisor <input checked="" type="checkbox"/> Compare <input checked="" type="checkbox"/> Read <input checked="" type="checkbox"/> Write <input type="checkbox"/> Self <input type="checkbox"/> Dynamic <input type="checkbox"/> Nested	<input checked="" type="checkbox"/>

The `srvprvUserPrefsPlus` property has no space limitations, so it can save a large amount of user preference information. If you have used the `srvprfUserPrefs` property in a previous release, this property will be migrated to `srvprvUserPrefsPlus` the first time a user saves new preferences in the User Application.

2.2.9 Modifying the Trustee Rights for a Provisioning Request Definition

To view the details and comments associated with a task in the **Task Notifications** section of the Work Dashboard tab, the Domain Administrator or Delegated Administrator must have the proper rights to the provisioning request definition. In particular, the user must have the `nrfAccessMgrTaskAddressee` right to the provisioning request definition, with write access enabled. To set the proper permissions, the administrator needs to modify the trustees for the provisioning request definition, as described below:

- 1 Log into iManager as an administrator.
- 2 Select **Modify Trustee** from the **Rights** left-navigation menu.
- 3 Browse to the provisioning request definition.
- 4 If necessary, click **Add Trustee** to add the user.
- 5 Click on the **Assign rights** link.

Notice that `nrfAccessMgrTaskAddressee` is not listed with the write permission checked, which means that the user does not have the proper rights for the provisioning request definition.
- 6 Click the **Add Property** button.
- 7 Check the check box for **Show all properties in schema**.
- 8 Select `nrfAccessMgrTaskAddresses`.
- 9 Check the **write** checkbox for **Assigned Rights**.
- 10 Click **Done**.
- 11 Click **OK**.

2.2.10 Disabling the JMX Management Console on JBoss

The results from a Nessus scan show that the JBoss product installer does not secure the JMX management console by default. This creates a potential security hole.

To solve this problem, you need to disable the JMX console by following your JBoss documentation.

2.2.11 Updating a Password for a Database User on Tomcat

Perform the following actions to update the database user's password in the database server.

- 1 Stop Tomcat.
- 2 Update the password in the database server.
- 3 With Java in your path, enter the following command:

```
idm/apps/tomcat/lib/idm-datasource-factory-1.2.0-uber.jar novell %newpassword%
```

- 4 Copy the encrypted output of the password to the server.xml file.
- 5 Save and close the file.
- 6 Start Tomcat.

2.3 Digital Signature Configuration

This section provides instructions on configuring your environment to take advantage of the digital signature support provided with the Identity Manager User Application.

Digital signature support is available on JBoss only The User Application can be configured to support digital signatures on JBoss only. Digital signatures are not supported on WebSphere.

The User Application provides support for the digital signature as a service model. The digital signature as a service model is very easy to configure and manage. In most environments, you should use this model, unless you are not ready to upgrade from an older digital signature implementation, or require the ability to read digital signature cards, since this support is not available with digital signature as a service.

IMPORTANT: We strongly encourage you to use the digital signature as a service support, since it is the most reliable and easy-to-use configuration for managing digital signatures.

WARNING: You must use NetIQ Sentinel to preserve documents that you digitally sign. Digital signature documents are not stored with workflow data in the User Application database, but are stored in the logging database. You must enable logging to preserve these documents.

2.3.1 Setting Up the User Certificates

- 1 Create the user certificates using iManager.
 - 1a Log in as an administrator.
 - 1b Under **NetIQ Certificate Server**, select **Create User Certificate**.
 - 1c Select the users for whom you want to create certificates and click **Next**.

You can use the Object Selector or Object History to pick the users.
 - 1d Select the server and specify the certificate nickname. Specify **Custom** as the creation method and click **Next**.
 - 1e Specify a key size of 1024 or 2048 bits, depending on which size suits your requirements. Set the key type to **Signature**. Leave other settings as is and click **Next**.

- 1f If you're using the default configuration, leave the certificate parameters as is and click **Next**.
To enable certificate revocation list (CRL) support, select Custom and check the CRL signing check box.
For complete details on CRL configuration, see the NetIQ Certificate Server documentation.
- 1g Click **Finish**.
- 1h Log out.
- 2 Export the user certificate as a PFX file that contains the private key.
 - 2a Log in as the user for whom you want to export a certificate.
 - 2b Under **NetIQ Certificate Access**, select **View My Certificates**.
 - 2c Select a certificate and click the **Export** button.
 - 2d In the Export Certificate Wizard, click **Yes** to indicate that you want to export the private key with the certificate. Then click **Next**.
 - 2e Enter a password to protect the private key and click **Next**.
 - 2f Select **Export the certificate into the browser** if you do not have a card reader. Otherwise, click on the link that says **Save exported certificate to a file**.
You can also import to the browser later. Therefore, you might want to click on **Save exported certificate to a file** to import to a different browser.
 - 2g Click **Save to Disk** to save the file rather than opening it.
 - 2h Click **Close**.
- 3 If you're using a smart card, install the smart card reader driver.
- 4 Install the software needed to transfer certificate information to the smart card.
- 5 Import the key pair (certificate) to the smart card:

If you are planning to use browser certificate support, rather than the smart card, you can skip steps 3 through 5 above. Certificates can be imported into a browser using iManager or the browser certificate management user interface.

2.3.2 Configuring the Application Server

To configure the application server, follow these steps:

- 1 Download the `vaas.war` and `xmlsigner.war` files from a third-party component provider. Contact your sales representative to get a referral to a third-party provider.
- 2 For details on configuring the `vaas.war` file, see the third-party documentation.
- 3 To deploy to JBoss, copy the `vaas.war` and `xmlsigner.war` files to the `JBOSS_HOME/server/IDMProv/deploy` directory.
- 4 Export the trusted root and all intermediate certificates (using iManager) and import them into the key store specified in your system's local configuration using the `keytool` command.

For example, for JBoss:

```
keytool -import -trustcacerts -file certFile
```

The `certFile` is a fully qualified path to the certificate file.

If you're using the NetIQ Certificate Server, you do not need to export the trusted root.

- 5 (Not Required for Digital Signature as a Service) Start the User Application Configuration utility by running the configupdate script (`configupdate.bat` on Windows or `configupdate.sh` on Linux/Solaris).
- 6 (Not Required for Digital Signature as a Service) Click **Show Advanced Options**.
- 7 (Not Required for Digital Signature as a Service) Under **Trusted Key Store**, type the path to the certificate file in the **Trusted Store Path**. Also, type your password in the **Keystore Password** field. The default password is `changeit`.

The Trusted Key Store contains all trusted signers' certificates used to validate digital signatures.

NOTE: For JBoss, if you're using the NetIQ Certificate Server, you can simply paste the complete string (for example, `C:\Program Files\Java\jdk1.6.0_31\jre\lib\security\cacerts`) from the **Keystore Path** field under **eDirectory Certificates** to the **Trusted Store Path** under **Trusted Key Store**. You can also paste the **Keystore Password** to the **Trusted Store Password** field.

- 8 (Not Required for Digital Signature as a Service) If you are using OCSP, under **Miscellaneous**, type the URI for OCSP in the **OCSP URI** field. This value is used to update the status of trusted certificates online. The URI points to the access point for the Online Certificate Status Protocol server.

2.3.3 Configuring Logging

To enable logging of digital signatures, you need to configure the logging Platform Agent. The Platform Agent is required on any client that reports events to Sentinel. You configure the platform agent through the `logevent` configuration file.

IMPORTANT: If you are logging events that include digital signatures, it is critical that the value of the `LogMaxBigData` parameter be large enough to handle the data being logged.

For details on logging configuration, see [Chapter 3, "Setting Up Logging," on page 75](#).

2.3.4 Configuring the User Application

To configure digital signature support for the User Application:

- 1 Open the `sys-configuration-xmldata.xml` file:
This file is in the `conf` folder on JBoss. For example: `/opt/netiq/idm/rbpm/jboss/server/IDMProv/conf`.
- 2 Add the mapping to the `vaas.war` that you received from a third-party component provider in the `sys-configuration-xmldata.xml` before the `</properties>` element.

Here is the format:

```
<property>
  <key>com.netiq.dss.vaas.uri</key>
  <value>http(s)://%server%:%port%/vaas/verify</value>
</property>
```

For example:

```
<property>
  <key>com.netiq.dss.vaas.uri</key>
  <value>http://myserver.netiq.com:8180/vaas/verify</value>
</property>
```

- 3 Configure the Digital Signature Service for the User Application. To do this, you need to use the **Digital Signature Service** page on the **Administration** within the User Application. For details, see [Section 7.2.2, “Configuring the Digital Signature Service,” on page 163](#).
- 4 Log out of the User Application.
- 5 Stop the application server.
- 6 Complete the rest of the steps outlined by the documentation provided by the third-party component provider to configure and deploy the `xmlsigner.war` and `vaas.war` files.
- 7 Restart the application server.

2.3.5 Configuring the Provisioning Request Definitions

You can use Designer for Identity Manager to configure digital signature support for your provisioning request definitions.

To configure a provisioning request definition to support digital signatures, you need to:

- 1 Indicate whether a digital signature is required to initiate the provisioning request.
- 2 Indicate whether a digital signature is required for each approval step within the workflow. Because each approval step might have more than one outgoing link, you need to specify whether a digital signature is required for each link.

After you have indicated whether a digital signature is required to initiate a request or perform an approval step, you need to also specify the following for each request or approval step where a digital signature is required:

Table 2-1 Digital Signature Settings

Setting	Description
Digital Signature Type	<p>Specifies whether the digital signature uses data or form as its type:</p> <ul style="list-style-type: none"> ◆ Data: Specifies that the XML signature serves as the user agreement. When Data is selected, the XML data is written to the audit log. The user can preview XML data before submitting a signature. ◆ Form: Specifies that a PDF document that includes the digital signature declaration be generated. This document serves as the user agreement. The user can preview the generated PDF document before submitting a request or approval. When Form is selected, the PDF document (encapsulated in XML) is written to the audit log. <p>WARNING: You must use Sentinel to preserve documents that you digitally sign. Digital signature documents are not stored with workflow data in the User Application database, but are stored in the logging database. You must enable logging to preserve these documents.</p>
Digital Signature Declaration	<p>Specifies a digital signature confirmation string that confirms the user's signature.</p>

For details on configuring provisioning request definitions in Designer, see the *Identity Manager User Application: Design Guide*.

2.4 Performance Tuning

Performance tuning is a complex subject. The Identity Manager User Application relies on diverse technologies with many interactions. It is not possible to anticipate every single configuration scenario or user interaction scenario that could result in poor performance. Nevertheless, some subsystems are subject to best practices that can boost performance.

2.4.1 Logging

The User Application allows logging with the open source Apache *log4j* framework. File and console logging with *log4j* are enabled by default.

NOTE: The kinds of events you can log, and how to enable or disable logging, are covered in [Chapter 3, “Setting Up Logging,” on page 75](#).

The *log4j* configuration settings are contained in a file called

- ♦ `jboss-log4j.xml` in the install directory (if you are using a JBoss application server)
- ♦ `log4j.xml` in the User Application WAR (if you are using a non-JBoss application server)

Near the bottom of the `jboss-log4j.xml` file, look for the following entry:

```
<root>
  <priority value="INFO" />
  <appender-ref ref="CONSOLE" />
  <appender-ref ref="FILE" />
</root>
```

Assigning a value to `root` ensures that any log appenders that do not have a level explicitly assigned inherit the root level (in this case, INFO). For example, by default, the FILE appender does not have a threshold level assigned and so it assumes the root's.

The possible log levels used by *log4j* are DEBUG, INFO, WARN, ERROR, and FATAL, as defined in the `org.apache.log4j.Level` class. Inattention to the proper use of these settings can be costly in terms of performance.

A good rule of thumb is to use INFO or DEBUG only when debugging a particular problem.

Any appender included in the root that does have a level threshold set, should set that threshold to ERROR, WARN, or FATAL unless you are debugging something.

The performance hit with high log levels has less to do with verbosity of messages than with the simple fact that console and file logging, in *log4j*, involve synchronous writes. An `AsyncAppender` class is available, but its use does not guarantee better performance. The issues are well-known and are Apache *log4j* issues, not Identity Manager issues.

The default of INFO in the User Application's log config file (above) is satisfactory for many environments, but where performance is critical, you should consider changing the above `jboss-log4j.xml` entry to:


```
<root>
  <priority value="ERROR"/>
  <appender-ref ref="FILE"/>
</root>
```

In other words, remove `CONSOLE` and set the log level to `ERROR`. For a fully tested/debugged production setup, there is no need to log at the `INFO` level, nor any need to leave `CONSOLE` logging enabled. The performance payoff of turning these off can be significant.

For more information on `log4j`, consult the documentation available at <http://logging.apache.org/log4j/docs>.

2.4.2 Identity Vault

LDAP queries can be a bottleneck in a heavily utilized directory-server environment. To maintain a high level of performance with large numbers of objects, `eDirectory` (which is the basis of the Identity Vault in Identity Manager) records frequently requested information and stores it in indexes. When a complex query is run against objects with indexed attributes, the query returns much faster.

Out of the box, `eDirectory` comes with the following attributes already indexed:

```
Aliased Object Name
cn
dc
Equivalent to Me
extensionInfo
Given Name
GUID
ldapAttributeList
ldapClassList
Member
NLS: Common Certificate
Obituary
Reference
Revision
Surname
uniqueID
uniqueID_SS
```

When you install Identity Manager, the default directory schema is extended with new object class types and new attributes pertaining to the User Application. User-application-specific attributes are by default not indexed. For better performance, you might find it useful to index some of those attributes (and perhaps a few traditional LDAP attributes as well), particularly if your user container contains over 5,000 objects.

The general idea is to index only those attributes that you know are regularly queried, which could be different attributes in different production environments. The only way to know which attributes are heavily used is to collect predicate statistics at runtime. The collection process itself degrades performance, however.

The process for collecting predicate statistics is discussed in detail in the [eDirectory Administration Guide](https://www.netiq.com/documentation/edir88/edir88/data/a2iii88.html) (<https://www.netiq.com/documentation/edir88/edir88/data/a2iii88.html>). Indexing is also discussed in more detail there. In general, you need to do the following:

- ◆ Use `ConsoleOne` to turn on predicate-statistics collection for attributes of interest
- ◆ Put the system under load
- ◆ Disable statistics collection and analyze the results
- ◆ Create an index for each type of attribute that might benefit from having one

If you already know which attributes you want to index, there is no need to use ConsoleOne. You can create and manage indexes in iManager with eDirectory **Maintenance > Indexes**. For example, if you know that users of your org chart are likely to perform searches based on the isManager attribute, you can try indexing that attribute to see if performance is enhanced.

NOTE: As a best practice, it is recommended that you index, at a minimum, the manager and isManager attributes.

For an in-depth discussion of attribute indexing and performance, see “Tuning eDirectory” in *Guide to Troubleshooting eDirectory* by Peter Kuo and Jim Henderson (QUE Books, ISBN 0-7897-3146-0).

For more information about performance tuning, see “Maintaining eDirectory” (<https://www.netiq.com/documentation/edir88/edir88/data/a5zek7a.html>) in the *eDirectory Administration Guide* (<https://www.netiq.com/documentation/edir88/edir88/data/a2iii88.html>).

2.4.3 JVM

The amount of heap memory allocated to the Java virtual machine can impact performance. If you specify minimum or maximum memory values that are either too low or too high (too high meaning more than the physical memory of the machine), you could experience excessive pagefile swapping.

For a JBoss server, you can set the maximum JVM* size by editing the `run.conf` or `run.bat` file (the former for Linux, the latter for Windows) under `[IDM]/jboss/bin/` in a text editor. Increase “**-Xmx**” from `128m` to `512m`, or possibly higher. Some experimentation might be needed to determine the optimal setting for your particular environment.

NOTE: JBoss and Tomcat performance tuning tips are at <http://wiki.jboss.org/wiki/Wiki.jsp?page=JBossASTuningSliming> (<http://wiki.jboss.org/wiki/Wiki.jsp?page=JBossASTuningSliming>)

Increasing the stack size for recursive workflows If you have workflows that are recursive in nature (that execute loops), you might see a `StackOverflowError` at execution time. Java does not handle the stack space for recursive type functions effectively. Therefore, for recursive workflows, you need to increase the stack size for the JVM. The JVM defaults to 512K. You might want to increase the stack size to 1M.

To increase the stack size, you can include the `-Xss1M` setting with the `JAVA_OPTS` in your start JBoss script file.

```
JAVA_OPTS="-server -Xss1M -Xms512M -Xmx512M -XX:MaxPermSize=512m"
```

2.4.4 Session Time-out Value

The session time out (the amount of time a user can leave a page unattended in his or her Web browser before the server causes a session-time-out warning dialog box to appear) can be changed in the `web.xml` file in the `IDMProv.war` archive. This value should be tuned to match the server and usage environment in which the application runs. In general, it is advised that the session time out be

as small as practicable. If business requirements can tolerate a 5-minute session time out, this would allow the server to release unused resources twice as early as it would if the time-out value were 10 minutes. This improves performance and scalability of the Web application.

Consider the following when adjusting the session time out:

- ♦ Longer session time-outs can cause the JBoss server to run out of memory if many users log in over a short period of time. This is true of any application server that has too many open sessions.
- ♦ When a user logs in to the User Application, an LDAP connection is created for the user and bound to the session. Thus, the more sessions that are open, the greater the number of LDAP connections that are held. The longer the session time out, the longer these connections are held open. Too many open connections to the LDAP server (even if they are idle) can cause system performance degradation.
- ♦ If the server starts experiencing out-of-memory errors, and the JVM heap and garbage collection tuning parameters have already been optimally tuned for the server and usage environments, consider lowering the session time out.

You can set the session time out value on the User Application Configuration screen at installation time. Alternatively, you can modify the session time out after installation by performing a configuration update.

2.4.5 Tuning JBoss

By default, the JBoss deployment scanner runs every five seconds. For a production server, this is typically not necessary and might impact performance. You should consider changing the scan period so that the deployment scanner runs less frequently, or turn the deployment scanner off entirely. For information about configuring the deployment scanner, see “[Turn the Deployment Scanner frequency down or turn it off if you do not hot deploy.](https://community.jboss.org/wiki/TurnDeploymentScannerDown)” (<https://community.jboss.org/wiki/TurnDeploymentScannerDown>)

For more information about tuning JBoss for production environments, see “[JBoss 5.x Tuning/Slimming](https://community.jboss.org/wiki/JBoss5xTuningSlimming)” (<https://community.jboss.org/wiki/JBoss5xTuningSlimming>).

2.4.6 Using Secure Sockets for User Application Connections to the Identity Vault

By default, secure sockets are used for communication between the User Application server and the Identity Vault. However, in some environments, not all communication needs to be secured. For example, if the User Application and Identity Vault servers are on an isolated network, and the only ports available to the outside are the HTTP ports, it might be acceptable for some communication between the two servers to be accomplished using non-secure sockets. Some aspects of the application will *always* use a secure connection (for example, a user changing a password) even though the setting might indicate that secure connections are not required. Turning off secure connections, especially for user connections, can greatly increase performance and scalability. If, in a particular environment, there are many concurrent logins, and communication between the User Application server and the Identity Vault server have been secured using the network setup, then turning off the secure connection for user connections greatly increase the number of concurrent

logins that can be processed. We recommend that this option be used only when there is actual evidence of scaling or performance problems in the environment, and adding additional eDirectory servers is not an option.

Additionally, secure connections can be turned off for administrative connections. These connections are used for general queries on the Identity Vault server that do not require user credentials. These connections are pooled and used round-robin. The bind over a secure connection is only done once at application startup (or possibly again later on if the connection becomes unresponsive) and so does not represent the scalability issues that can arise with the user connections. However, the time it takes to encrypt and decrypt the data at both ends does add overhead. We recommend that the default setting be used, unless there is a need to gain extra performance.

Secure communications for administrative and user connections must be disabled in both the User Application and in iManager.

Disabling Secure Communications Using the User Application Configuration Tool

To disable the secure administrative and user connections in the User Application:

- 1 Run the `configupdate` script, located in the User Application directory, as follows:

- ♦ Linux: Type the following to run `configupdate.sh`:

```
./configupdate.sh
```

- ♦ Windows: Run `configupdate.bat`

The User Application configuration utility starts.

- 2 Deselect **Secure Admin Connection** and **Secure User Connection**.
- 3 Click **OK**.

Disabling Secure Communications Using iManager

To disable the requirement for secure LDAP (LDAPS) connections for administrative and user connections to eDirectory using iManager or ConsoleOne:

- 1 Log into your eDirectory tree.
- 2 Navigate to the **LDAP** group object and display its properties.
- 3 Click **General**.
- 4 Deselect **Require TLS for Simple Binds with Password**.

NOTE: In a multi-server eDirectory tree, disabling TLS on the LDAP group removes the TLS requirement from all servers. If you want mixed TLS requirements for each individual server in your tree, you must enable the TLS requirement on each server.

2.5 Clustering

You can install the User Application in a clustered environment. For more information, see the following sections:

- ♦ [Ensuring High Availability for Identity Manager](#) in the [NetIQ Identity Manager Setup Guide](#)
- ♦ [“Configuring Logging in a Cluster”](#) on page 93

- ♦ “Configuring Cache Settings” on page 110
- ♦ Section 7.2.4, “Configuring the Workflow Engine and Cluster Settings,” on page 168

2.6 Localizing and Customizing User Application Strings and Name Formats

Identity Manager provides several tools for localizing or customizing the User Application’s text.

2.6.1 Localizing User Application Strings

See [Table 2-2](#) for links to the documentation that describes how to localize User Application components.

Table 2-2 Localization and Customization Topics

Topic	Where to find it:
To set the User Application’s preferred locale	See the sections “Preferred Locale” and “Choosing a Preferred Language” in the Identity Manager User Application User Guide .
To localize email templates	See Section 10.4.4, “Adding Localized Email Templates,” on page 230.
To localize challenge questions	See “Security: Best Practices” in the <i>NetIQ Identity Manager Administration Guide</i> .
To localize the password sync status application name	See Table 5-11, “Password Sync Status Application Settings,” on page 140.
To localize the names of container or shared pages	See the Page Name property in Section 6.2.1, “Creating Container Pages,” on page 148. See Section 6.3.1, “Creating Shared Pages,” on page 153.
For general information about localizing provisioning objects or customizing their display text, such as: <ul style="list-style-type: none"> ♦ Directory abstraction layer objects ♦ Provisioning request definitions ♦ Workflow activity display names 	See “ Localizing Provisioning Objects, ” in the <i>NetIQ User Application: Design Guide</i> .

2.6.2 Customizing Text

The text displayed in the User Application is stored in either the User Application driver or in a set of language-based JAR files (UserAppStrings_xx.JAR) located in the User Application WAR. You use different tools to customize the text depending on where the text is stored. For text stored in the:

- ♦ **User Application driver:** Use the provisioning tools in Designer for Identity Manager.

For more information, see the section “Localizing Provisioning Objects” in the [User Application: Design Guide](#) (<http://www.netiq.com/documentation/idm45/index.html>).

- ♦ **User Application WAR:** Follow the procedure described in this section.

The text strings stored in the User Application WAR file are stored in a language-based JAR file. There is one JAR for each supported language, and the JAR file names indicate the associated language by appending the Java locale code for the associated language. The base name of the JAR file is `UserAppStrings_locale-code.JAR`. For example, the English language strings are stored in the `UserAppStrings_en.JAR`. The JAR files are copied to the `110n-resources` subfolder of the User Application's installation directory.

NOTE: To customize graphics or theming, see [Section 5.1.6, "Theme Administration," on page 123](#)

To customize the text stored in the User Application WAR, decide:

- ♦ ["What strings do I want to change?" on page 62](#)
- ♦ ["What languages do I want to support?" on page 63](#)

Once you know what to change, follow these steps:

- ♦ ["Modify or customize the strings" on page 63](#)
- ♦ ["Repackage the updated files and deploy" on page 63](#)
- ♦ ["Test your changes" on page 64](#)

You can follow the example outlined in ["Text Change Example" on page 64](#)

The User Application strings typically change between versions. This means that you have to apply your string changes or customizations to each new release.

What strings do I want to change?

- Review the User Application tabs.
- Make a list that includes each string you want to change and where the string is located in the User Application user interface. You can use this list when testing your changes to ensure that you made the text changes in the appropriate places.
- Determine whether the strings are stored in the User Application driver or in the language-based JAR. The following guidelines help you determine where the strings are stored:
 - ♦ **Navigation, headers, names, and instructions:** This content is stored in language-based JARs.
 - ♦ **Identity Self-Service tab:** This content is stored in language-based JARs, unless it is related to directory abstraction layer entities (such as display name, lists, and categories). Content related to directory abstraction layer entities is stored in the User Application driver.
 - ♦ **Work Dashboard tab:** This content is stored in language-based JARs, unless it is in the Form Details section. The content in the Form Details section is stored in the User Application driver.
 - ♦ **Administration tab:** This content is stored in language-based JARs.
 - ♦ **Roles and Resources tab:** This content is stored in language-based JARs unless it is related to role or SoD names or descriptions. Role or SoD names and descriptions are stored in the User Application driver.
 - ♦ **Compliance tab:** This content is stored in the language-based JARs unless it is related to the provisioning request definition. Content related to the provisioning request definition is stored in the User Application driver.

For information on changing the strings stored in the User Application driver, see the section “Localizing Provisioning Objects” in the *User Application: Design Guide*.

What languages do I want to support?

- Locate the JAR files corresponding to the language(s) you want to support. For example, English strings are located in `UserAppStrings_en.JAR`.
- Unzip the JARs into a working directory using a zip tool or the Java JDK JAR program.

Modify or customize the strings

- Search the files in the working directory to locate the strings you want to change.

Use a tool that allows you to search for a specified string across all of the files in a folder (the string(s) you want to change might be located in more than one file).

TIP: The strings in the files are in the properties file format of `key=value`. Modify only the value, and not the key. Modifying the key can cause run-time errors. Follow the additional editing tips identified below.

- When editing the properties files:
 - ◆ Use an editor that formats properties files for improved readability, when possible.
 - ◆ Use an editor that displays characters rather than unicode encoding for improved readability. Some of the properties files contain unicode-encoding.
 - ◆ In the editor, turn off wrapping to improve readability. It helps identify each property on a separate line.
 - ◆ In some files, the key value might be difficult to identify, especially if your editor does not provide automatic property file formatting. In these cases, search for the equals sign, and find the first occurrence of the equals sign that does not have a backslash preceding it (`\=`). The key precedes the equals sign (`=`), and the value follows it.
 - ◆ Make sure to maintain the proper properties file format.

To learn more, see [Java Properties Object](http://www.java.sun.com/j2se/1.5.0/docs/api/java/util/Properties.html#load) (<http://www.java.sun.com/j2se/1.5.0/docs/api/java/util/Properties.html#load>)

Repackage the updated files and deploy

- Add the properties files back to the JAR using the Java JDK `jar` program. You must maintain the folder location within the JAR.
- Add the modified JAR into the User Application WAR maintaining the folder location within the WAR. You can use the Java JDK `Jar` program.

This example uses the Java JDK `Jar` program and maintains the folder location:

```
jar -uvf IDMPProv.WAR WEB-INF/lib/UserAppStrings_en.jar
```

- Redeploy the WAR file to your application server.

Test your changes

- Access the User Application.
- Using your list of changes, review each occurrence of the string you changed to determine if you made the change appropriately.

Text Change Example

This section provides a simple text change example.

Open the User Application Welcome page and notice the text that says

```
Securely manage the access needs of your ever-changing user community
```

To change this string in the English language JAR:

- 1 Expand the UserAppStrings_en.JAR file to a temporary directory.
- 2 Use a find tool to locate the string.
- 3 Open the properties file and change the text from:

```
Securely manage the access needs of your ever-changing user community  
to
```

```
Test text change
```

- 4 Save and close the file.
- 5 Follow the steps described in [“Repackage the updated files and deploy”](#) on page 63.
- 6 Open the User Application’s Welcome page and verify your changes.

2.6.3 Ensuring that Characters Display Properly in Role Report PDF Files

By default, the role report feature of the Roles Based Provisioning Module uses “UniGB-UCS2-H” for the PDF encoding and “STSong-Light” for the PDF font for Chinese simplified, Chinese traditional, Russian and Japanese locales. For the other locales, “Cp1252” is used for PDF encoding and “Helvetica” or “Helvetica-Bold” is used for the PDF font.

If the user’s browser locale or preferred locale is set to one of the above four locales, the report will be able to display most of characters from these locales. However, some extended characters found in ISO-8859 may not be displayed properly in the report.

Conversely, if the browser locale or preferred locale is not set to one of these four locales then some Asian characters will not display properly.

To allow all characters to display properly in generated PDF files, you need to:

- ◆ Edit the Configuration XML Data in iManager
- ◆ Configure the User Application

Detailed instructions are provided below.

NOTE: You may also notice problems displaying some characters in role reports for languages that are not in the standard set of supported languages. If you add a new language (such as Polish), you may also need to perform the steps provided in this section to ensure that all characters display properly for that language as well.

Editing the Configuration XML Data in iManager

- 1 Login to iManager as your Administrator.
- 2 Click the View Objects icon.
- 3 In the Tree, navigate to the following location:

Context > Driver Set > Driver > AppConfig > AppDefs

For example:

netiq > TestDrivers > UserAppDriver > AppConfig > AppDefs

- 4 Click **configuration**.
- 5 In the Valued Attributes list, select `XmlData` and click **Edit**.
- 6 In the Edit Attribute window, search for `PREF_FONT` and replace the corresponding `<value></value>` with `<value>Arialuni.ttf</value>`.
- 7 Search for `PREF_ENCODING` and replace the corresponding `<value></value>` with `<value>Identity-H</value>`.
- 8 Click **OK**, then click **OK** again.
- 9 Restart the User Application Driver.

Configuring the User Application

- 1 If you edited the `</property>` tag (in [Step 6](#) and [Step 7 on page 65](#)) to add support for the four locales by specifying `Arialuni.ttf` as the preferred font, the directory that contains the file `Arialuni.ttf` (the name of the file must match the entry specified for `PREF_FONT` above) has to be added to the Application Server's classpath.

For example, if the file `Arialuni.ttf` was saved to the directory `/home/lab/font`, and the Application Server that the User Application war file is deployed on is JBoss, the start script for JBoss could be modified with the following entry:

```
JBOSS_CLASSPATH="$JBOSS_CLASSPATH:/home/user/font"
```

- 2 Restart the Application Server:
 - ♦ For JBoss, restart the Application Server once all configuration manual steps have been completed.
 - ♦ For WebSphere, the patched User Application war must be re-deployed with their Administration tools once all manual configuration steps have been completed.

NOTE: `Arialuni.ttf` is the Arial Unicode MS distributed by Microsoft. If you do not have permission to use it, then try to find and use another unicode font that supports as many characters as possible. Then update the font and encoding in [Step 6](#) and [Step 7 on page 65](#) and [Step 1 on page 65](#) with this information.

2.6.4 Ensuring that Dates Display Correctly in Norwegian

For language codes `no` and `nb`, you need to perform a workaround to ensure that dates display correctly in Norwegian. The `Date.js` file contains `no` but not `nb`, however, the `dmask` value (`dd/MM/yyyy`) is not correct. For both `no` and `nb`, the format should be `dd.MM.yyyy`.

To ensure that dates display correctly in Norwegian:

- 1 Copy the file `com/netiq/common/i18n/I18nDateTimeRsrc_en.properties`, modifying the locale portion of the file name to match the desired locale (for example, `I18nDateTimeRsrc_nb.properties`).
- 2 Modify the format(s) in the file to match the desired format. There are four format types: short, medium, long and full. These formats correspond to the `java.text.DateFormat.SHORT`, `.MEDIUM`, `.LONG` and `.FULL` constants.
- 3 Add the file to the `IDMProv.war` under `WEB-INF/classes/com/netiq/common/i18n` using the `jar` utility (file must be placed in a directory tree corresponding to the above path).

```
jar uvf IDMProv.war
WEB-INF/classes/com/netiq/common/i18n/I18nDateTimeRsrc_nb.properties
```

2.7 Configuring the Roles and Resources Tab

This section provides details on configuring the underlying subsystem for the Roles and Resources tab.

2.7.1 Role Service Driver Configuration

After creating the Role Service driver at installation time, you can optionally modify some of the driver configuration settings in iManager. To configure the Role Service driver:

- 1 In iManager, click **Identity Manager>Identity Manager Overview**.
- 2 Browse to the driver set where the driver exists, then click **Search**.
- 3 Click the upper-right corner of the Role Service driver icon, then click **Edit Properties**.
- 4 Click on the **Driver Configuration** tab.
- 5 Scroll down to the **Driver Settings** section of the page.
- 6 Make any changes you would like to the settings, and click **OK** to commit your changes.

You can modify the following standard driver settings (listed under **User Application/Workflow Connection** on the Driver Configuration page), which get their initial values at installation time:

Table 2-3 Standard Driver Settings

Option	Description
User Application Driver DN	The distinguished name of the User Application driver object that is hosting the role system. Use the eDirectory format, such as <code>UserApplication.driverset.org</code> , or browse to find the driver object. This is a required field.

Option	Description
User Application URL	The URL used to connect to the User Application in order to start Approval Workflows. This is a required field.
User Application Identity	<p>The distinguished name of the object used to authenticate to the User Application in order to start Approval Workflows. This needs to a user who has been assigned as a Provisioning Administrator for the User Application. Use the eDirectory format, such as admin.department.org, or browse to find the user.</p> <p>The identity needs to be entered in LDAP format (for example, cn=admin,ou=department,o=org), rather than dot format. Note that this is different from the format required at driver install time, where dot notation is expected.</p> <p>This is a required field.</p>
User Application Password	Password of the account specified in the User Application Identity field. The password is used to authenticate to the User Application in order to start approval workflows. This is a required field.
Reenter User Application Password	Re-enter the password of the account specified in the User Application Identity field.

In addition, you can modify the following additional settings (listed under **Miscellaneous** on the Driver Configuration page) to customize the behavior of the Role Service driver:

Table 2-4 Additional Settings for Customizing the Role Service Driver

Option	Description
Number of days before processing removed request objects	Specifies the number of days the driver should wait before cleaning up request objects that have finished processing. This value determines how long you are able to track the status of requests that have been fulfilled.
Frequency of reevaluation of dynamic and nested groups (in minutes)	Specifies the number of minutes the driver should wait before reevaluating dynamic and nested groups. This value determines the timeliness of updates to dynamic and nested groups used by the User Application. In addition, this value can have an impact on performance. Therefore, before specifying a value for this option, you need to weigh the performance cost against the benefit of having up-to-date information in the User Application.
Generate audit events	<p>Determines whether audit events are generated by the driver.</p> <p>For details on audit configuration, see Chapter 3, "Setting Up Logging," on page 75.</p>

Indexing for the Role Service Driver

The Role Service driver creates relevant indexes in eDirectory for roles definitions. If you upload a large number of roles, the indexing of these values may take some time. You can monitor these indexes under Index Management in iManager.

Here is the list of Index Names for the indexes created for the Role Service driver:

```
nrf(Object Class)
nrf(nrfMemberOf)
nrf(nrfStatus)
nrf(nrfStartDate)
nrf(nrfNextExpiration)
nrf(nrfParentRoles)
nrf(nrfChildRoles)
nrf(nrfCategory)
nrf(nrfRoleCategoryKey)
nrf(nrfLocalizedNames)
nrf(nrfLocalizedDescrs)
nrf(nrfRoles)
```

2.7.2 User Application Configuration

The **Configure Roles and Resources Settings** action on the **Roles and Resources** tab of the User Application allows you to specify administrative settings for the Role Subsystem. For details on using the **Configure Roles and Resources Settings** action, see the section on configuring the role subsystem in the *Identity Manager User Application: User Guide* (<http://www.netiq.com/documentation/idm402/pdfdoc/ugpro/ugpro.pdf>).

2.7.3 Security Roles

The Role Subsystem uses a set of system roles to secure access to functions within the **Roles and Resources** tab. Each menu action in the **Roles and Resources** tab is mapped to one or more of the system roles. If a user is not a member of one of the roles associated with an action, the corresponding menu item is not displayed on the **Roles and Resources** tab.

The *system roles* are administrative roles automatically defined by the system at install time for the purpose of delegated administration. These include the following:

- ◆ Role Administrator
- ◆ Role Manager

To assign users to the system roles, you need to use the **RBPM Provisioning and Security** action on the **Administration** tab. For details on assigning users to roles, see [Section 7.3, “Administrator Assignments,” on page 171](#).

The initial assignment of the Role Administrator is specified at installation time and processed when the Role Subsystem is first initialized at startup time.

2.7.4 View Request Status Search Limit

By default, the View Request Status action retrieves up to 10,000 request objects. If a user attempts to retrieve a larger result set, the user will see a message indicating that the limit has been reached. In this case, the user should narrow the search (by specifying a particular user or status, for example) to limit the number of objects returned in the result set. Note that when a user applies a filter to a role name, the filter limits what the user sees and its order, not the number of objects returned.

The administrator can change the maximum number of request objects retrieved by modifying the entity definition for the nrfRequest object in iManager. To do this, the administrator needs to modify the `<search-max>10000</search-max>` setting by editing the XmlData attribute of the sys-nrf-request object. The sys-nrf-request object can be found under EntityDefs.DirectoryModel.AppConfig within the User Application driver for the Roles Based Provisioning Module.

2.7.5 Provisioning Display Settings

The Administration tab in the User Application provides several settings that control various aspects of the user interface, including general display settings, task settings, and request status settings.

For details on the specifying the provisioning display settings for the user interface, see [Section 7.2.3, "Configuring the Provisioning UI Display Settings,"](#) on page 165.

2.7.6 E-Mail Notification

The Role Subsystem uses two templates that are specific to roles-based provisioning:

- ♦ **New Role Request** (Role Request Notification)
- ♦ **Role Request Approval Notification** (Role Request Approval Completed Notification)
- ♦ **New Resource Request** (Resource Request Notification)
- ♦ **Resource Request Approval Notification** (Resource Request Approval Completed Notification)

You can edit the templates to change the content and format of email messages. For more information on these templates, see [Section 10.4, "Working with Email Templates,"](#) on page 216.

2.7.7 Enabling Drivers for Resource Mappings

The Roles Based Provisioning Module ships with updated configuration files for the following drivers:

- ♦ Active Directory
- ♦ GroupWise
- ♦ LDAP
- ♦ Notes
- ♦ eDirectory
- ♦ SAP User Management
- ♦ SAP GRC Access Control

All of these updated driver configuration files contain a new section on the driver's Global Configuration Values (GCV) page labeled **Role and Resource Mapping**.

To display the configuration options available in the new section, select **show** for the **Show role and resource mapping configuration** GCV.

To enable resource mapping for the driver, select **Yes** for the **Enable resource mapping** GCV.

Depending on the driver's capabilities, one or more lower-level options are displayed once resource mapping is turned on. The Active Directory driver, for example, has three lower-level options:

- ◆ **Allow mapping of user accounts**
- ◆ **Allow mapping of groups**
- ◆ **Allow mapping of Exchange mailboxes**

Each option can be turned on or off individually by selecting **Yes** or **No**.

After saving the changes and restarting the driver, RBPM will detect the driver as enabled for resource mapping.

NOTE: Before RBPM can detect the driver, RBPM must query the entitlement system. RBPM sends the query to the entitlement system every 1440 minutes by default, but you can force the application to send the query immediately using the User Application.

To force the query to run immediately, log into the User Application using a User Application administrator account. Click **Roles and Resources > Configure Roles and Resources Settings**, then click the **Refresh** button under **Entitlement Query Status > Refresh Status**.

2.7.8 Creating a List in the RBPM Database

You can use lists in request forms to display various options for specifying a resource assignment. This section provides instructions for adding lists to the database by executing a few SQL statements. Once these lists have been created, they can be displayed on a request form on the Roles and Resources tab.

The following example shows how you would create a simple set of values for a list:

```
INSERT INTO PROVISIONING_CODE_MAP SET VIEWID='Factory-Locations', VERSIONNO=1,
DESCRIPTION='Factory Locations', NAME='Factory
Locations', ENTITYKEY='Factory-Locations', ENTITYTYPE=1,
LASTREFRESHED=UNIX_TIMESTAMP();
```

```
INSERT INTO PROVISIONING_VIEW_VALUE SET VALUEID='Factory-Locations-1',
VERSIONNO=1, VIEWID='Factory-Locations', PARAMVALUE='Cambridge, MA 02440';
```

```
INSERT INTO PROVISIONING_VIEW_VALUE SET VALUEID='Factory-Locations-2',
VERSIONNO=1, VIEWID='Factory-Locations', PARAMVALUE='Provo, UT 97288';
```

The following example uses SQL statements that work with PostgreSQL:

```
INSERT INTO PROVISIONING_CODE_MAP
(VIEWID,VERSIONNO,DESCRIPTION,NAME,ENTITYKEY,ENTITYTYPE,LASTREFRESHED)
VALUES ('Factory-Locations',1,'Factory Locations','Factory-Locations','Factory-
Locations',1,extract(epoch FROM now()));
```

```
INSERT INTO PROVISIONING_VIEW_VALUE (VALUEID,VERSIONNO,VIEWID,PARAMVALUE)
VALUES ('Factory-Locations-1','1','Factory-Locations','Cambridge, MA 02440');
```

```
INSERT INTO PROVISIONING_VIEW_VALUE (VALUEID,VERSIONNO,VIEWID,PARAMVALUE)
VALUES ('Factory-Locations-2','1','Factory-Locations','Waltham, MA 02451');
```

```
INSERT INTO PROVISIONING_VIEW_VALUE (VALUEID,VERSIONNO,VIEWID,PARAMVALUE)
VALUES ('Factory-Locations-3','1','Factory-Locations','Provo, UT 97288');
```

The VIEWID is the primary key for the PROVISIONING_CODE_MAP. The ENTITYTYPE value 1 identifies the map type as a list. The VIEWID is the foreign key for the PROVISIONING_VIEW_VALUE relationship to the PROVISIONING_CODE_MAP table. The VALUEID is the primary key for the PROVISIONING_VIEW_VALUE table.

After the Company Location field has been added to the form, you can specify that the company location value should come from the Company Locations list at request time.

After the Factory Location field has been added, you can specify that the factory location value must come from the Factory Locations list at request time.

At request time, the user can then select the company location and factory location values when assigning the resource.

After the resource has been assigned, the Request Status tab for the resource displays the parameter values chosen from the lists for the request form fields.

2.8 Configuring the Compliance Tab

This section provides details on configuring the Compliance tab.

NOTE: For compliance and attestation processes, use NetIQ Access Review instead of the User Application. Access Review enables administrators and managers to easily collect all user and access information in one central location and certify that each user has only the level of access that they need to do their job. Following the principle of least privilege, Access Review helps you ensure that your users have focused access to those applications and resources that they use and cannot access resources that they do not need to access. You can review all permissions assigned to your employees, either individually or as a group, and decide whether those permission assignments are appropriate. For more information, see the [NetIQ Access Review documentation \(https://www.netiq.com/documentation/access-review-10/\)](https://www.netiq.com/documentation/access-review-10/).

2.8.1 Security Roles

The **Compliance** tab enforces the RBPM security model to secure access to compliance functions. If a user is not a Compliance Administrator, the **Compliance** tab is not available.

A Compliance Administrator is designated at installation time. After installation, the Security Administrator can assign additional users to the Compliance Administrator role. For details on assigning users to the Compliance Administrator role, see [Section 7.3, “Administrator Assignments,” on page 171](#).

2.8.2 E-Mail Notification

When an attestation process is initiated, each attester receives an email message indicating that they must complete a compliance task. The message provides a link to the workflow activity that has been assigned to the attester. This behavior is enabled by default, but can be disabled in Designer.

The **Compliance Task** (Attestation Notification) template determines the content and format of email messages sent to attesters. For more information on this template, see [Section 10.4, “Working with Email Templates,” on page 216](#).

2.9 Configuring the Work Dashboard Tab

If you want users to be able to save their preferences within the Work Dashboard tab, you need to set the permissions appropriately for a couple of attributes in the Identity Vault. Specifically, if you want users to be able to save customizations associated with the Customize, Filter, and Rows actions (actions which allow control the display of data on several sections of the Work Dashboard), or change the sort of the data displayed, you need to ensure that the permissions on the `srvprvUserPrefs` and `srvprvQueryList` attributes are set so that the user is able to write to these attributes.

2.10 Recreating the Database After Installation

When you install the identity applications, you can postpone connecting to the database or creating tables in the database. If you do not have permissions to the database, you might need to choose this option. The installation program creates a SQL file that you can use to create the database schema. You can also recreate the database tables after installation without having to reinstall. To do so, you delete the database for the identity applications and create a new database with the same name. For more information, see “[Manually Creating the Database Schema](https://www.netiq.com/documentation/idm45/setup_guide/data/b1bc7sld.html)” (https://www.netiq.com/documentation/idm45/setup_guide/data/b1bc7sld.html) in the *NetIQ Identity Manager Setup Guide*.

2.11 Changing the Default Administrator Assignments After Installation

In this release, the following administrative accounts are assigned once during the initialization of the User Application:

- ◆ Compliance Administrator
- ◆ Provisioning Administrator
- ◆ RBPM Configuration Administrator
- ◆ Resource Administrator
- ◆ Roles Administrator
- ◆ Security Administrator

Modifying the mappings for these administrative accounts in the `configupdate` utility after the installation and initialization process will not work in this release. The check for assigning the administrative roles happens only once. At this time, a property is set that keeps track of when these roles were assigned.

NOTE: To modify the default administrator assignments for the User Application, you must first edit the `configupdate.sh` or `configupdate.bat` file and change the `-edit_admin` property to `true`. You can then use `configupdate` to modify the default assignments.

If you want to modify the default assignments for the administrative roles without deleting the Driver (which would cause all role assignments to be removed), you need to follow one of the three procedures shown below.

2.11.1 Granting or Removing Assignments in the User Application

To grant or remove the role assignment through the User Application:

- 1 Login to the User Application as the Security Administrator.
- 2 Go to the **Roles Catalog** on the **Roles and Resources** tab.
- 3 Select the administrative role you want to change (for example, the Provisioning Administrator).
- 4 Select **Edit**.
- 5 Select the **Assignments** tab.
- 6 If you want to remove the current assigned user, then select the user and press the **Remove** link.
- 7 To add a user, press the assign button where you will need to provide a description and the user to assign the role to and the press the **Assign** button.

2.11.2 Changing the Assignments in Configupdate Without Removing the Existing Users

To change any or all of the administrative assignments and keep the existing (default) users that have been granted the role assignment:

- 1 Stop the Application Server that the User Application WAR is deployed on.
- 2 Stop the User Application Driver.
- 3 Stop the Roles and Resource Service Driver.
- 4 Launch the configupdate utility.
- 5 Change the mappings for the administrative roles outlined above as required, and then press OK.
- 6 Using either iManager or ConsoleOne, go to `%DriverSet% -> %userApplication Driver% -> AppConfig -> AppDefs -> Configuration`. Then, go to the **Other** tab and open the XMLData.
- 7 Find the following entry:

```
<property>
  <key>com.novell.idm.security.domain-admin.initialized</key>
```

- 8 Delete the entry, from `<property>` to `</property>`. For example:

```
<property>
  <key>com.novell.idm.security.domain-admin.initialized</key>
  <value>20090831124642Z</value>
</property>
```

- 9 Close and Save.
- 10 Restart the User Application.
- 11 Restart the User Application Driver.
- 12 Restart the Roles and Resource Service Driver.
- 13 Access the User Application and in the logs you will see the administrative roles will be issued.

2.11.3 Changing the Assignments in Configupdate and Removing the Existing Users

To change any or all of the administrative assignments and remove the existing (default) users that have been granted the role assignment:

- 1 Stop the Application Server that the User Application WAR is deployed on.
- 2 Stop the User Application Driver.
- 3 Stop the Roles and Resource Service Driver.
- 4 Launch the configupdate utility.
- 5 Change the mappings for the administrative roles outlined above as required, and then press OK.
- 6 Using either iManager or ConsoleOne, go to *%DriverSet% -> %userApplication Driver% -> AppConfig -> AppDefs -> Configuration*. Then, go to the **Other** tab and open the XMLData.
- 7 Find the following entry:

```
<property>
    <key>com.novell.idm.security.domain-admin.initialized</key>
```

- 8 Delete the entry, from `<property>` to `</property>`. For example:

```
<property>
    <key>com.novell.idm.security.domain-admin.initialized</key>
    <value>20090831124642Z</value>
</property>
```

- 9 Close and Save.
- 10 Using either ConsoleOne or iManager, remove the user from the role and then the role from the user.
- 11 Restart the User Application.
- 12 Restart the User Application Driver.
- 13 Restart the Roles and Resource Service Driver.
- 14 Access the User Application and in the logs you will see the administrative roles will be issued.

3 Setting Up Logging

Logging is the main tool you use for debugging the identity applications configuration. The logging service provides facilities for writing, viewing, filtering, and listening for log messages. The application server processes and applications that run on the application server or in client JVMs generate these log messages.

This sections discusses the following topics:

- ◆ [Section 3.1, “How Logging Services Help,” on page 75](#)
- ◆ [Section 3.2, “What Can Be Logged,” on page 82](#)
- ◆ [Section 3.3, “How Logging Works,” on page 82](#)
- ◆ [Section 3.4, “Types of Log Files,” on page 85](#)
- ◆ [Section 3.5, “Understanding the Log Format,” on page 87](#)
- ◆ [Section 3.6, “Configuring Logging,” on page 89](#)
- ◆ [Section 3.7, “Configuring Logging in a Cluster,” on page 93](#)
- ◆ [Section 3.8, “Logging to a Sentinel Server,” on page 95](#)
- ◆ [Section 3.9, “Logging to OpenXDAS,” on page 97](#)
- ◆ [Section 3.10, “Using Log Files for Troubleshooting,” on page 98](#)
- ◆ [Section 3.11, “Log Events,” on page 99](#)

3.1 How Logging Services Help

An application server’s instance uses logging services to communicate its status and respond to specific events, including server startup and shutdown information, failures of one or more subsystems, errors, warning messages, access information on HTTP requests, and additional information. For example, you can use Tomcat application server’s logging services to report error conditions or listen for log messages from a specific subsystem.

All administrative and end-user actions and events are logged to the server console and to Tomcat server’s log file. This allows easy access to this information for security and operational purposes. Additionally, the audit log system provides the ability to monitor ongoing activities such as authentication activity, up time of the system, and so on. File logging is enabled by default.

The identity applications features are implemented in a layered architecture. Each feature uses one or more packages. Each package handles a specific area of a feature and has its own independent log level that obtains event messages from different parts of the application. The logs contain information about processing and interactions among identity applications components that occur while satisfying users and administrative requests and during general system processing. By enabling the correct log levels for various packages, an administrator can monitor how identity applications processes users and administrative requests. The package names are based on log4j conventions. The event messages include these package names indicating the context of the message output. The logs include tags and values that allow the administrator to identify and correlate which package log entries pertain to a given transaction and user. [Table 3-1](#) describes some of the features and the packages they use.

Table 3-1 Identity Manager User Application Packages

Feature	Description	Packages	Notes
Roles	<p>Roles are permanently stored in the Identity Vault. For fast access to roles information, Identity Manager stores roles in a local cache called permission index. When a role is requested, the User Application queries the permission index for that role. When a role is modified through the User Application driver, the change is reflected in the permission index. For more information about roles, see Section 1.1.2, "About Roles-Based Provisioning," on page 20.</p>	<ul style="list-style-type: none"> ◆ com.novell.idm.nrf.service ◆ com.novell.idm.nrf.persist ◆ com.novell.srvprv.impl.vdata.model ◆ com.netiq.idm.rest.catalog 	<p>For troubleshooting any issues when a role is assigned, revoked, or expired, monitor the Roles and Resource driver log.</p> <p><code>com.novell.srvprv.impl.vdata.model</code> is a verbose package when set to Debug log level. It generates messages for each object class and attributes present in Virtual Data Access (DAL). For example, it shows all DAL lookups. This can result in a large amount of logs. To limit the number of messages, you can set the log level to Warn. For more information about the messages generated by <code>com.novell.srvprv.impl.vdata.model</code>, see Section 33.1.2, "Virtual Data Access Logging," on page 632.</p> <p>For troubleshooting issues related to managing roles, see Section 33.1.6, "When a Role Is Requested," on page 637.</p>

Feature	Description	Packages	Notes
Resources	Resources are permanently stored in the Identity Vault. For fast access to resources information, Identity Manager stores resources in a local cache called permission index. When a resource is requested, the User Application queries the permission index for that resource. When a resource is modified, the change is reflected in the permission index is updated. For more information about resources, see Section 1.1.3, "About Resource-Based Provisioning," on page 21.	<ul style="list-style-type: none"> ◆ com.novell.idm.nrf.service ◆ com.novell.idm.nrf.persist ◆ com.novell.srvprv.impl.vdata.model 	For troubleshooting any issues when a resource is assigned or revoked, monitor the Roles and Resource driver log.
Code Map Refresh	Code map is a local cache used by the User Application to store entitlements values for all connected systems from the Identity Vault. The User Application queries the Identity Vault for the drivers that are in running state and have entitlements. The User Application updates the User Application database at configurable intervals with entitlement changes.	<ul style="list-style-type: none"> ◆ com.novell.idm.nrf.service ◆ com.novell.idm.nrf.persist ◆ com.novell.srvprv.impl.vdata.model 	<p>For troubleshooting any connected system issue, enable DSTrace on the driver.</p> <p>For viewing sample log messages related to code map refresh, see Section 33.1.3, "When a Code Map Refresh Is Triggered," on page 635.</p>
Proxy	Enables you to manage proxy configuration. Identity Manager stores proxy definition in the ProxyDefs container in the User Application driver. For more information about configuring proxy, see Section 7.2.1, "Configuring Delegation and Proxy Settings," on page 161.	<ul style="list-style-type: none"> ◆ com.novell.srvprv.impl.security.service ◆ com.netiq.idm.rest.access ◆ com.novell.soa.af.impl.persist ◆ com.novell.srvprv.apwa.actions 	When a user is designated as a proxy, check the audit events for any suspicious activity.

Feature	Description	Packages	Notes
Delegation	<p>Enables you to manage delegation configuration based on a user's availability. A delegate is another user that you can temporarily grant permission to view and resolve your workflow work items. A delegate can view his delegator tasks in the task page and act on them. Identity Manager stores delegate definitions in the DelegateDefs container in the User Application driver. For more information about configuring delegation, see Section 7.2.1, "Configuring Delegation and Proxy Settings," on page 161.</p>	<ul style="list-style-type: none"> ◆ com.novell.srvprv.impl.security.service ◆ com.novell.srvprv.apwa.actions 	<p>When a user is made a delegate for another user, check the audit events for any suspicious activity that can occur through delegation.</p>
Database connectivity/updates	<p>Any schema changes made in the User Application are updated in the database when the User Application server is started and <code>com.netiq.idm.create-db-on-startup</code> flag is set to true in the <code>ism-configuration</code> properties file.</p> <p>When this flag is set, the database compares the existing schema with target schema and then updates the database schema.</p> <p>To update the database with any application configuration changes, you must set <code>com.netiq.idm.rbpm.updateConfig-On-StartUp</code> flag to true in the <code>ism-configuration</code> properties file.</p>	com.novell.soa.persist	.
Configuration Item (Landing page)	<p>Allows you to manage application items on the landing page. You can quickly navigate to internal and external pages of the application. For more information, see Configuring Identity Manager Home Items in the <i>NetIQ Identity Manager Home and Provisioning Dashboard User Guide</i> .</p>	com.netiq.idm.icfg	

Feature	Description	Packages	Notes
Workflow Tasks	<p>A task can be controlled by a workflow process. A workflow process can include one or more steps that must be performed before Identity Manager can complete a task that is under workflow control. A job is a runtime instance of a workflow process.</p> <p>The Workflow Engine is responsible for managing and executing steps in a workflow and for keeping track of state information which is persisted in a database. For more information, see Part IV, “Configuring and Managing Provisioning Workflows,” on page 185.</p>	<ul style="list-style-type: none"> ◆ com.novell.soa.af.impl.core ◆ com.novell.soa.af.impl.activity ◆ com.netiq.idm.rest.access 	
Separation of Duties	<p>Allows you to prevent users from being assigned to conflicting roles unless someone in your organization makes an exception for the conflict. To eliminate conflicts in role assignments, you perform certain management tasks such modify role definition and set up a proper approval process. For more information, see Section 1.1.2, “About Roles-Based Provisioning,” on page 20.</p>	<ul style="list-style-type: none"> ◆ com.novell.idm.nrf.service ◆ com.novell.idm.nrf.persist ◆ com.novell.srvprv.impl.vdata.model 	
My Permission	<p>A user can view a list of role and resource permissions assigned to him or for other users. For more information, see Viewing Your Permissions in the <i>NetIQ Identity Manager Home and Provisioning Dashboard User Guide</i> .</p>	<ul style="list-style-type: none"> ◆ com.netiq.idm.rest.access ◆ com.netiq.idm.rest.access.util 	
History	<p>A user can review the status and history of the permission requests (role, resource, PRD) for himself or for other users. For more information, see Viewing Your History in the <i>NetIQ Identity Manager Home and Provisioning Dashboard User Guide</i> .</p>	<ul style="list-style-type: none"> ◆ com.netiq.idm.rest.access ◆ com.novell.idm.nrf.persist ◆ com.netiq.idm.rest.access.util 	
Group	<p>Allows you to manage groups. For example, you can create, modify and delete a group based on access privileges. For more information, see Creating Users or Groups in the <i>NetIQ User Application: User Guide</i>.</p>	<ul style="list-style-type: none"> ◆ com.netiq.idm.rest.catalog ◆ com.novell.srvprv.spi.vdata.model 	

Feature	Description	Packages	Notes
Organization Chart	The Organization Chart page shows the hierarchy of users in your organization. A user can view the organization chart and quick information about the users based on the access rights set by the administrator. For more information, see Using the Organization Chart in the <i>NetIQ User Application: User Guide</i> .	<ul style="list-style-type: none"> ◆ com.netiq.idm.rest.access ◆ com.novell.srvprv.impl.portlet.orgchart ◆ com.novell.srvprv.impl.servlet.service ◆ com.novell.soa.portlet ◆ com.netiq.idm.rest.access.util 	
User Catalog	<p>You can create, modify, and delete users. A new user is created under the base container configured for the user. Based on the access control list rights, the user information can be edited.</p> <p>The user attributes can be configured to view, edit, and search by using the client settings. For more information, see Creating Users or Groups in the <i>NetIQ User Application: User Guide</i>.</p>	<ul style="list-style-type: none"> ◆ com.netiq.idm.rest.access ◆ com.netiq.idm.infosrv ◆ com.novell.srvprv.impl.vdata.model ◆ com.netiq.idm.settings ◆ com.novell.idm.nrf.service ◆ com.novell.idm.security.authorization.service ◆ com.novell.idm.nrf.persist ◆ com.novell.idm.security.authorization.ldap ◆ com.netiq.idm.rest.access.util 	
Make a request	A user can request a permission for himself or for another user. The Request page directly fetches the permission from the Permission index. The requested permission is directly assigned or through an approval process. For more information, see Requesting Permissions in the <i>NetIQ Identity Manager Home and Provisioning Dashboard User Guide</i> .	<ul style="list-style-type: none"> ◆ com.netiq.idm.rest.access ◆ com.netiq.idm.infosrv ◆ com.novell.srvprv.spi.vdata.model ◆ com.novell.idm.nrf.ajaxservice ◆ com.novell.idm.nrf.service ◆ com.novell.idm.nrf.persist ◆ com.novell.idm.security.authorization ◆ com.novell.soa.af.impl.core ◆ com.novell.soa.af ◆ com.novell.idm.nrf.assignment 	

Feature	Description	Packages	Notes
Permission Index	<p>Roles, resources, and PRDs are permanently stored in the Identity Vault. For fast access, Identity Manager stores this information on the User Application server in a set of cache files called Permission Index. When you install the identity applications, the process creates a permission index for the application server hosting the identity applications. For more information, see Preparing Your Environment for the Identity Applications in the <i>NetIQ Identity Manager Setup Guide</i>.</p> <p>When a request is issued, the identity applications query the permission index for the requested information.</p>	<ul style="list-style-type: none"> ◆ com.netiq.idm.cis ◆ com.netiq.cis.permindex ◆ com.netiq.idm.cis.permfilter ◆ com.sssw.fw.core ◆ com.netiq.uaconfig 	Only applicable to NetIQ Identity Manager Home and Provisioning Dashboard.
Directory Abstraction Layer	<p>The directory abstraction layer provides a virtual access to the Identity Vault data. You define a set of entities and their related attributes (virtual data) based on the Identity Vault objects that you want users to view, modify, or delete in the User Application.</p> <p>For more information, see Preparing Your Environment for the Identity Applications in the <i>NetIQ Identity Manager Setup Guide</i>.</p>	<ul style="list-style-type: none"> ◆ com.novell.srvprv.impl.vdata.mod el 	For viewing sample log messages related to Virtual Data Access, see Section 33.1.2, "Virtual Data Access Logging," on page 632.

The logs generated by the packages are primarily intended for debugging the software, although they can be used to detect any other software that is not behaving properly. System administrators and support personnel can identify and isolate problems caused by configuration errors, invalid user data, or network problems such as broken connections. However, component file logging is typically the first step in identifying software bugs.

Package logging is more verbose than audit logging. It increases the processing load. On a day-to-day basis, you are recommended to enable only log levels of error conditions and system warnings. If a specific problem occurs, logging can be set to **Info** or **Debug** to gather extra information needed to isolate and resolve the detected problem. When the problem is resolved, logging should be reconfigured to log only error conditions and system warnings.

3.2 What Can Be Logged

The identity applications functionality that deals with workflows (PRDs) and actions such as granting and managing of roles, resources, and entitlements can be logged. The log level for these features is controlled by configuring logging for the packages used by them. You can change log levels from the Logging page. For more information, see [“Changing Log Levels” on page 121](#).

The identity applications functionality that executes on a client and does not directly execute in context of the User Application web server cannot be logged in the same way. For example, most of the form processing that occurs in the client. The action scripts defined on the form control's onLoad, onChange, or custom events execute in the browser of the user's client computer and not in the User Application web server. Therefore, if an error occurs while rendering the form or processing an action script, it cannot be directly logged. However, Identity Vault queries issued from a form or Start Activity can be logged for troubleshooting the identity applications features. To view the identity applications client errors and informational messages, click the Console tab or the Network tab under Developer Tools section in the client browser. It contains HTTP response codes for both success and failed requests. The identity applications allow you to record the outcome of a user's request and response.

Identity Applications allow you to log what happens with a user's request and response during certain times:

- ◆ Between the browser and the application server
- ◆ Between the application server and the User Application database
- ◆ Between the application server and the Identity Vault

You can configure the log files to include entries for the following events:

- ◆ Configuration
- ◆ Events processed by the identity applications components, such as authentication, role assignment, and resource access
- ◆ Error conditions

The log files help you determine which of the following reasons is responsible for a request failure:

- ◆ The browser did not send the required information
- ◆ Directory Access Layer or the Identity Vault did not send the web client browser the required information

To view Identity Manager processing events in Identity Manager drivers, use Trace. Specify appropriate trace values to the driver set and the drivers in Designer or iManager. For more information, see [Viewing Identity Manager Processes in the NetIQ Identity Manager Driver Administration Guide](#).

3.3 How Logging Works

The following sections describe the identity applications logging environment and provide an overview of the logging process.

3.3.1 Terminology

Log4j has three main components: loggers, appenders, and layouts. These components work together to accomplish the following tasks:

- ♦ Record messages based on message type and level.
- ♦ Control how log messages are formatted and where they are reported at runtime.

Logger: In Log4j terminology, a logger is a named entity. Log4j defines a `Logger` class. A `Logger` object records messages for a specific subsystem or application component. An application can create multiple loggers, each with a unique name. In a typical usage of Log4j, an application creates a `Logger` instance for each application class that will emit log messages. Logger names are case-sensitive and they follow the Java package dot notation naming convention.

All loggers specific to the identity applications are defined in the `idmuserapp_logging.xml` file. You can set the severity level for each logger at any level in the hierarchy from the Logging Administration page or by editing the `log4j` file. For more information, see [Section 3.6.3, “Specifying the Severity Level for Commons Logging API Loggers,” on page 91](#).

Appender: In Log4j terminology, an output destination is called an appender. Log4j defines appenders to represent destinations for logging output. You can define multiple appenders. For example, an application might define an appender that sends log messages to standard out, and another appender that writes log messages to a file. Additionally, you can configure individual loggers to write to zero or more appenders. For example, you can configure the loggers to send all logging messages (all levels) to a log file, but only Error level messages to standard out. To change the destination of the log files, stop the identity applications and then change the settings in the Logging Location and Appender section of the `log4j.properties` file. Identity Applications provide a full suite of appenders offered by Log4j. For more information about appenders, see [Log4j documentation](#).

The Console and File appenders are defined in the `tomcat-log4j.xml` file. The `NAudit` appender that is specific to the identity applications is defined in the `idmuserapp_logging.xml` file.

Layout: Log4j defines layouts to control the format of log messages. Each layout specifies a particular message format. A specific layout is associated with each appender. This lets you specify a different log message format for standard out than for file output if required.

3.3.2 Components for Logging

A logging system includes the following basic components:

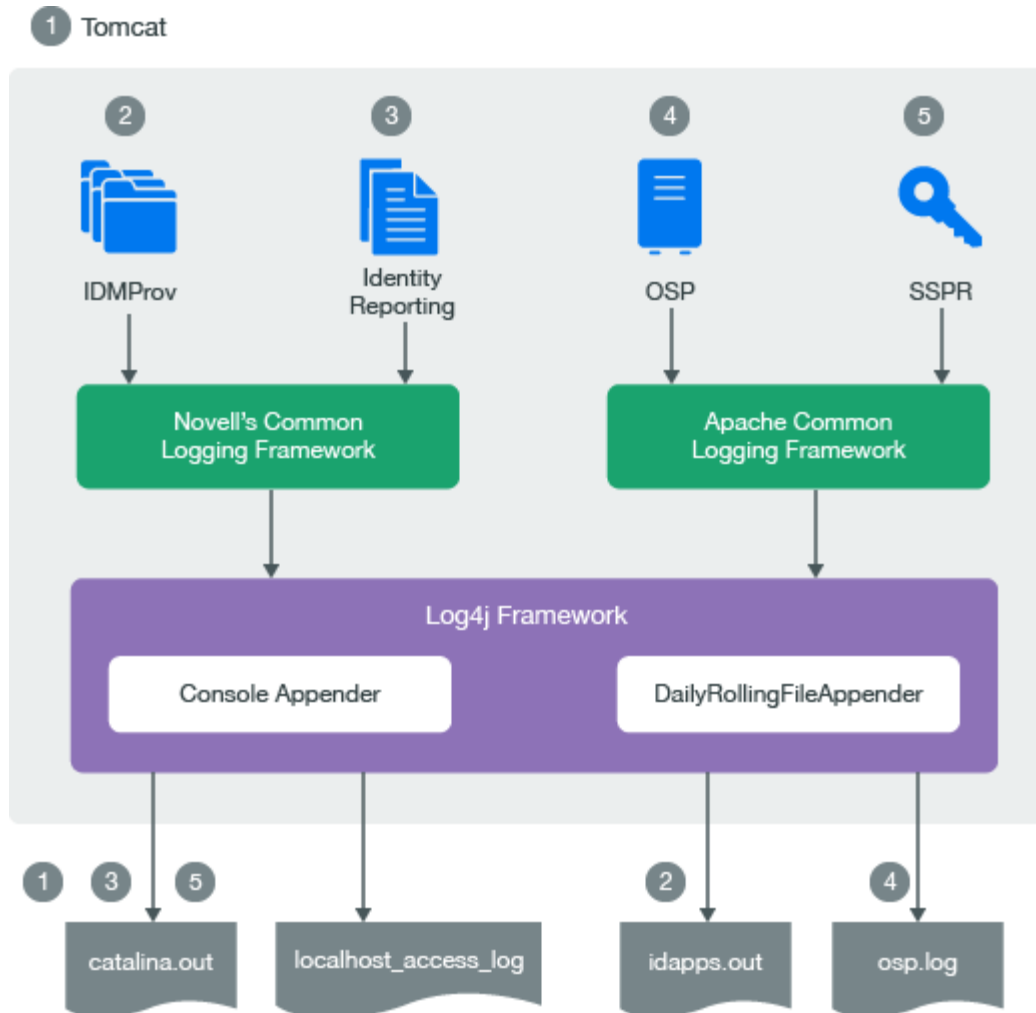
- ♦ A component that produces log messages
- ♦ A component that distributes (publish) log messages

The Tomcat subsystems use log4j to produce messages. By default, Tomcat supports Java based logging to distribute messages. The `LoggingHelper` class provides access to the `java.util.logging.Logger` object used for server logging. The Java Logging APIs can be used to add custom handlers, filters, and formatters. Alternatively, you can configure Tomcat to use Log4j APIs to distribute log messages.

3.3.3 How Logging Works

The identity applications support logging by using a custom-developed logging framework that integrates with log4j, an open-source logging package distributed by The Apache Software Foundation. In identity applications environment, Tomcat subsystems and identity applications components send log requests to the `Logger` objects. The `Logger` objects then assign `LogRecord`

objects, which are passed to Appender objects for publication. By default, the logger objects log messages to the system console and to the Tomcat server's log file at `Info` logging level and above. Events are logged to all activated loggers.



- The Tomcat server's log messages are directed to `catalina.out` and `idapps.out`.
- User Application's log messages are directed to `idapps.out`.
- Identity Reporting's log messages are directed to `catalina.out`.
- OSP's log messages are directed to `osp.out`.
- SSPR's log messages are directed to `catalina.out`.

Custom appenders like `NAuditAppender` are created to handle log messages in order to convert the messages to a specific format and send them to the configured auditing service. To configure event message output to an auditing service, see [“Sending Log Messages to an Auditing Service” on page 122](#).

To configure logging, see [Section 3.6, “Configuring Logging,” on page 89](#).

3.4 Types of Log Files

An identity applications framework uses several components such as OSP, SSPR, Dashboard, User Application driver, and Role and Resource driver. Each component has its own logging configuration that defines the default log levels and appender configuration for that component. Logs for all identity applications components including OSP and Identity Reporting are logged to the `catalina.out` file while the Localhost log records the interactions between the Tomcat server and the client. Some components such as OSP maintain additional log files that help in auditing their individual interactions. This section discusses different log files that are generated in an identity applications environment and what each of them contains.

3.4.1 Difference Among Catalina, Application, and Localhost Log Files

Catalina Log: This is the global log. It records information about events such as the startup and shutdown of the Tomcat application server, the deployment of new applications, or the failure of one or more subsystems. The messages include information about the time and date of the event and the ID of the user who initiated the event.

The `catalina.log` file contains all log messages that are written to Tomcat's `system.out` and `system.err` streams. Tomcat's internal log statements use the `java.util.logging` package (`juli`) to log. The default destination for that log is `standard.out`.

The `catalina.out` file can include:

- ◆ Uncaught exceptions printed by `java.lang.ThreadGroup.uncaughtException(..)`
- ◆ Thread dumps, if you requested them via a system signal

Each Tomcat server instance prints a subset of its messages to `standard.out` or `idapps.out` file. The `idapps.out` log file is specific to User Application. The `Userapp-log4j.xml` file stores the logging configuration for User Application, which directs all log messages to `idapps.out`.

The `catalina.out` file is located on the computer that hosts the Tomcat server instance. Each server instance has its own `catalina.out` file. By default, the `catalina.out` file is located in the `logs` directory under Tomcat's root directory. For example, `/opt/netiq/idm/apps/tomcat/logs/catalina.out`. To view messages in the `catalina.out` file, log in to the computer hosting Tomcat and use a standard text editor.

NetIQ recommends that you do not modify the log files by manually editing them. Modifying a file changes the timestamp and can confuse log file rotation. In addition, editing a file might lock it and prevent it from recording information from the Tomcat server.

Some operating systems enable you to redirect standard out to some other location. By default, a server instance prints only messages of `Info` severity level or higher to standard out. You can modify the severity threshold as a logging configuration so that the server prints more or fewer messages to respective log files.

Localhost Log : This is the log for all HTTP transactions between the client and the application server. The log file is named as, `localhost_access_log.<date of log generation>.txt` file. The default location and rotation policy for this log is the same as `catalina.out` file.

Application Log: Each identity application component is responsible for its own logging. Tomcat provides no support for application logs. Each component will have its own logging configuration where default log levels and appender configurations are defined. These logging configuration files are placed under `\conf` directory of Tomcat server.

3.4.2 Additional Log Files

The `catalina.out` log messages and log files communicate events and conditions that affect Tomcat server's operations. Logs for all identity applications components including OSP and Identity Reporting are also logged to the `catalina.out` file. This file also records the interactions between the Tomcat server and the client.

Some subsystems and components also maintain additional log files that help in auditing their individual interactions. The following list describes some of the additional log files:

- ♦ Logging information from the User Application is also logged into the `idapps.out` file. However, this file does not contain Tomcat server specific information.
- ♦ OSP logs are additionally stored in a separate file, `osp-idm-<date of log generation>.log` file located in `/opt/netiq/idm/apps/tomcat/logs/` directory. Logging is turned off by default and must be enabled in the `setenv.sh` file in the `/TOMCAT_INSTALLED_HOME/bin/` directory.
- ♦ Identity Reporting logs are additionally stored in `/var/opt/netiq/idm/log/`.
- ♦ SSPR logs are stored in `/opt/netiq/idm/apps/sspr/sspr_data/logs/SSPR.log`.
- ♦ The HTTP subsystem keeps a log of all HTTP transactions between the client and the application server in a text file, `localhost_access_log.<date of log generation>.txt` file. The default location and rotation policy for this log is the same as the `catalina.out` file. You can set the attributes that define the behavior of HTTP access logs for your server.
- ♦ By default, logs for User Application driver and Role and Resource Service driver are added to DTrace. The trace is turned off by default and must be enabled in the driver configuration by using Designer or iManager or console. When enabled, DTrace displays messages related to operations that the driver performed or tried to perform, at the level of detail specified by the driver trace level, as the engine processes the events. The driver trace level affects only the driver or driver set where it is set. You can also specify to write the trace information for a driver to a separate file. For more information, see [Viewing Identity Manager Processes](#) in the *NetIQ Identity Manager Driver Administration Guide*.
- ♦ Each server has a transaction log which stores information about committed transactions coordinated by the server that may not have been completed. Tomcat uses the transaction log when recovering from a system crash or a network failure. You cannot directly view the transaction log. The file is in a binary format.
- ♦ The auditing service records information from a number of security requests, which are determined internally by the security framework. The service also records the event data associated with these security requests and the outcome of the requests. Each server writes auditing data to its own log file in the server directory.
- ♦ The JDBC subsystem records various events related to JDBC connections, including registering JDBC drivers and SQL exceptions. The events related to JDBC are written to the server log, such as when connections are created or refreshed or when configuration changes are made to the JDBC objects.
- ♦ The JMS logging is enabled by default when you create a JMS server. The identity applications use Apache ActiveMQ as a JMS provider. Logs can be found in the ActiveMQ installed path at `data/activemq.log`. The identity applications rely on a Java Message Service (JMS) persistent store to persist e-mail messages. If JMS is not properly configured, any e-mail messages in the memory queue will be lost if the application server is shut down.
- ♦ The Hibernate framework writes log messages in different categories and log levels. For example, it logs messages for establishing connections with the database, executed SQL statements, or cache interactions. Hibernate logging is enabled by default at Info level. The events related to JDBC subsystem are written to the server log. All Hibernate related information is logged when the data is persisted in the application.

3.5 Understanding the Log Format

The identity applications have a specific format for file log entries. To improve log entry readability, the log entries in the `catalina.out` files use standard elements. This facilitates the use of non-interactive stream-oriented editors such as `sgrep`, `sed`, `awk`, and `grep`. The first part of each message begins with locale-formatted timestamp followed by a data portion that contains information specific to the log entry. The data portion is the most flexible part of a log entry.

A log entry has the following fields:

```
time-date-stamp [Severity] [Subsystem] [Message Text]
```

The following entry is an example entry that is logged when a user has requested for a role:

```
2017-03-08 08:43:10,660 INFO com.novell.idm.nrf.service.RoleManagerService- [RBPM]
[Role_Request] Requested by cn=David.Scully,ou=Active,ou=People,o=acme, Target DN:
CN=Kaitlin.Demore,OU=active,OU=People,O=acme, Source
DN:cn=PennDOT_Vehicle_Certification,cn=Application
Access,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm
46,ou=services,o=acme, Request DN:cn=20160308084310-
15da49b28ddf4ee1b7d71b4ce220c080-
0,cn=Requests,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm46,ou=services,o
=acme, Request Category: 10, Request Status: 0, Original Request Status: 0,
Correlation ID: UserApp#UserStartWorkflow#082080ec-5b10-498f-b87d-28825ab63278
```

Most log entries do not use the optional line breaks (`\n`). Notice that the time-date-stamp, the severity, subsystem, and message text are on the same line so that stream-oriented editors that use only one line (such as `grep`) can be used to locate related log entries.

If a message is logged within the context of a transaction, the message text contains the Correlation ID assigned to the transaction.

The Tomcat server uses the host computer's default character encoding for the messages it writes.

3.5.1 Message Fields

A Tomcat server message contains a consistent set of fields as described in the following table. In addition, if your applications use Tomcat logging services to generate messages, those messages will contain these fields.

Table 3-2 Fields in a Log Entry

Field	Description
Time-date-stamp	Time and date when the message originated in a format that is specific to the locale. The JVM that runs a Tomcat server instance refers to the host computer operating system for information about the local time zone and format. The date and time is specified in the W3C profile format of ISO 8061. It has the following fields: year-month-day-T-hour-minutes-seconds-time zone. The Z value for the time zone indicates that the time is specified in UTC.
Severity	Indicates the degree of impact of the event reported by the message such as warning, informational, or debug. In the example log entry, the level of severity is Info.

Field	Description
Subsystem	<p>Indicates Tomcat's subsystem or the type of the module that was the source of the message. For example, RBPM or Java Messaging Service (JMS).</p> <p>In the example log entry, this field contains the following string:</p> <pre>com.novell.idm.nrf.service.RoleManagerService- [RBPM] [Role_Request]</pre>
Message text	<p>A description of the event or condition specific to the log entry. It can be as simple as an informational string, such as the string in the example log entry:</p> <pre>Requested by cn=David.Scully,ou=Active,ou=People,o=acme, Target DN: CN=Kaitlin.Demore,OU=active,OU=People,O=acme, Source DN:cn=PennDOT_Vehicle_Certification,cn=Application Access,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=UserApplic ation,cn=idm46,ou=services,o=acme, Request DN:cn=20160308084310- 15da49b28ddf4ee1b7d71b4ce220c080- 0,cn=Requests,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm46,o u=services,o=acme, Request Category: 10, Request Status: 0, Original Request Status: 0,</pre> <p>If a message is logged within the context of a transaction, the message text contains an identifier assigned to the identity applications transaction. Identity applications transactions are actions such as authenticating a user, processing a request for a role, and request for access to a resource.</p> <p>An identifier is assigned to each identity applications transaction.</p> <p>If a user requests access to multiple resources, multiple request objects are created and each request is given a separate Correlation ID. Each request is processed separately and status of each can be seen in the user request history.</p> <p>The example log entry contains the following Correlation ID:</p> <pre>UserApp#UserStartWorkflow#082080ec-5b10-498f-b87d-28825ab63278</pre>

3.5.2 Message Severity

The severity attribute of a log message indicates the potential impact of the event or condition that the message reports. The following table lists the severity levels of log messages from the identity applications subsystems, starting from the lowest to the highest level of impact.

Level	Description
Fatal	The least detail. Writes fatal errors to the log.
Error	Writes errors (plus all of the above) to the log.
Warn	A suspicious operation or configuration has occurred but it might not affect normal operation. Writes warnings (plus all of the above) to the log.
Info	Used for reporting normal operations; a low-level informational message. Writes informational messages (plus all of the above) to the log.
Debug	Writes debugging information (plus all of the above) to the log.
Trace	Used for most detailed messages. You can configure this level to report the request path of a method. Writes tracing information (plus all of the above) to the log.

You can set a log severity level on loggers and appenders. When set on a logger, none of the appenders receives any events that are rejected by that logger. For example, if you set the log level to `Info` on a logger, none of the appenders will receive `Warn` level events. When you set a log level on the appender, the restriction only applies to that appender and not to other appenders. For example, if you turn `Error` off for the `File` appender, no `Error` messages are written to the log file. However, `Error` messages are written to the `standard.out` file.

```
<logger name="com.sssw" level="INFO" additivity="true">
    <appender-ref ref="NAUDIT"/>
</logger>
<logger name="com.netiq" level="DEBUG" additivity="true">
    <appender-ref ref="NAUDIT"/>
</logger>
```

You set log levels for loggers using the Logging Administration page or the `log4j` file. For more information, see [Section 3.6.4, “Configuring Logging in User Application Administration Tab,” on page 91](#). Loggers can also be configured through APIs. You can only enable or disable an appender.

The identity applications modules generate many messages of lower severity and fewer messages of higher severity. For example, under normal circumstances, they generate many `Info` or `Trace` messages. If your application uses Tomcat logging services, it can use an additional severity level of `Debug`.

3.6 Configuring Logging

You can configure logging to troubleshoot errors or to receive notification for specific events. For example, configure logging to perform the following activities:

- ◆ Stop recording of `Debug` and `Info` messages in the log file.
- ◆ Allow recording of `Info` level messages from the HTTP subsystem in the log file.
- ◆ Configure an appender to publish messages only whose severity level is `Warning` or higher.
- ◆ Track log information for individual servers in a cluster.

You can change the log settings in two ways:

- ◆ **Configure logging in User Application Administration Page:** Your changes will be in effect only until you restart the identity applications.
- ◆ **Edit the properties file:** Your changes will take effect next time you start the identity applications. These changes are permanently stored for future sessions.

You may need to configure the `logging.properties` file in some cases.

The following sections describe basic configuration tasks:

- ◆ [Section 3.6.1, “Understanding Logging Configuration,” on page 90](#)
- ◆ [Section 3.6.2, “Understanding the Log Level Settings,” on page 90](#)
- ◆ [Section 3.6.3, “Specifying the Severity Level for Commons Logging API Loggers,” on page 91](#)
- ◆ [Section 3.6.4, “Configuring Logging in User Application Administration Tab,” on page 91](#)
- ◆ [Section 3.6.5, “Editing the log4j Files,” on page 92](#)
- ◆ [Section 3.6.6, “Managing Log File Size,” on page 93](#)

3.6.1 Understanding Logging Configuration

When you enable logging, a logging request is sent to subscribed appenders. Tomcat provides appenders for sending log messages to the `standard.out` file and the server log (`catalina.out`) file. You can control logging for each type of handler by filtering log messages based on severity level and other criteria. For example, the Stdout Handler has a Notice threshold severity level by default. Therefore, Info and Debug level messages are not sent to the `standard.out` file.

By default, event messages are logged to the system console and to the Tomcat server's log file at Info logging level and above. Events are logged to all activated loggers.

The default behavior of the Tomcat or JBoss server is to limit the console log4j appender to display log messages with a verbosity of Info or less. To see log messages for more verbose levels (for example, Debug), you need to examine the server log file. Notice that the low threshold settings, such as Debug are extremely verbose and will increase Tomcat's startup time.

WARNING: You must use NetIQ Sentinel to preserve your digitally signed documents. Digital signature documents are not stored with workflow data in the database. These are stored in the audit logging database. You must enable logging to preserve these documents. For more information, see [Section 2.3, "Digital Signature Configuration," on page 52](#).

3.6.2 Understanding the Log Level Settings

Console logging involves synchronized writes. Therefore, logging can become a processor usage issue and concurrency impedance. You can change the priority value default setting to Error, on a Tomcat server, by modifying the setting in the `<installdir>/Tomcat/server/IDMProv/conf/tomcat-log4j.xml` file. Locate the root node that looks similar to this:

```
<root>
  <priority value="INFO"/>
  <appender-ref ref="CONSOLE"/>
  <appender-ref ref="FILE"/>
</root>
```

Change the priority value to:

```
<root>
  <priority value="ERROR"/>
  <appender-ref ref="CONSOLE"/>
  <appender-ref ref="FILE"/>
</root>
```

Assigning a value to the root ensures that any appenders do not have a level assigned inherit the root's level.

3.6.3 Specifying the Severity Level for Commons Logging API Loggers

If you are using the Commons Logging API, logger names follow the Java package dot notation naming convention. For example, a logger name can be `com.acme.Barlogger`, corresponding to the name of the classes in which it is used. Each dot-separated identifier appears as a node in the Logger tree. In this case, the logger named `com.acme` is a parent of the logger named `com.acme.Barlogger`.

You can configure the severity for a package or for any logger at any level in the tree. For example, if you specify the severity level for package `com.acme=Warn`, then `Fatal` and `Error` messages from the child nodes of this package will be blocked. You can override the severity level of a parent node by explicitly setting a value for a child node. For example, if you specify the severity level for `com.acme.Barlogger=Debug`, all log messages from `Barlogger` will be allowed, while `Fatal` and `Error` messages will be filtered for other child nodes under `com.acme`.

You can specify the severity level for a package or a logger in the following ways:

- ◆ Change the log level settings in the Logging page of the User Application. The changes will be immediately applied. When you restart User Application, the changes are not preserved. To persist the changes for subsequent sessions, select **Persist the logging changes**. Alternatively, modify the `log4j.properties` file.
- ◆ Edit the `log4j.properties` file. Your changes will take effect when you restart the User Application. Identity Applications preserves this configuration for subsequent sessions.

If you change, enable, or disable logging, you need not restart the identity applications to apply the changes.

3.6.4 Configuring Logging in User Application Administration Tab

You can change the logging behavior of the individual packages (loggers) via the Administration tab of the User Application while the application is running. Any changes made in this way will apply only to the currently running application session and are not written to the `log4j.properties` file.

The Logging page shows a list of all currently defined loggers. On this page, you can:

- ◆ Add a new logger for a class or package name.
- ◆ Remove a logger for a class or package name.
- ◆ Set the logging level (`Fatal`, `Error`, `Warn`, `Info`, `Debug`, `Trace`) for each class or package name.
- ◆ Reset all logging levels.

All logging configuration cannot be changed in the Logging administration page, such as Tomcat server specific configuration and appender configuration. For making such changes, stop the identity applications and then edit the `log4j.properties` file.

Perform the following actions to change the logging configuration:

- 1 Log in to the User Application as the User Application Administrator.
- 2 Select **Administration**.
- 3 Click **Logging**.
- 4 Change the **Log Level** of any package. Higher logging levels include the lower levels in the log.

Level	Description
Fatal	The least detail. Writes fatal errors to the log.
Error	Writes errors that can cause system processing to not proceed.
Warn	Logs potential failures, but the impact on execution is minimal. Warnings indicate that you should be aware that this event is happening and might want to make a configuration change to avoid it.
Info	Logs informational messages. No execution or data impact occurred.
Debug	Includes debugging information.
Trace	The most detail. Writes tracing information (plus all of the above) to the log.

5 To save the changes for application server restarts, select **Persist the logging changes**.

6 Click **Submit**.

Identity Manager saves the logging configuration in `idmuserapp_logging.xml` file. For example, on a Tomcat server, the default location is `<installdir>/tomcat/server/IDMProv/conf/` directory.

3.6.5 Editing the log4j Files

The configuration settings for the identity applications logging are stored in the `idmuserapp_logging.xml` in the install directory on the application server. For example, the log4j configuration settings are contained in `tomcat-log4j.xml` in Tomcat's install directory. To configure the logging levels and other settings permanently, stop your application server and change the settings in these files.

The log4j configuration settings are stored in the following locations:

- ♦ `jboss-log4j.xml` in the install directory on a JBoss application server
- ♦ `log4j.xml` in the User Application WAR on a non-JBoss application server

The User Application logging configuration settings are stored in the following locations:

- ♦ `idmuserapp_logging.xml` in the install directory on a JBoss application server
- ♦ custom location specified by the variable `idmuserapp.logging.config.dir` on Websphere

NOTE: The Console and File appenders are defined in `<application server>-log4j.xml`. The NAudit appender that is specific to the identity applications is defined in `idmuserapp_logging.xml`. All loggers specific to the identity applications are defined in `idmuserapp_logging.xml`.

To change the log level in the `tomcat-log4j.xml` file, open the file in a text editor and locate the following entry at the end of the file:

```
<root>
  <priority value="INFO" />
  <appender-ref ref="CONSOLE" />
  <appender-ref ref="FILE" />
</root>
```

Assigning a value to root ensures that any log appenders that do not have a level explicitly assigned inherit the root level (in this case, Info). For example, the File appender does not have a default threshold level assigned. It assumes the root's threshold level.

The possible log levels used by log4j are Debug, Info, Warn, Error, and Fatal, as defined in the `org.apache.log4j.Level` class. Inattention to the proper use of these settings can be costly in terms of performance.

A good rule of thumb is to use Info or Debug only when debugging a particular problem.

Any appender included in the root that does have a level threshold set, should set that threshold to Error, Warn, or Fatal unless you are debugging something.

The performance hit with high log levels has less to do with verbosity of messages than with the simple fact that console and file logging, in log4j, involve synchronous writes. An `AsyncAppender` class is available, but its use does not guarantee better performance. These are known issues of Apache log4j.

The default log level of Info in the log configuration file for the identity applications is suitable for many environments. However, for a performance intensive environment, you can change the entry as follows:

```
<root>
  <priority value="ERROR"/>
  <appender-ref ref="FILE"/>
</root>
```

For a fully tested and debugged production setup, enabling Info and Console logging are not needed. For more information about log4j, see [Apache Logging Services](#).

3.6.6 Managing Log File Size

By default, event messages are logged to both of the following:

- ♦ The system console of the application server where the identity applications components are deployed
- ♦ A log file on that application server. For example: `/opt/netiq/idm/apps/tomcat/logs/catalina.out` on a Tomcat server.

This is a rolling log file. By default, the server rotates the file based on a time interval of 24 hours. However, you can instruct the server to rotate the file over to another file after it reaches a certain size by specifying the size in the `log4j.appender.R.MaxFileSize` property in the `$TOMCAT_HOME/lib/log4j.properties` file. It does not rotate the local server log file when you start the server.

To cause the immediate rotation of the log file, change the appender configuration in the `log4j.properties` file.

By default, the rotated files are stored in the same directory where the log file is stored. You can specify a different directory location for the archived log files in the `log4j.properties` file.

3.7 Configuring Logging in a Cluster

This section includes tips for configuring logging in a cluster.

- ♦ [Tomcat Logging](#)
- ♦ [JBoss Logging](#)
- ♦ [User Application Logging](#)

3.7.1 Tomcat Logging

You can configure Tomcat for logging in a cluster. To enable logging for clusters, you need to edit the `tomcat-log4j.xml` configuration file, located in the `\conf` directory for the Tomcat server configuration (for example, `\server\IDM\conf`), and uncomment the following section at the end of the file:

```
<!-- Clustering logging
-->
- <!--
  Uncomment the following to redirect the org.jgroups and
  org.tomcat.ha categories to a cluster.log file.
  <appender name="CLUSTER"
class="org.tomcat.logging.appender.RollingFileAppender">
  <errorHandler class="org.tomcat.logging.util.OnlyOnceErrorHandler"/>
  <param name="File" value="{tomcat.server.home.dir}/log cluster.log"/>
  <param name="Append" value="false"/>
  <param name="MaxFileSize" value="500KB"/>
  <param name="MaxBackupIndex" value="1"/>
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>
  </layout>
</appender>
<category name="org.jgroups">
  <priority value="DEBUG" />
  <appender-ref ref="CLUSTER"/>
</category>
<category name="org.tomcat.ha">
  <priority value="DEBUG" />
  <appender-ref ref="CLUSTER"/>
</category>
-->
```

You can find the `cluster.log` file in the `log` directory for the Tomcat server configuration (for example, `\server\IDM\log`).

3.7.2 JBoss Logging

You can configure JBoss for logging in a cluster. To enable logging for clusters, you need to edit the `jboss-log4j.xml` configuration file, located in the `\conf` directory for the JBoss server configuration (for example, `\server\IDM\conf`), and uncomment the section at the bottom that looks like this:

```

<!-- Clustering logging
-->
- <!--
  Uncomment the following to redirect the org.jgroups and
  org.jboss.ha categories to a cluster.log file.
  <appender name="CLUSTER"
class="org.jboss.logging.appender.RollingFileAppender">
  <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
  <param name="File" value="{jboss.server.home.dir}/log cluster.log"/>
  <param name="Append" value="false"/>
  <param name="MaxFileSize" value="500KB"/>
  <param name="MaxBackupIndex" value="1"/>
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>
  </layout>
</appender>
<category name="org.jgroups">
  <priority value="DEBUG" />
  <appender-ref ref="CLUSTER"/>
</category>
<category name="org.jboss.ha">
  <priority value="DEBUG" />
  <appender-ref ref="CLUSTER"/>
</category>
-->

```

You can find the `cluster.log` file in the `log` directory for the JBoss server configuration (for example, `\server\IDM\log`).

3.7.3 User Application Logging

The logging configuration is not propagated to all servers in cluster. For example, if you use the Logging administration page on a server in a cluster to set the logging level for `com.netiq.afw.portal.aggregation` to Trace, this setting is not propagated to the other servers in the cluster. You must individually configure the level of logging messages for each server in the cluster.

For more information, see [Section 5.1.4, “Logging Configuration,” on page 119](#).

3.8 Logging to a Sentinel Server

To enable NetIQ Sentinel logging in your Identity Manager environment, you need to configure the Platform Agent on your application server and then enable Sentinel logging.

3.8.1 Configuring the Platform Agent

The Platform Agent is required on any client that reports events to Sentinel. You configure the Platform Agent through the `logevent` configuration file. This file provides the configuration information that the Platform Agent needs to communicate with the Sentinel server. The default location for this file, on the application server, is:

- ◆ Linux: `/etc/logevent.conf`
- ◆ Windows: `/<WindowsDir>/logevent.cfg` (Usually `c:\windows`)

To configure the Platform Agent:

- 1 Configure the Platform Agent on your application server.

Specify the following properties:

Loghost: The IP address or DNS name of your Sentinel server. For example:

```
LogHost=xxx.xxx.xxx.xxx
```

LogJavaClassPath: The location of the lcache jar file `NAuditPA.jar`. For example:

```
LogJavaClassPath=/opt/netiq/idm/NAuditPA.jar
```

LogCacheDir: Specifies where lcache stores cache files. For example:

```
LogCacheDir=/opt/netiq/idm/naudit/cache
```

LogCachePort: Specifies on which port lcache listens for connections. The default is 288, but in a Linux server, set the port number greater than 1000. For example:

```
LogCachePort=1233
```

BigData Specifies the maximum number of bytes that the client will allow. Larger amounts of logging data will be truncated. The default value is 3072 bytes, but you should change this to at least 8192 bytes to handle a typical form that has approximately 15 fields on a half page.

```
LogMaxBigData=8192
```

IMPORTANT: If your data is very large, you might want to increase this value. If you are logging events that include digital signatures, it is critical that the value of `LogMaxBigData` be large enough to handle the data being logged.

Specify any other settings needed for your environment.

NOTE: You must restart the Platform Agent any time you change the configuration.

- 2 Restart the Platform Agent for the changes to take effect.

3.8.2 Enabling Sentinel Logging

- 1 Log in to the User Application as the User Application Administrator.
- 2 Select **Administration**.
- 3 Click **Logging**.
- 4 Select **Also send logging messages to audit service**.
- 5 To save the changes for subsequent restart of the Tomcat server, make sure **Persist the logging changes** is selected.
- 6 Click **Submit**.

NOTE: To enable logging for Role events, you must select the **Generate audit events** property for the Role and Resource Service driver. For more information on this property, see [Section 2.7.1, "Role Service Driver Configuration," on page 66](#).

3.9 Logging to OpenXDAS

NOTE: Support for OpenXDAS will be deprecated from Identity Manager 4.6.

To use OpenXDAS for logging events, you must install, configure, then enable OpenXDAS in the User Application. To learn about:

- ♦ Installing Open XDAS, see the OpenXDAS instructions at [OpenXDAS.org \(http://downloads.sourceforge.net/openxdas\)](http://downloads.sourceforge.net/openxdas)
- ♦ Configuring OpenXDAS logging, see the “OpenXDAS Logger Configuration” in the OpenXDAS User’s Manual.

The OpenXDAS daemon or service (`xdasd`) must be running when you start the User Application (if you have enabled OpenXDAS logging). For this reason, you should configure the OpenXDAS daemon to start automatically.

- ♦ On Linux, use the `/etc/init.d/xdasd` start command to start the daemon. To start it automatically, change the runlevel using the “System Services” editor (SUSE) or directly edit the `/etc/init.d/rc*` directories.
- ♦ On Windows, install as a service, or use the command-line options on `xdasd.exe` to create the service. See “Command Line Configuration” in the OpenXDAS User’s Manual.

If your OpenXDAS daemon/service is not running and you are configured for OpenXDAS logging you’ll get a error stack trace and the User Application might not start successfully. The error message looks like this:

```
"ERROR [com.sssw.fw.servlet.Boot:contextInitialized] Unable to configure logging. com.novell.soa.common.LocalizedRuntimeException: Error Initializing OpenXDAS Audit."
```

3.9.1 Using OpenXDAS with Sentinel

To use OpenXDAS with Sentinel, you must configure the netstream logger. The netstream logger is undocumented, but it is required to send XDAS audit messages to a Sentinel server. The netstream logger does not perform any encryption, so the stream needs to be secured in another way (for example, SSH tunnel).

You must specify netstream entries in the `xdasd.conf` file for the loggers, server and port. For example on Linux:

```
xdasd.loggers=/usr/lib64/openxdas/libxdm_netstream.so
```

```
xdasd.loggers.netstream.server = 151.155.226.50
xdasd.loggers.netstream.port = 1468
```

On Windows, the server and port entries are the same, but the location of the `xdasd.loggers` entry is different. For example, on Windows:

```
xdasd.loggers=c:\Program Files\OpenXDAS\Loggers\xdm_netstream.dll
```

```
xdasd.loggers.netstream.server = 151.155.226.50
xdasd.loggers.netstream.port = 1468
```

NOTE: On Windows, you must move the `xdasd.conf` file to the `c:\windows` folder. If you do not, the `xdasd.exe` is unable to locate it.

3.9.2 Enabling OpenXDAS Logging in the User Application

You can enable OpenXDAS logging in your Identity Manager User Application in two ways:

- ♦ Select OpenXDAS as a logging option during the installation procedure.
- ♦ Enable OpenXDAS logging using the User Application Administration (described next).

To enable OpenXDAS logging, perform the following actions:

- 1 Log in to the User Application as the User Application Administrator.
- 2 Select the **Administration** tab.
- 3 Select the **Logging** link.
- 4 Select the **Also send logging messages to OpenXDAS** check box (near the bottom of the page).
- 5 To save the changes for any subsequent application server restarts, make sure **Persist the logging changes** is selected.
- 6 Click **Submit**.

NOTE: To enable logging for Role events, the Role Service driver [Generate audit events](#) property must be selected. For more information on this property, see [Section 2.7.1, “Role Service Driver Configuration,”](#) on page 66.

For troubleshooting information, see [Section 25.2, “Troubleshooting Open XDAS Logging,”](#) on page 621.

3.10 Using Log Files for Troubleshooting

The following example has trace messages logged to the `catalina.out` file when a code map refresh cycle is triggered.

```
2015-08-29 16:05:05,500 [INFO] CodeMapEngine [RBPM] Refreshing the Entitlementment CODE
MAP tables...
2015-08-29 16:05:05,499 [TRACE] BestLocaleServletFilter [RBPM] Using Resource-
Group[common-resgrp] with bestLocale[en] for /IDMProv/GwtServiceRouter
2015-08-29 16:05:05,510 [DEBUG] DirXMLDriverDAO [RBPM] Entering
getDriversEnabledForMappings() mappingType=2
2015-08-29 16:05:05,600 [DEBUG] DirXMLDriverDAO [RBPM] Exiting
getDriversEnabledForMappings()
2015-08-29 16:05:05,600 [INFO] CodeMapEngine [RBPM] Done refreshing the Entitlementment
CODE MAP tables
```

The descriptions of the log messages are sequentially listed below:

- ♦ The first message indicates that a code map refresh is in process. This is the first entry when a code map refresh is triggered.
- ♦ The second message specifies the entitlement values are persisted to the best suited locale.
- ♦ The third message provides information about the drivers enabled with entitlements.
- ♦ The fourth message indicates that the process is exiting after obtaining the information about the drivers enabled with entitlements.
- ♦ The fifth message indicates that the code map is updated with the entitlements values from the drivers and the process is completed.

3.11 Log Events

The Identity Manager User Application logs a set of events automatically from workflow, search, detail, and password requests. By default, the Identity Manager User Application automatically logs the following events to all active logging channels:

Table 3-3 Logged Events

Event ID	Process	XDAS Event	Severity
31400	Detail portlet		Info
31401			Info
31410	Change Password portlet		Error
31411			Info
31420	Forgot Password portlet		Error
31421			Info
31430	Search portlet		Info
31431			Info
31440	Create portlet		Info
31520	Workflow		Error
31521			Info
31522			Info
31523			Info
31524			Info
31525			Info
31526			Info
31527			Info
31528			Info
31529			Info
31534			Info
31535			Info
31537			Info
3152A			Info
3152B			Info
3152C			Info
31533			Info
31538			Info
31539			Info

Event ID	Process	XDAS Event	Severity
3153A			Info
3153B			Info
3153C		XDAS_AE_CREATE_DATA_ITEM	Info
3153D		XDAS_AE_CREATE_ROLE	Info
3152D	Provisioning		Error
3152E			Info
3152F			Info
31530			Error
31531			Info
31532			Info
31550		XDAS_AE_CREATE_SESSION	Info
31551		XDAS_AE_CREATE_SESSION	Info
31450	Security Context		Info
31451			Error
31452			Info
31453			Error
31454			Info
31455			Error
31456			Info
31457			Error
31458			Info
31459			Error
3145A			Info
3145B			Error
3145C			Info
3145D			Error
3145E			Info
3145F			Error
31600	Role Provisioning	XDAS_AE_APPROVAL_REQUESTED	Info
31601		XDAS_AE_APPROVAL_REQUESTED	Error

Event ID	Process	XDAS Event	Severity
31610	Role Assignment Request		Info
31611			Error
31612			Info
31613		XDAS_AE_CREATE_DA TA_ITEM_ASSOC	Info
31614		XDAS_AE_TERMINATE_ PEER_ASSOC	Info
31615		XDAS_AE_TERMINATE_ PEER_ASSOC	Error
31620	User Entitlement	XDAS_AE_CREATE_DA TA_ITEM_ASSOC	Info
31621		XDAS_AE_CREATE_DA TA_ITEM_ASSOC	Error
31622		XDAS_AE_TERMINATE_ DATA_ITEM_ASSOC	Info
31623		XDAS_AE_TERMINATE_ DATA_ITEM_ASSOC	Error
31624			Error
31630	Role Management		Info
31631			Error
31632			Info
31633			Error
31634			Info
31635			Error
31640			Info
31641			Error
31642			Info
31643			Error
31644			Info
31645			Error
31646		XDAS_AE_MODIFY_DA TA_ITEM_ATT	Info
31647		XDAS_AE_MODIFY_DA TA_ITEM_ATT	Error

Events Not Sent Through OpenXDAS The User Application supports sending events to both Sentinel Audit through NAudit Platform Agent and to Sentinel Syslog through OpenXDAS. However, the following events, which are generated from the IDM Engine, are only sent to Sentinel Audit through NAudit Platform Agent, and not through OpenXDAS:

00031665,Resource Provisioning
 00031666,Resource Provisioning Failure

00031600,Role Provisioning
 00031601,Role Provisioning Failure

00031677,Create Resource Association Failure
 00031678,Delete Resource Association
 00031679,Delete Resource Association Failure
 0003167A,Modify Resource Association
 0003167B,Modify Resource Association Failure

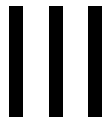
#^GROUP^Engine events logged from vrdim^00030001-00030032

00030001,Status Success
 00030002,Status Retry
 00030003,Status Warning
 00030004,Status Error,Channel
 00030005,Status Fatal
 00030006,Status Other
 00030007,Search
 00030008,Add Entry
 00030009,Delete Entry,Channel
 0003000A,Modify Entry
 0003000B,Rename Entry
 0003000C,Move Entry
 0003000D,Add Association
 0003000E,Remove Association
 0003000F,Query Schema
 00030010,Check Password
 00030011,Check Object Password
 00030012,Change Password
 00030013,Sync,Channel
 00030014,Input XML Document
 00030015,Input Transformation Document
 00030016,Output Transformation Document
 00030017,Event Transformation Document
 00030018,Placement Rule Transformation Document
 00030019,Create Rule Transformation Document
 0003001A,Input Mapping Rule Transformation Document
 0003001B,Output Mapping Rule Transformation Document
 0003001C,Matching Rule Transformation Document
 0003001D,Command Transformation Document
 0003001E,Publisher Filter Transformation Document
 0003001F,User Agent Request
 00030020,Resync Driver
 00030021,Migrate
 00030022,Driver Start
 00030023,Driver Stop
 00030024>Password Sync
 00030025>Password Reset
 00030026,DirXML Error
 00030027,DirXML Warning
 00030028,Custom Operation
 00030029,Clear Attribute
 0003002A,Add Value - Modify Entry
 0003002B,Remove Value
 0003002C,Merge Entries
 0003002D,Get Named Password
 0003002E,Reset Attributes
 0003002F,Add Value - Add Entry

```
#^GROUP^Job events logged from vrdim^000303E4-000303E7
000303E4,Job Result Aborted
000303E5,Job Result Error
000303E6,Job Result Warning
000303E7,Job Result Success
```

```
#
#^GROUP^Server events Logged from DXevent^000307D0-000307E2
000307D0,Config:Log Events
000307D1,Config:Driver Cache Limit
000307D2,Config:Driver Set
000307D3,Config:Driver Start Option
000307D4,Driver Resync
000307D5,Migrate Application
000307D6,Shim Password Set
000307D7,Keyed Password Set
000307D8,Remote Loader Password Set
000307D9,Regenerate Key Pair
000307DA,Get Server Certificate
000307DB,Cache Utility
000307DC,Check Object Password
000307DD,Initialize Driver Object
000307DE,Notify Job Update
000307DF,Open Driver Action
000307E0,Queue Driver Event
000307E1,Start Job
000307E2,Abort Job
```

```
#^GROUP^Remote Loader^00030BB8-00030BBB
00030BB8,Remote Loader Start
00030BB9,Remote Loader Stop
00030BBA,Remote Loader Connection Established
00030BBB,Remote Loader Connection Dropped
```

Administering the User Application

These sections describe how to configure and manage the Identity Manager User Application by using the **Administration** tab of the user interface.

4 Using the Administration Tab

This section introduces you to the **Administration** tab of the Identity Manager user interface. You'll learn how to use the **Administration** tab to configure and manage the Identity Manager User Application.

4.1 About the Administration Tab

The Identity Manager user interface is primarily accessed by end users, who work with the tabs and pages it provides for identity self-service and workflow-based provisioning. However, this browser-based user interface also provides an **Administration** and page, which administrators can use to access a page and configure various characteristics of the underlying Identity Manager User Application.

For example, choose the **Administration** to:

- ♦ Change the theme used for the look and feel of the user interface
- ♦ Customize the identity self-service features available to end users
- ♦ Specify who is allowed to perform administration actions
- ♦ Manage other details about the User Application and how it runs

4.2 Who Can Use the Administration Tab

The **Administration** tab is not visible to typical end users of the Identity Manager user interface. There are three kinds of users who can see and access this tab:

User Application Administrators: A User Application Administrator is authorized to perform all management functions related to the Identity Manager User Application. This includes accessing the **Administration** tab of the Identity Manager user interface to perform any administration actions that it supports. During installation, a user is specified as User Application Administrator. After installation, that user can use the Security page on the **Administration** tab to specify other User Application administrators, as needed. For details, see [Section 5.1.7, "Assigning the User Application Administrator," on page 128](#).

Domain Administrators and Domain Managers: Domain Administrators and Domain Managers are authorized to perform provisioning and security tasks for the Identity Manager User Application. For details, see [Chapter 7, "RBPM Provisioning and Security Configuration," on page 161](#).

4.3 Accessing the Administration Tab

When you are a User Application Administrator (or other permitted user), you can use any supported Web browser to access the **Administration** tab of the Identity Manager user interface to manage the Identity Manager User Application.

For a list of supported Web browsers, see the "[System Requirements for the Identity Applications](#)" in the [NetIQ Identity Manager Setup Guide](#).

NOTE: To use the Identity Manager user interface, make sure your Web browser has JavaScript* and cookies enabled.

To access the Administration functionality using the Identity Manager Home page:

- 1 Log in to the Identity Manager Home page using the OSP login as a User Application Administrator.

After you log in, the Identity Manager Home page displays. This page provides default links to the basic tasks an administrators can perform in Identity Manager. For more information, see [“Accessing Identity Manager Home”](#) in the *NetIQ Identity Manager Home and Provisioning Dashboard User Guide*.

- 2 Under the Administration section, click the appropriate link.

To access the Administration functionality directly with a URL:

- 1 In your Web browser, go to the URL for the Identity Manager user interface as configured in your environment, with the context you used when you installed the User Application.

For example:

```
http://myappserver:8080/IDMProv
```

- 2 Specify the username and password of a User Application Administrator (or a user with some Administration permissions), then click **Login**.
- 3 Click the **Administration** tab.

For more general information about accessing and working in the Identity Manager user interface, see the *Identity Manager User Application: User Guide*.

4.4 Administration Tab Actions You Can Perform

After you're on the **Administration** tab, you can use any available actions to configure and manage the Identity Manager User Application. [Table 4-1](#) contains a summary.

Table 4-1 Administration Tab Actions Summary

Action	Description
Application Configuration	Controls User Application configuration of caching, logging, password management, and LDAP connection parameters. Provides read-only information about the driver status and the portal. Provides access to tools that allow you to export or import portal content (pages and portlets used in the Identity Manager User Application). For details, see Chapter 5, “Application Configuration,” on page 109.
Page Admin	Controls the pages displayed in the Identity Manager user interface and who has permission to access them For details, see Chapter 6, “Page Administration,” on page 145.
RBPM Provisioning & Security	Controls the provisioning configuration, as well as security permissions and navigation access. For details, see Chapter 7, “RBPM Provisioning and Security Configuration,” on page 161.

5 Application Configuration

This section describes the tasks that you can perform from the Application Configuration page.

5.1 Portal Configuration Tasks

NOTE: The portal functionality within the User Application has been deprecated in Identity Manager 4.0.2.

5.1.1 Caching Management

You can use the Caching page to manage various caches maintained by the Identity Manager User Application. The User Application employs these caches to store reusable, temporary data on the application server so it can optimize performance.

You have the ability to control these caches when necessary by flushing their contents and changing their configuration settings.

Flushing caches

The caches are named according to the subsystems that use them in the Identity Manager User Application. Normally, you don't need to flush them yourself, because the User Application does that automatically based on how frequently their data is used or when the source data changes. However, if you have a specific need, you can manually flush selected caches or all caches.

- 1 Go to the Caching page.
- 2 In the **Flush Cache** section of the page, use the drop-down list to select a particular cache to flush (or select **Flush all**).

The list of available caches is dynamic; it changes depending on what data is cached at the moment.

- 3 Click **Flush Cache**.

Flushing the Directory Abstraction Layer Cache

The User Application's directory abstraction layer also has a cache. The DirectoryAbstractLayerDefinitions cache stores abstraction layer definitions on the application server to optimize performance for all data model operations.

In a typical situation, the User Application automatically keeps the DirectoryAbstractLayerDefinitions cache synchronized with the abstraction layer definitions stored in the Identity Vault. But, if necessary, you can manually flush the DirectoryAbstractLayerDefinitions cache as described in "[Flushing caches](#)" on page 109 to force the latest definitions to be loaded from the Identity Vault.

For more information on the User Application's directory abstraction layer, see the *Identity Manager User Application: Design Guide*.

Flushing Caches in a Cluster

Cache flushing is supported in both clustered and non-clustered application server environments. If your application server is part of a cluster and you manually flush a cache, that cache is automatically flushed on every server in the cluster.

Configuring Cache Settings

You can use the Caching page to display and change cache configuration settings for a clustered or non-clustered application server environment. Your changes are saved immediately, but they don't take effect until the next User Application restart.

TIP: To restart the User Application, you can reboot the application server; redeploy the application (if the WAR has been changed in some way); or force the application to restart (as described in your application server's documentation).

How Caching Is Implemented

In the Identity Manager User Application, caching is implemented via JBoss Cache. JBoss Cache is an open source caching architecture that's included with the JBoss Application Server but also runs on other application servers.

How Cache Settings Are Stored

Two levels of settings are available for controlling cache configuration: global, and local. Use these settings to customize the caching behavior of the Identity Manager User Application. [Table 5-1 on page 110](#) describes the cache configuration settings.

Table 5-1 Cache Configuration Settings

Level	Description
Global settings	<p>Global settings are stored in a central location (the Identity Vault) so that multiple application servers can use the same setting values. For example, someone with a cluster of application servers would typically use global settings for the cluster configuration values.</p> <p>To find the global settings in your Identity Vault, look for the following object under your Identity Manager User Application driver:</p> <pre>configuration.AppDefs.AppConfig</pre> <p>For example:</p> <pre>configuration.AppDefs.AppConfig.MyUserApplicationDriver.MyDriverSet.MyOrg</pre> <p>The XmlData attribute of the configuration object contains the global settings data.</p>

Level	Description
Local setting	<p>Local settings are stored separately on each application server so that an individual server can override the value of one or more global settings. For example, you might want to specify a local setting to remove an application server from the cluster specified in the global settings, or to reassign a server to a different cluster.</p> <p>To find the local settings on your JBoss application server, look for the following file under your JBoss server configuration's <code>conf</code> directory: <code>sys-configuration-xmldata.xml</code>. For example: <code>jboss/server/IDMProv/conf/sys-configuration-xmldata.xml</code>.</p> <p>To find the local settings on your WebSphere application server, look for the <code>sys-configuration-xmldata.xml</code> file at the location you specified in the <code>extend.local.config.dir</code> property that you set at installation.</p> <p>If your server has local settings, that data is contained in this file. If no local settings have been specified, the file won't exist.</p>

You should think of global settings as the default values for every application server that uses a particular instance of the User Application driver. When you change a global setting, you are affecting each of those servers (at the next User Application restart), except for those cases where an individual server specifies a local override.

How Cache Settings Are Displayed

The Caching page displays the current cache settings (from the latest User Application restart). It also displays the corresponding global and local values of those settings, and lets you change them (for use at the next User Application restart).

The global settings always have values. The local settings are optional.

Basic Cache Settings

These cache settings apply to both clustered and non-clustered application servers.

To configure basic cache settings:

- 1 Go to the Caching page.
- 2 In the **Cache Configuration** section of the page, specify global or local values for the following settings, as appropriate:

Setting	What to do
Lock Acquisition Timeout	Specify the time interval (in milliseconds) that the cache waits for a lock to be acquired on an object. You might want to increase this setting if the User Application gets a lot of lock timeout exceptions in the application log. The default is 15000 ms.
Wake Up Interval Seconds	Specify the time interval (in seconds) that the cache eviction policy waits before waking up to do the following: <ul style="list-style-type: none"> ◆ Process the evicted node events ◆ Clean up the size limit and age-out nodes

Setting	What to do
Eviction Policy Class	<p>Specify the classname for the cache eviction policy that you want to use. The default is the LRU eviction policy that JBoss Cache provides:</p> <pre>org.jboss.cache.eviction.LRUPolicy</pre> <p>If appropriate, you can change this to another eviction policy that JBoss Cache supports.</p>
Max Nodes	<p>Specify the maximum number of nodes allowed in the cache. For no limit, specify:</p> <pre>0</pre> <p>You can customize this setting for some cache holders. See “Customizable Cache Holders” on page 112.</p>
Time To Live Seconds	<p>Specify the time to idle (in seconds) before the node is swept away. For no limit, specify:</p> <pre>0</pre> <p>You can customize this setting for some cache holders. See “Customizable Cache Holders” on page 112.</p>
Max Age	<p>Specifies the number of seconds an entry should be allowed to stay in the cache holder since its creation time. For no time limit, specify:</p> <pre>0</pre> <p>This setting is only available for “Customizable Cache Holders” on page 112.</p>

These settings are required, which means that there must be a global value for each, and optionally a local value too.

If you want to override the global value of a setting with a local value, select the **Enable Local** check box for that setting. Then specify the local value. (Make sure that all of your local values are valid. Otherwise, you won't be able to save your changes.)

NOTE: For those settings where **Enable Local** is deselected, any existing local values are deleted when you save.

- 3 Click **Save**.
- 4 When you're ready for your saved settings to take effect, restart the User Application on the applicable application servers.

Customizable Cache Holders

You can customize the **Max Nodes**, **Time To Live**, and **Max Age** settings for some cache holders. The cache holders are listed in [Table 5-2](#).

Table 5-2 Customizable Cache Holders

Cache Holder Name	Description
DirectoryAbstractionLayerDefinitions	Caches the Directory Abstraction Layer definitions to optimize performance for all data model operations. See “Flushing the Directory Abstraction Layer Cache” on page 109.
DirectoryService.ContainerCacheHolder	Caches containers in the directory layer. Containers are shared by many users and groups, and reading them from the directory layer involves both network communication (with the LDAP server) and object creation. By default, the cache is limited to 50 containers, and the LRUs have a default Time To Live (TTL) of 10 minutes. Depending on the directory topography in your enterprise, you might need to adjust the maximum number of nodes or the TTL if you find the performance is suffering because of queries to the LDAP server for container objects. Making settings too high in combination with a large number of usable containers can cause unneeded memory consumption and net lower performance from the server.
DirectoryService.DeIProxyRuntimeServiceDelegate	Caches delegate assignments.
DirectoryService.DeIProxyRuntimeService.Delegation	Caches user availability settings.
DirectoryService.DeIProxyRuntimeService.Delegator	Caches the delegator entities.
DirectoryService.DeIProxyRuntimeService.Proxy	Caches proxy assignments.
DirectoryService.GroupCacheHolder	Caches groups in the directory layer. Groups are often shared by many users, and reading them from the directory layer involves both network communication (with LDAP server) and object creation. By default, the cache is limited to 500 groups, and the LRUs have a default TTL of 10 minutes. Depending on the user/group topography in your enterprise, you might need to adjust the maximum number of nodes or the TTL if you find the performance is suffering because of queries to the LDAP server for groups objects. Settings that are too high, in combination with a large number of usable groups, can cause unneeded memory consumption, and net lower performance from the server.

Cache Holder Name	Description
DirectoryService.MemberhipCacheHolder	Caches the relationship between a user and a set of groups. Querying the set of groups a user belongs to can be a network and CPU intensive operation on the LDAP server, especially if dynamic groups are enabled. For this reason, relationships are cached with an expiration interval so that changes in the criteria for inclusion/exclusion in a group (such as time-based dynamic groups) are reflected. The default Max Age is five minutes. However, if you use dynamic groups which have a requirement for finer grained time control, then you can adjust the Max Age on this cache holder to be just below the minimum time your finest grained time based dynamic group requires. The lower this value is, the more times the user's groups are queried during a session. Setting a value too high keeps the user/group relationships in memory perhaps longer than the user's session needlessly consuming memory.
DirectoryService.RolesMembershipCacheHolder	Caches the application role membership list by role.
DirectoryService.TeamManagerRuntime.Team	Caches the application team instances and team provisioning requests.
DirectoryService.UserCacheHolder	Caches users in the directory layer. Reading users from the directory layer involves both network communication (with LDAP server) and object creation. By default, the cache is limited to 1000 users, and the LRUs have a default TTL of 10 minutes. Depending on the user topography in your enterprise, you might need to adjust the maximum number of nodes or the TTL if you find the performance is suffering because of queries to the LDAP server for user objects. Making settings too high combined with a large number of different users logging in can cause unneeded memory consumption, and net lower performance from the server.
GlobalCacheHolder	The general purpose cache holder. This configuration applies to all caches that are not customizable (that is, all cache holders not listed in this table.)
JUICE	<p>Caches the resource bundles used by the user interface controls and DN display expression lookup results. Changing the setting of the cache holder has a performance impact for the DN display expression lookups because they are frequently used in the User Application.</p> <p>The low value should be at least 300 seconds, but a higher value than 900 seconds is ok. A lower value should be used if the customer is frequently changing the attributes that are used in the DN display expression</p>

Cache Holder Name	Description
RoleManager.RolesCacheHolder	Caches user role memberships listed by user.
Workflow.Model.Process	Caches the provisioning process XML object structure.
Workflow.Model.Request	Caches the provisioning request XML object structure.
Workflow.Provisioning	Caches provisioning request instances that have not completed. The default maximum capacity for the LRU cache is 500. The capacity can be modified by clicking the Administration/Provisioning and choosing the Engine and Cluster settings. The Process Cache Maximum Capacity appears on this page. This cache reduces the memory footprint for workflow processing without compromising performance.

Cache Settings for Clusters

This section discusses how to configure caching when you run the Identity Manager User Application across a cluster of application servers.

In the Identity Manager User Application, cluster support for caching is implemented via *JGroups*. JGroups is an open-source clustering architecture that's included with the JBoss Application Server but also runs on other application servers.

The User Application's cluster consists of nodes on a network that run JGroups and use a common Group ID. By default, the Group ID provided for the User Application's cluster is a UUID that looks like this:

```
c373e901aba5e8ee9966444553544200
```

The UUID helps ensure uniqueness, so that the Group ID of the User Application's cluster doesn't conflict with the Group IDs of other clusters in your environment. For instance, the JBoss Application Server itself uses several JGroups clusters and reserves associated names including the Group IDs DefaultPartition and Tomcat-Cluster for them.

How Caching Works with a Cluster

When you start the User Application, the application's cluster configuration settings on the **Caching** page determine whether to participate in a cluster and invalidates cache changes in the other nodes in that cluster. If clustering is enabled, the User Application accomplishes this by sending cache entry invalidation messages to each node as changes occur.

In most cases, you should use global settings when configuring a cluster. However, global settings might be a problem if you need to use TCP, because the IP address of the server must be specified in the JGroups initialization string for each server. You can use local settings to specify a JGroups initialization string. For more information, see ["Configuring Cache Settings for Clusters" on page 116](#).

Preparing to Use a Cluster

To use caching across a cluster:

- 1 Set up your JGroups cluster. This involves using the User Application installation program to install the Identity Manager User Application to each application server in the cluster (see [Section 2.5, “Clustering,” on page 60](#)).
- 2 Enable the use of that cluster in the User Application’s cache configuration settings
See [“Configuring Cache Settings for Clusters” on page 116](#).

Configuring Cache Settings for Clusters

After you have a cluster ready to use, you can specify settings for the support of caching across that cluster.

- 1 Go to the Caching page.
- 2 In the **Cluster Configuration** section of the page, specify global or local values for the following settings, as appropriate:

Setting	What to do
Cluster Enabled	Select True to invalidate cache changes to the other nodes in the cluster specified by Group ID. If you don’t want to participate in a cluster, select False .
Group ID	<p>Specify the Group ID of the JGroups cluster in which you want to participate. There’s no need to change the default Group ID that’s provided for the User Application’s cluster, unless you want to use a different cluster.</p> <p>The Group ID must be unique and must not match any of the known JBoss cluster names such as DefaultPartition and Tomcat-Cluster.</p> <p>TIP: To see the Group ID in logging messages, make sure that the level of the caching log (<code>com.sssw.fw.cachemgr</code>) is set to Info or higher.</p>
Cluster Properties	<p>Specify the JGroups protocol stack for the cluster specified by Group ID. This setting is for experienced administrators who might need to adjust the cluster properties. Otherwise, you should not change the default protocol stack.</p> <p>To see the current cluster properties, click view.</p> <p>For information on JGroups protocol stack, see Section 2.2.6, “Encryption of Sensitive User Application Data,” on page 48.</p>

If you want to override the global value of a setting with a local value, select the **Enable Local** check box for that setting. Then specify the local value. For those settings where **Enable Local** is unselected, any existing local values are deleted when you save.

WARNING: If you specify local settings and enter an incorrect configuration in the JGroups initialization string, the cache cluster function might not start. Unless you know how to configure JGroups correctly and understand the protocol stack, you should not use local settings.

Make sure that all nodes in your cluster specify the same Group ID and Cluster Properties. To see these settings for a particular node, you must access the Identity Manager user interface running on that node—by browsing to the URL of the user interface on that server—and then display the Caching page there.

To use the TCP protocol instead of the default UDP protocol, you can add a token to the global settings for the **Cluster Properties**. For example, `IDM_HOST_ADDR`. You can then edit the `hosts` file on each server in the cluster to specify the IP address for that server. For more information, see [“Configuring User Application Caching to use TCP” on page 117](#).

- 3 Click **Save**.
- 4 When you're ready for your saved settings to take effect, restart the User Application on the applicable application servers.

Configuring User Application Caching to use TCP

You can configure caching for the User Application to use TCP. The configuration process must be completed on each server in the cluster because the Local Settings are saved on the file system for each server.

To configure User Application caching to use TCP:

- 1 Log in as the User Application Administrator and go to Administration->Application Configuration->Caching.
- 2 Enable the **Enable Local** checkbox in the **Cluster Enabled** row and set `Local=TRUE`. And for each of the individual properties in the following steps enable the **Enable Local** checkbox for that property and specify a value in the textfield in the **Local** column. The Local value for the property will then override the Global value.
- 3 Copy this string and paste it in to the **Cluster Properties** field. It is very important to paste as a single string with no carriage returns embedded:

```
TCP(bind_addr=164.99.208.68;start_port=7815;loopback=true):TCPPING(initial_hosts=164.99.208.68[7815],164.99.208.36[7815];port_range=3;timeout=3500;num_initial_members=3;up_thread=true;down_thread=true):MERGE2(min_interval=5000;max_interval=10000):FD(shun=true;timeout=2500;max_tries=5;up_thread=true;down_thread=true):VERIFY_SUSPECT(timeout=1500;down_thread=false;up_thread=false):pbcast.NAKACK(down_thread=true;up_thread=true;gc_lag=100;retransmit_timeout=3000):pbcast.STABLE(desired_avg_gossip=20000;down_thread=false;up_thread=false):pbcast.GMS(join_timeout=5000;join_retry_timeout=2000;shun=false;print_local_addr=true;down_thread=true;up_thread=true):pbcast.STATE_TRANSFER(up_thread=true;down_thread=true)
```

The properties in this string are defined by JBoss. Refer to JBoss documentation for more information.

- 4 Set `bind_addr` to the local host IP address of the server you are logged into.
- 5 Next you need to set the `start_port`. This value must take into account ports already in use as well as the value for `port_range` in order to avoid port conflicts. Depending on your configuration you may need to troubleshoot to find an unused port.
- 6 Change the IP addresses for TCPPING to include the IP addresses of all the nodes in the cluster and their `start_port` values. The list should begin with the local IP address.
- 7 Save changes. These changes are written to the local file system for your server. Remember to make these changes for all servers in the cluster. Any server that does not have these changes will use the Global Settings values.
- 8 Restart the server.

5.1.2 Driver Status

You can use the Driver Status pane to determine the expiration status of your driver.

The Driver Status pane displays the following two entries:

- ◆ Driver Name
- ◆ Expiration Date

The Expiration Date displays one of the following values:

1. Unlimited (if the activation has occurred)
2. Expiration date of the driver (if the driver is a trial driver)

5.1.3 Identity Vault Settings

You can use the Identity Vault Settings pane to:

- ◆ Change the credentials used by the Identity Manager User Application when connecting to the Identity Vault (LDAP provider)
- ◆ Change the credentials for the guest account, if your system is configured to use a specific guest account, rather than LDAP anonymous account.
- ◆ View other LDAP properties of the Identity Manager User Application. The values of these settings are determined when you install the User Application.

The user interface displays different fields depending on how you configured the guest account during installation. If you specified a guest account, the user interface includes fields that let you update the credentials for that account. If you have configured your system to use the LDAP Public Anonymous account, the user interface displays this message: `The application is configured to use public anonymous account. To use a specific guest account, enable the guest account using the ldap configuration tool.`

To administer Identity Vault settings:

- 1 On the Application Configuration page, select **Identity Vault Settings** from the navigation menu on the left.
- 2 Examine and modify the settings, as appropriate. For details, see: [“LDAP Settings You Can Change” on page 118.](#)
- 3 If you make changes that you want to apply, click **Submit**.

LDAP Settings You Can Change

On the Identity Vault Connection Settings panel, you can modify settings for the credentials for:

- ◆ The Identity Manager User Application whenever it connects to the Identity Vault (LDAP provider).
- ◆ The guest account (if configured).

The initial values for the credentials are specified during installation. These installation values are written to the `sys-configuration.xml` data file. If you make changes to these credentials via the Administration tab, your changes are saved to the User Application’s database; they are not saved to the `sys-configuration.xml` data file. After values are written to the database, the User Application no longer checks the values written to the `sys-configuration.xml` data file. This means that you cannot use the `configupdate` utility to change the credentials because they are ignored. However, you can use `configupdate` to change the type of guest user (LDAP Guest or Public Anonymous Account).

Table 5-3 LDAP Parameters

Setting	What to do
Identity Vault Administrator	<p>Type the name of a user who has full administrator rights in the Identity Vault. The Identity Manager User Application needs to access the Identity Vault as an administrator in order to function.</p> <p>It is typical to specify the Identity Vault's <code>root</code> administrator as the LDAP connection username. The <code>root</code> administrator has full control over the tree, so you need not assign any special trustee rights.</p> <p>For example:</p> <pre>cn=admin,o=myorg</pre> <p>If you specify some other user, you need to assign inheritable trustee rights to the properties [All Attributes Rights] and [Entry Rights] on your User Application driver.</p> <p>NOTE: To avoid confusion, it is recommended that you do not specify the User Application's User Application Administrator as the LDAP connection username. It is best to use separate accounts for these two different purposes.</p>
Identity Vault Administrator Password	Type the password that is currently set for that username in the Identity Vault.
and	
Confirm Identity Vault Administrator Password	
Guest Username	Type the guest user's distinguished name
Confirm Guest Password	Type the password for the guest user.

If TLS is enabled for your LDAP server, you might encounter the following error when you update the Admin username and password: `Unable to authenticate to LDAP Provider`. Disable this error by disabling TLS via `iManager`.

5.1.4 Logging Configuration

You can use the Logging page to control the levels of logging messages you want the Identity Manager User Application to generate. For information about understanding how logging works in an identity applications environment, see [Chapter 3, "Setting Up Logging," on page 75](#).

The Identity Manager User Application implements logging by using a custom-developed logging framework that integrates with `log4j`, an open-source logging package distributed by The Apache Software Foundation. By default, event messages are logged to both of the following:

- ◆ The system console of the application server where the Identity Manager User Application is deployed
- ◆ A log file on that application server. For example:

```
/opt/netiq/idm/apps/tomcat/logs/catalina.out
```

This is a rolling log file; after it reaches a certain size, it rolls over to another file.

Logs for all identity applications components including OSP are logged to the `catalina.out` file.

OSP logs are also stored in a separate file, `osp-idm-<date of log generation>.log` file located in `/opt/netiq/idm/apps/tomcat/logs/` on a Tomcat server. Logging is turned off by default and must be enabled in the `setenv.sh` file in the `/TOMCAT_INSTALLED_HOME/bin/` directory. By default, logs for User Application and Role and Resource Service drivers are added to DSTrace.

About the Logs

The Logging page lists a variety of logs, each outputting event messages from a different part of the Identity Manager User Application. Each log has its own independent output level.

The log names are based on log4j conventions. You'll see these log names in the event messages that are generated, indicating the context of the message output.

[Table 3-1 on page 76](#) lists and describes some of the logs.

Table 5-4 Identity Manager User Application Logs

Log Name	Description
<code>com.novell</code>	Parent of other Identity Manager User Application logs
<code>com.novell.afw.portal.aggregation</code>	Messages related to portal page processing
<code>com.novell.afw.portal.persist</code>	Messages related to the persistence of portal data (including portal pages and portlet registrations)
<code>com.novell.afw.portal.portlet</code>	Messages from the portal core portlets and accessory portlets
<code>com.novell.afw.portal.util</code>	Messages from the portal import/export and navigation portlets
<code>com.novell.afw.portlet.consumer</code>	Messages related to portlet rendering
<code>com.novell.afw.portlet.core</code>	Messages related to the core portlet API
<code>com.novell.afw.portlet.persist</code>	Messages related to the persistence of portlet data (including portlet preferences and setting values)
<code>com.novell.afw.portlet.producer</code>	Messages related to the registration and configuration of portlets within the portal
<code>com.novell.afw.portlet.util</code>	Messages related to utility code used by portlets
<code>com.novell.afw.theme</code>	Messages from the theme subsystem
<code>com.novell.afw.util</code>	Messages related to portal utility classes
<code>com.novell.soa.af.impl</code>	Messages from the approval flow (provisioning workflow) subsystem
<code>com.novell.srvprv.apwa</code>	Messages from the Web application (actions and tags)
<code>com.novell.srvprv.impl.portlet.core</code>	Messages from the core identity portlets and password portlets
<code>com.novell.srvprv.impl.portlet.util</code>	Messages from the identity-related utility portlets
<code>com.novell.srvprv.impl.servlet</code>	Messages from the UI control framework's ajax servlet and ajax services

Log Name	Description
<code>com.novell.srvprv.impl.uictrl</code>	Messages from the UI control registry API and approval form rendering
<code>com.novell.srvprv.impl.vdata</code>	Messages from the directory abstraction layer
<code>com.novell.srvprv.spi</code>	Messages from the UI control registry API
<code>com.sssw.fw.cachemgr</code>	Messages related to the framework cache subsystem
<code>com.sssw.fw.core</code>	Messages related to the framework core subsystem
<code>com.sssw.fw.directory</code>	Messages related to the framework directory subsystem
<code>com.sssw.fw.event</code>	Messages related to the framework event subsystem
<code>com.sssw.fw.factory</code>	Messages related to the framework factory subsystem
<code>com.sssw.fw.persist</code>	Messages related to the framework persistence subsystem
<code>com.sssw.fw.resource</code>	Messages related to the framework resource subsystem
<code>com.sssw.fw.security</code>	Messages related to the framework security subsystem
<code>com.sssw.fw.server</code>	Messages related to the framework server subsystem
<code>com.sssw.fw.servlet</code>	Messages related to the framework servlet subsystem
<code>com.sssw.fw.session</code>	Messages related to the framework session subsystem
<code>com.sssw.fw.usermgr</code>	Messages related to the framework user subsystem
<code>com.sssw.fw.util</code>	Messages related to the framework utility subsystem
<code>com.sssw.portal.manager</code>	Messages related to the Portal Manager
<code>com.sssw.portal.persist</code>	Messages related to portal persistence

The User Application logs are hierarchical. For example, `com.novell` is the parent of other logs underneath it. Any additional logs inherit its properties.

Changing Log Levels

Logging requirements vary widely; therefore, NetIQ cannot make recommendations regarding what information you need in each package that suits your business needs. You can control the amount of information that is written to a particular log by changing the level that is set for it. By default, all logs are set to *Info*, which is an intermediate level.

- 1 Go to the Logging page.
- 2 At the top of the page, find a log whose level you want to change.
- 3 Use the drop-down list to select one of the following levels:

Level	Description
Fatal	The least detail. Writes fatal errors to the log.
Error	Writes errors (plus all of the above) to the log.
Warn	Writes warnings (plus all of the above) to the log.
Info	Writes informational messages (plus all of the above) to the log.
Debug	Writes debugging information (plus all of the above) to the log.
Trace	The most detail. Writes tracing information (plus all of the above) to the log.

4 Repeat [Step 2](#) and [Step 3](#) for other logs, as needed.

5 Click **Submit**.

You can change the log level for all of the logs to one setting by selecting **Change log level** of all above logs and using the drop-down list to select the level.

Adding Logs for Other Packages

You can add logs for other packages used by the User Application.

- 1 Go to the Logging page:
- 2 At the bottom of the page, select **Add Log Level for Package**, then use the drop-down list to select the package.
- 3 Choose a log level from the drop-down, then click **Submit**.

Sending Log Messages to an Auditing Service

You can use the Logging page to control whether the Identity Manager User Application sends event message output to an auditing service. Logging is off by default, unless you turn it on when installing the User Application.

To toggle logging on/off:

- 1 Go to the Logging page.
- 2 Select or deselect the following settings, as appropriate:
 - ◆ **Also send logging messages to audit service**
 - ◆ **Also send logging messages to OpenXDAS**
- 3 Click **Submit**.

Persisting Your Log Settings

By default, changes you make on the Logging page stay in effect until the next application-server restart or User Application redeployment. After that, the log settings revert to their default values.

However, the Logging page does offer you the option of persisting your changes to its settings. If you turn on this feature, values for the log settings are stored in a logging configuration file on the application server where the Identity Manager User Application is deployed. For example:

- ◆ On JBoss, this file is in the following location by default:

jboss/server/IDMProv/conf/idmuserapp_logging.xml

- ◆ On WebSphere, the location of this file is specified according to the custom property named `idmuserapp.logging.config.dir`.

To toggle persistence of settings on or off:

- 1 Go to the Logging page.
- 2 Select or deselect the following setting, as appropriate: **Persist the logging changes**
- 3 Click **Submit**.

5.1.5 Portal Settings

You can use the Portal page to view characteristics of the Identity Manager User Application. The settings are for informational purposes and cannot be changed. The values of these settings are set in the User Application WAR. (**Default Theme** reflects your current theme choice from the Themes page.)

5.1.6 Theme Administration

You can use the Themes page to control the look and feel of the Identity Manager user interface.

A theme is a set of visual characteristics that apply to the entire user interface (including the guest and login pages, the **Identity Self-Service** tab, the **Work Dashboard** tab, and the **Administration** tab). There's always just one theme in effect for the user interface. The Themes page offers a choice of several themes, in case you want to switch to a different one.

The Themes page also enables you to:

- ◆ Preview each theme choice to see how it looks
- ◆ Customize any theme choice to reflect your own branding (such as a logo)

Previewing a Theme

Before choosing a theme, you can preview how it will change the look of the Identity Manager user interface.

- 1 Go to the Themes page.

The following themes are supported in this release:

- ◆ BlueGloss
- ◆ Neptune (new theme introduced in this release)

Several of the themes introduced in earlier versions of the User Application have been deprecated in this release. The following themes have been deprecated:

- ◆ Manilla
- ◆ Linen
- ◆ Medico
- ◆ IDMStandard

These themes are no longer supported with the current release. You cannot select any of these themes on the **Theme Administration** page on the **Administration** tab.

The Manilla, Linen, Medico, and IDMStandard themes will most likely be removed in a future release. If you use any of these themes, you should migrate them to this release of the User Application. If you use a custom theme that is based on one of the deprecated themes, you need to follow these steps to migrate the theme:

- 1a** Look inside the theme.css for your custom theme and copy any custom selectors (new or edited) from this theme into either the BlueGloss or Neptune theme.
 - 1b** Save a new custom theme, which now includes your customizations as well as selectors from the BlueGloss or Neptune theme.
- 2** Find a theme that you are interested in, then click the corresponding **Preview** button.
The preview for that theme displays in a new browser window.
- 3** Scroll through the preview to see the characteristics of this theme.
- 4** When you're done, click **Close Preview Page** (in the top left corner) or close the preview window manually.

Choosing a Theme

When you find a theme that you like, you can choose to make it the current theme for the Identity Manager user interface.

- 1** Go to the Themes page.
- 2** Click the radio button for the theme you want.
- 3** Click the **Save** button.

The look of the user interface changes to reflect your chosen theme.

Customizing a Theme's Branding

You can tailor any theme by substituting your own images and changing some color settings. This enables you to give the Identity Manager user interface a custom look to meet the branding requirements of your company or organization.

- 1** Go to the Themes page.
- 2** Find a theme that you want to customize, then click the corresponding **Customize** button.
The Themes page displays the Customize Branding settings for that theme.
- 3** Specify your customizations by changing the settings in one or more tabs (as needed). Each contains the settings for different parts of the User Application interface. They include:
 - ◆ **General:** Lets you specify general theming properties such as a favorites icon, background, link and hover color, and the left navigation area properties.
 - ◆ **Header:** Lets you specify the header color, texture, logo and username properties.

NOTE: The Left Background image needs to be the size indicated on the Header page (which defaults to 272 x 79 pixels) in order to display properly. The user interface does not attempt to resize the image automatically. For example, it will not stretch the image if it is too small.

- ◆ **Header tabs:** Lets you specify the properties for the header tabs.
- ◆ **Admin subnavigation:** Lets you specify the properties for the **Admin** tab.

Follow the on-screen instructions for specifying each setting. The changes are not reflected in the User Application until you save them. If you have made unsaved changes, the **Save** button displays an asterisk * to indicate that the changes are pending a save.

4 Click **Save**.

If you're editing the current theme, the look of the user interface changes to reflect your customizations. If you want to undo all of your customizations to the theme, click the *Reset* button.

5 When you're done working on this theme, click **Back to Theme Selector**.

Defining a Custom Theme

You can also create and deploy your own custom themes and deploy them in their own WAR file. When they are deployed, the custom themes are available through the Themes management page of the **Administration** tab. Before attempting to create your own custom theme, make sure you have a working knowledge of the following technologies:

- ♦ The structure of J2EE WAR files, how to modify the contents of a WAR file, and how to deploy one to your application server.
- ♦ How to modify CSS and XML files
- ♦ How to create the graphic elements for your theme

Creating a Custom Theme

To create a custom theme, begin with a copy of an existing theme (such as **BlueGloss**) from the User Application WAR:

1 Back up the deployed User Application WAR file (`IDMProv.WAR`) to the directory in which you install, for example the `/opt/netiq/idm` subdirectory.

2 In a test environment, extract the contents of the User Application WAR file.

The files that comprise the User Application's themes are located in the `resource\themes` subdirectory. Each theme resides in its own directory with an appropriate name.

3 In the test environment, create a directory for the custom theme.

The directory name can be any valid directory name, but it should reflect the name of the theme, and it should not contain spaces.

4 Copy the contents of the BlueGloss theme from the extracted WAR file to the new subdirectory. You will be working with the following files:

File Name	Description
<code>theme.xml</code>	The theme descriptor file. It includes entries for display name and description. They are used in the Themes page of the Administration tab. The remaining entries correspond to the brandable selectors. The width and height attributes on these entries are used in the branding page to reference the exact dimensions needed when a user uploads a customized version of these images. These entries must match their respective images, width and height as found in the <code>themes.css</code> .
<code>theme.css</code>	Contains the CSS selectors used to style the look and feel of the user interface.
<code>print.css</code>	Contains the CSS selectors used to style a print friendly version of the user interface.
<code>dojo.css</code>	Contains a pointer to additional CSS files used by RBPM.
An images subdirectory	Contains the images used by the theme.

Rules for working with these files:

- ◆ Do not change the names of the `theme.xml`, `theme.css`, `print.css` and `dojo.css` files.
- ◆ The CSS Selector names must remain the same, but you can change the properties of the selectors to establish the look and feel.
- ◆ The images subdirectory can have any name, but you must reference it correctly in the CSS and XML files.

5 Make your changes to the images, CSS style sheets and other theme elements as needed. The following changes are recommended:

- ◆ In the `theme.xml` file:
 - ◆ **display-name:** Change this to a value that represents your theme. It displays as the Theme-name in the Themes page of the User Application's **Administration** tab.
 - ◆ **description:** Change this to a value that describes your theme. It displays as the Description in the Themes page of the User Application's **Administration** tab.
 - ◆ Consider whether to localize the **display-name** and **Description** fields.
 - ◆ Remove the following:

```
<resource-bundle>com.novell.afw.portal.artifacts.theme.BlueGloss</resource-bundle>
<resource-group>admin-resgrp</resource-group>
```

- ◆ In the `dojo.css` file, change the `@import` line to the following value:

```
@import url("../../../../../IDMProv/javascript/dijit/themes/idmua/idmua.css");
```

where `IDMProv` is the name of your WAR context.

- ◆ If you wish to change the appearance of some Dojo elements, such as the menu buttons within the profile section on the Work Dashboard, you should take the following steps, instead of performing the steps above:

1. Copy the following from your extracted WAR in this location: `/javascript/dijit/themes/`

```
dijit.css
dijit_rtl.css
idmua (folder)
```

Paste these items into your new theme folder.

2. Change the `@import` line in the `dojo.css` file, as follows:

```
@import url("idmua/idmua.css");
```

- ◆ In the graphics directory:
 - ◆ **thumbnails.gif:** Replace the copy with your own image. This image displays along with the Theme-name and Description of the theme (described above) that is shown in the Themes page of the **Administration** tab. It typically illustrates what the User Application landing page looks like when the associated theme is applied
 - ◆ **Renaming graphics files:** If you change the names of graphics files (rather than just substituting a different image of the same name), make sure to change the reference to the image in both the `theme.xml` and the `theme.css` file. If the image is not used in the branding interface (for example, if it is not listed as one of the subset of brandable images in the `theme.xml` file), then you will only need to change the reference to the

image in the `theme.css` file. Suppose you want to rename `images/header_left.gif` to `images/my_company_name.gif`. Edit the `theme.css` file to reflect the new image name.

- 6 After you make all of the desired changes to the theme files, add your customized theme directory to a new WAR file that contains one or more custom themes. Deploy the new WAR to your test application server.

Testing tip: Open the Themes page (available under the **Administration** tab). Your theme should display along with the prepackaged themes. Use the Theme Preview action to see how the customized changes to your new theme will render. This is a useful way to preview many of your intended changes to your theme. Running through commonly used features of the application is also a recommended testing step.

- 7 After your changes are fully tested, you can deploy the WAR containing the custom theme to your production application server.

Any number of custom themes can reside in a single WAR. Any number of custom WARs containing custom themes can be deployed.

If you want to use a custom theme for the User Application on Tomcat, perform the following steps:

- 1 Stop Tomcat.
- 2 Open the `context.xml` file located in the `%tomcat-install%/conf` directory.
- 3 Modify `<Context>` to `<Context crossContext="true">`.
- 4 Add the custom theme WAR to the `%tomcat-install%/webapps` directory.
- 5 Run the `configupdate` utility to map the context name of the custom theme WAR.
- 6 Start Tomcat.

To undeploy the theme, remove the WAR that contains the theme from the application server's deploy directory. Before undeploying, make sure that any themes it contains are not defined as the User Application's default theme. If you remove the WAR and it does contain the default theme, the Theme Administration screen displays an error message and reverts the User Application theme to the original default theme defined at installation time.

Customizing the Theme for External Password WAR

If you configured Password Management to use an **External Password WAR**, the theme for the Forgot Password page is defined in that external password WAR. The default name for the external password WAR is `IDMPwdMgt.WAR`. The `IDMPwdMgt.WAR` contains one theme; by default, it is **BlueGloss**. It does not include a user interface for modifying or branding this theme.

You can define a custom theme for the external Forgot Password page. The procedure for defining a custom theme is described in ["Defining a Custom Theme" on page 125](#); however, the deployment procedure for the external Forgot Password page is different and the rules about the custom theme WAR are more restrictive. After you define the custom theme:

- ♦ Package the theme in a WAR named `IDMPwdMgtTheme.WAR`.
- ♦ The `IDMPwdMgtTheme.WAR` can contain a single theme, and the theme must be located in the `resource/themes/Theme` directory within the WAR.
- ♦ Deploy the `IDMPwdMgtTheme.WAR` on the application server where the external WAR is located. Only one custom theme can be deployed at a time.

5.1.7 Assigning the User Application Administrator

The User Application Administrator performs administrative tasks for the Identity Manager User Application, using the **Administration** panel of the Identity Manager User Application. The User Application Administrator does not have provisioning administration rights, and is considered an ordinary user while using the **Work Dashboard** panel. There can be more than one User Application Administrator.

One user *must* be assigned to the User Application Administrator role at installation. The User Application Administrator created during installation can administer everything in the User Application including the Provisioning system and can designate other users as User Application Administrators.

You can assign the User Application Administrator at installation and on the Application Configuration page on the Administration tab of the Identity Manager User Application. When you assign the administrator at installation, IDM writes the assignment to the User Application configuration file, which is editable with the configupdate utility. But, at deployment of the WAR, the assignment is written to the User Application database. Thus, after you start the JBoss Application Server the first time after installation, you cannot change the assignment with the configupdate utility--it must be changed from the Application Configuration page.

A user who is to be a User Application Administrator should typically be located under the user root container specified in the User Application's LDAP configuration. This enables the user to log in simply by username (instead of requiring the fully distinguished name each time).

The user who is a User Application Administrator does not need special directory rights because this role controls application-level access.

When assigning User Application Administrators, you can specify users, groups, or containers.

- 1 Go to the **Application Configuration** page.
- 2 Under **Portal Configuration**, select **User App Administrator Assignment**.
- 3 Specify values for the following search settings:

Setting	What to Do
Search for	Select one of the following from the drop-down menu: <ul style="list-style-type: none">◆ Users◆ Groups◆ Containers
Starts with	If you want to: <ul style="list-style-type: none">◆ Find all available objects of your specified type (user), then make this setting blank.◆ Find a subset of those objects, then enter the starting characters of the CN values you want. (Case is not considered. Wildcards are not supported.)

- 4 Click **Go**.
The results of your search appear in the Results list.
- 5 Select the users, group, or container you want to assign as User Application Administrators, then click **Add (>)**.
Hold down the Ctrl key to make multiple selections.
- 6 Click **Save**.

To unassign User Application Administrators:

- 1 In the Current Assignments list, select the users, group, or container you want to unassign as User Application Administrators, then click **Remove** (<).
Hold down the Control key to make multiple selections.
- 2 Click **Save**.

You cannot delete yourself as User Application Administrator. This is a safeguard to ensure that the User Application always has at least one User Application Administrator.

5.2 Working with the Import and Export Tools

You can use the Tools page to export or import portal content (pages and portlets) used in the Identity Manager User Application. This content is also known as the *portal configuration state* and it includes:

- ♦ Container and shared pages (including each page's assigned portlets, and each portlet's preferences and settings)
- ♦ Portlet registrations

Table 5-5 Portal Data Export and Import Tools

Tool	How it works
Portal Data Export	Generates XML descriptions of a set of selected container and shared pages, and portlets. The XML files are stored in a portal data export ZIP file that can be used as input to the Portal Data Import tool.
Portal Data Import	Accepts a portal data export ZIP file as input. Uses the portal data export ZIP file to generate container and shared pages, and portlets in a portal (User Application).

The Export and Import tools enable you to move the portal configuration state from one portal (User Application) to another, as needed. [Table 5-5 on page 129](#) describes how these tools work.

You can use the Portal Data Export and Import tools to:

- ♦ Move your portal configuration state from a test (source) environment to a production (target) environment
- ♦ Update the configuration state of a portal incrementally
- ♦ Clone a portal
- ♦ Optionally, overwrite the configuration state on the target portal

5.2.1 Requirements

To use the Portal Data Export and Import tools, make sure that the Identity Manager User Application (portal) is deployed and running on your source and target application servers.

It is not required that your source and target servers access the same Identity Vault; they can access different ones, if appropriate. The users, groups, and containers in those Identity Vaults are not required to be the same.

5.2.2 Restrictions

You cannot use the Portal Data Export and Import tools to:

- ♦ Export or import portal configuration state when a server is currently servicing user requests
- ♦ Export or import portal classes and resources
- ♦ Export or import portlet classes and resources
- ♦ Export or import the identity and provisioning data used in a portal
- ♦ Export or import administration settings other than for pages and portlets
- ♦ Migrate configuration state from an earlier portal version to a later version (the portals must be the same version)

5.2.3 Exporting Portal Data

This section describes how to export a portal's configuration state to a portal data export ZIP file.

- 1 If you are performing an incremental update, back up the target portal.
- 2 On the Application Configuration page, select **Portal Data Export** from the navigation menu on the left.

The Portal Data Export panel displays.

- 3 Follow the on-screen instructions to select the portal pages and portlets that you want to export.

Some portlets that you have not selected for export might still be exported. If you export a page that contains a portlet, but do not select that portlet for export, the portlet is still exported (to ensure that a runtime error does not occur for the exported page).

- 4 When you are done making selections, click **Export Portal Data**.

Your new portal data export ZIP file is generated, with a default name that includes the current date and time. For example:

```
PortalData.21-Oct-05.09.12.16.zip
```

You are then prompted to save this ZIP file locally (or to open it in an appropriate archive utility). For example:

- 5 Save the portal data export ZIP file to an appropriate location.

5.2.4 Importing Portal Data

This section describes how to import a portal data export ZIP file to a portal.

NOTE: Remember that, during the import, your target application server must be running but not currently servicing user requests.

- 1 If you are performing an incremental update, back up the target portal.
- 2 On the Tools page, select **Portal Data Import** from the navigation menu on the left.

The Portal Data Import panel displays

- 3 Specify the following general import settings:

Setting	What to Do
Archive	Click Browse to select the portal data export ZIP file to import. For example: PortalData.21-Oct-05.09.12.16.zip
Import security settings?	Select one of the following: <ul style="list-style-type: none"> ◆ Yes: If you want to import the permissions that the portal data export ZIP file specifies for access to pages and portlets by users, groups, and containers. Make sure that the users, groups, and containers involved exist in the target portal's Identity Vault; permissions for missing entities fail to be imported. ◆ No: If you want to ignore the permissions that the portal data export ZIP file specifies.

4 Click **View Import Archive**.

The panel displays more specifics about your selected portal data export ZIP file and how you want to import it:

5 Specify the following detailed import settings:

Setting	What to Do
Replace existing data?	Select one of the following: <ul style="list-style-type: none"> ◆ Yes: If you want the contents of the portal data export ZIP file to overwrite corresponding pages and portlets that already exist in the target portal. For example, if the portal data export ZIP file contains a shared page named MyPage and the target portal contains a shared page named MyPage, that existing page is overwritten in the target portal. ◆ No: If you want to skip the import for all existing pages and portlets.
Access level for imported objects	Select one of the following: <ul style="list-style-type: none"> ◆ All Users: For unrestricted access to imported pages and portlets. ◆ Administrator only: For restricted access to imported pages and portlets. <p>If you chose to import security settings, then this access level is applied only to those imported pages and portlets where a security setting failed to be imported, typically because specified users, groups, or containers do not exist in the target portal's Identity Vault.</p> <p>If you chose not to import security settings, then this access level is applied to all pages and portlets that are imported.</p>

Setting	What to Do
Import group settings?	<p>(If you chose to import security settings) Select one of the following:</p> <ul style="list-style-type: none"> ◆ Yes: If you want to import the default container page and default shared page assignments that the portal data export ZIP file specifies for groups. Make sure that the groups involved exist in the target portal's Identity Vault; assignments for missing groups fail to be imported. ◆ No: If you want to ignore the default page assignments that the portal data export ZIP file specifies for groups.
Import Container Pages	Follow the on-screen instructions to select the pages and portlets that you want to import from the portal data export ZIP file to the target portal.
Import Shared Pages	
Import Portlets	
Please map the portlet application names... Archive/ Local	NOTE: Some portlets that you have not selected for import might still be imported. If you import a page that contains a portlet, but do not select that portlet for import, the portlet is still imported to ensure that a runtime error does not occur for the imported page.
	Use the Archive and Local drop-down menus to map the portlet application names in the archive (portal data export ZIP file) to existing portlet applications on the local (target) application server.

6 When you're ready to begin the import, click **Import Portal Data**.

When the import completes, the Portal Data Import Results panel displays.

Unsuccessful imports display in red. To troubleshoot import or export problems, look at your application server's system console or log file (such as `jboss/server/IDMProv/log/catalana.out`) for messages from the following User Application log:

```
com.novell.afw.portal.util
```

7 Test the target portal to ensure that you imported the data that you expected.

5.3 Password Management Configuration

This section describes how to configure password self-service and user authentication features to your Identity Manager User Application.

5.3.1 About Password Management Features

The password management features supported by an Identity Manager User Application encompass user authentication and password self-service. When you put these features into use, they enable your application to:

- ◆ Prompt for *login* information (username and password) to authenticate against eDirectory
- ◆ Provide users with password change self-service
- ◆ Provide users with forgotten password self-service (including prompting for challenge responses, displaying a password hint, or allowing a password change, as needed). You can configure forgotten password self-service to run inside the firewall (the default), or you can configure it to run outside the firewall.

- ◆ Provide users with challenge question self-service
- ◆ Provide users with password hint self-service

Required Setup in eDirectory

Before you can use most of the password self-service and user authentication features, you need to do the following in eDirectory:

- ◆ Enable **Universal Password**
- ◆ Create one or more password policies
- ◆ Assign the appropriate password policies to users

A password policy is a collection of administrator-defined rules that specify the criteria for creating and replacing user passwords. NetIQ Identity Manager takes advantage of NMAS (NetIQ Modular Authentication Service) to enforce password policies that you assign to users in eDirectory.

You can use NetIQ iManager to perform the required setup steps. For example, here's how someone defined the DocumentationPassword Policy in iManager.

Figure 5-1 Sample Password Policy

Password Policy Summary		
Name	Documentation	
Description	<input type="text"/>	
Universal Password		
Options	Enable Universal Password	true
	Enable the Advanced Password Rules	true
	Remove the NDS password when setting Universal Password	false
	Synchronize NDS password when setting Universal Password	true
	Synchronize Simple Password when setting Universal Password	false
	Synchronize Distribution Password when setting Universal Password	true
	Allow user agent to retrieve password	true
	Verify whether existing passwords comply with the	false

OK Cancel Apply

This password policy specifies:

- ◆ Universal Password settings
- ◆ Settings to deal with forgotten-password situations
- ◆ Assignments that apply the policy to specific users

Case-Sensitive Passwords

By default passwords are not case-sensitive. You can create a password policy that allows case-sensitive passwords. You can specify the **Allow the password to be case-sensitive** in the **Password Policies > Universal Password > Advanced Password Rules**. If you enable case-sensitive password, you must also enable the **Allow user to retrieve password** setting. It is enabled by default, but you can verify it through the iManager **Password Policies > Universal Password > Configuration Options** tab.

Password Policy Compliance

If you enable Universal Password, it is recommended that you also configure the system to verify that existing passwords comply with the password policy. You can configure this through iManager. In iManager, go to **Passwords > Password Policies > Universal Password > Configuration Options**. Make sure the following option is selected: **Verify whether existing passwords comply with password policy (verification occurs on login)**. This ensures that users created through the User Application are forwarded to the Change Password page to enter a password that complies with the Identity Manager password policy.

5.3.2 Configuring Challenge Response

The Challenge Response self-service page lets users:

- ◆ Set up the valid responses to administrator-defined challenge questions, and set up user-defined challenge questions and responses
- ◆ Change the valid responses to administrator-defined challenge questions, and change user-defined challenge questions and responses

NOTE: The password management facility makes passwords case-sensitive, by default, and also allows you to configure case sensitivity for passwords. This is not the case with the Challenge Response facility. Challenge Response answers are not case sensitive, and cannot be configured to support case sensitivity.

TIP: If you have localized the Challenge Response questions in iManager set the **Login Configuration** setting **Enable Locale Check** to True.

Requirements

The Challenge Response requirements are described [Table 5-6 on page 135](#).

Table 5-6 Challenge Response Requirements

Topic	Requirements
Password policy	A password policy with forgotten password enabled and a challenge set.
Universal Password	Does not require Universal Password to be enabled.
eDirectory configuration	Requires that you grant supervisor rights to the LDAP Administrator for the container in which the logged-in user resides. Granting these privileges allows the user to write a challenge response to the secret store. For example, suppose the LDAP realm administrator is cn=admin, ou=sample, n=netiq and you log in as cn=user1, ou=testou, o=netiq. You need to assign cn=admin, ou=sample, n=netiq as a trustee of testou, and grant supervisor rights on [All attribute rights] .

Using the Challenge Response Feature

To use the Challenge Response feature, you need to know about the following:

- ♦ [“How Challenge Response Is Used During Login” on page 135](#)
- ♦ [“How Challenge Response Is Used in the User Application” on page 135](#)

How Challenge Response Is Used During Login

During the login process, the Login page automatically redirects to Challenge Response whenever the user needs to set up challenge questions and responses (for example, the first time a user attempts to log in to the application after an administrator assigns the user to a password policy in iManager. The password policy must have forgotten password enabled and include a challenge set).

How Challenge Response Is Used in the User Application

By default, the User Application provides users with self-service for changing challenge questions and responses.

Configuring Challenge Response

The Challenge Response Configuration settings (on the **Administration** tab) are described in the following table.

Setting	Description
Mask Response Text	Choosing Yes means that user-entered response text is masked with asterisk (*) characters.

5.3.3 Configuring Login

The Login page performs a very robust user authentication supported by Identity Manager (through Universal Password, password policies, and NMAS). The Login page redirects to the other password pages as needed during the login process.

Requirements

The Login page requirements are listed in [Table 5-7](#) below.

Table 5-7 Login Requirements

Topic	Requirements
Password policy	This page does not require a password policy, unless you want to use advanced password rules or let users click the Forgot Password link.
Universal Password	This page does not require Universal Password to be enabled, unless you want to use a password policy with advanced password rules.
SSL	This page uses SSL, so make sure that your application server is properly configured to support SSL connections to your LDAP realm.

Use the **Password Module Setup Login Action** to configure the following settings:

Table 5-8 Login Configuration Settings

Configuration Setting	Description
Allow ID Wildcard	If True, users can specify the first few characters of a username and a list of usernames that include those characters is displayed so the use can select the user to login as.
Enable Forgot Password Link	If True, the User Application Login page displays the Forgot Password link.
Enable Hint Migration	If True, any existing hints are moved from the nsimHint to the nsimPasswordReminder.
Enable Locale Check	If True, and the user has not set their locale preferences, the User Application displays a page that allows them to set their preferred locale.
Enable Password Autocomplete	If True and supported by the browser, the user's browser opens a window asking if the user wants to save the login credentials. If False (the default), the user does not receive a browser prompt to save the login credentials.
Guest Container Page	Allows you to specify a custom guest container page. For example, you might specify any of the following values to direct the user to the MyOrgChart page: <code>/IDMProv/portal/cn/DefaultContainerPage/MyOrgChart</code> <code>/portal/cn/DefaultContainerPage/MyOrgChart</code> <code>http://localhost:9000/IDMProv/portal/cn/DefaultContainerPage/MyOrgChart</code> The default value is: <code>GuestContainerPage</code>
Logout URL	This value specifies the URL that a user is redirected to after the user presses the Logout button in the User Application.

Configuration Setting	Description
Password Change Return Page	<p>This value specifies the URL that a user is redirected to after a password change. If you specify an URL for this setting, the User Application displays a link to the redirect page, along with a success message when the password has been changed.</p> <p>This setting only works when accessing the User Application via NetIQ Access Manager. If you access the User Application without going through Access Manager, the Password Change Return Page link will not display.</p> <p>Furthermore, this setting only works within the context of the User Application and not when you access the ChangePassword.jsp directly. If you access the ChangePassword.jsp directly, you will not see a link displayed that redirects to the Password Change Return Page.</p> <p>Stand-alone access to the ChangePassword.jsp When accessing ChangePassword.jsp directly, if you want users to receive a success message, you need to add the following URL parameter:</p> <pre>?changePasswordForcedLogout=true</pre> <p>For example:</p> <pre>http://myserver/IDMProv/jsps/pwdmgt/ChangePassword.jsp?changePasswordForcedLogout=true</pre> <p>Otherwise, the user will not receive a success message after changing their password.</p> <p>Using NetIQ Access Manager's Expired Password Servlet If you are using NetIQ Access Manager and want to utilize Password Expiration, then the URL for the Password Expiration Servlet within NetIQ Access Manager will need to be similar to the following:</p> <pre>http(s)://%server%:%port%/%context%/jsps/pwdmgt/ChangePassword.jsp?changePasswordForcedLogout=true&idp_return_url=<RETURN_URL>&store=<STOREID>&dn=<USERID>&action=expire</pre> <p>For example:</p> <pre>http://myserver.netiq.com/IDMProv/jsps/pwdmgt/ChangePassword.jsp?changePasswordForcedLogout=true&idp_return_url=<RETURN_URL>&store=<STOREID>&dn=<USERID>&action=expire</pre>
Enable Password Expiration Warning	<p>This setting gives you the ability to enable or disable the expired password warning. This feature is useful in configurations where another product has detected an expired password and already warned the user prior to redirecting to the Identity Manager portlets.</p>
Using SSL Login	<p>This setting gives you the ability to configure the Login to redirect to https. If you set Using SSL Login to true, then when user goes to the login.jsp (either directly or through a redirect from NONE SSL page), the login.jsp page will be presented with https with the SSL port configured (Server SSL Port). After user logs in, he see the https (SSL) Home page.</p>
Server SSL Port	<p>Specifies the SSL port that the User Application is running on.</p>

Using the Login Page

To use the Login page, you need to know about the following:

- ♦ “How Login Redirects to Other Pages” on page 138
- ♦ “Using Grace Logins” on page 138

How Login Redirects to Other Pages

At runtime, the Login page redirects to other password pages, depending on what’s needed to complete the login process. [Table 5-9 on page 138](#) directs you to descriptions.

Table 5-9 Login Directions to Other Pages

If the user	Login redirects to
Clicks the link Forgot Password	Forgot Password page
Needs to set up challenge questions and responses	Challenge response page
Needs to set up a password hint	Hint Definition page
Needs to reset an invalid password	Change password page

Using Grace Logins

If you use a grace login, the Login page displays a warning message that asks you to change your password and indicates the number of grace logins that remain. If you are on your last login, the Login page redirects you to the Change Password page.

5.3.4 Configuring Password Sync Status

Password Sync Status lets users check the progress of the password change process on connected systems. You can specify a different image to represent each connected system. To set up password sync status checking:

- ♦ Define the connected applications whose status the user should be able to view during the synchronization process. You define the connected applications in the Password Sync Status Application Settings described in [Table 5-11 on page 140](#).
- ♦ Define the settings for the password sync status page displayed to users. These settings are described in [Table 5-10, “Password Sync Status Client Settings,” on page 139](#).

By default, the User Application Administrator can view the password sync status of other users when the User Application Administrator accesses the Password Sync Status page. The administrator can access the sync status for another user by specifying the other user’s DN, then clicking **Check Sync Status**.

In addition to the User Application Administrator, you can define a set of users to perform the Check Sync Status for other users (for troubleshooting or other purposes). The members of a group called PasswordManagement are also automatically allowed to view the password synchronization status of other users. This group does not exist by default. If you choose to create this group, it must be:

- ◆ Named PasswordManagement.
- ◆ Given privileges to the Identity Vault. The group must have rights to read the user's eDirectory object attribute for users whose password synchronization status they need to view. The system accesses the DirXML-passwordSyncstatus, the pwdChangedTime, and the DirXML-Associations attributes.

Table 5-10 Password Sync Status Client Settings

Configuration Setting	Description
Password Sync Buffer Time (milliseconds)	<p>The password sync status checking compares time stamps across different Identity Vaults and connected systems. This buffer time is intended to account for differences between the system times on these different machines. This time is added to the time stamp on the user object's password change attribute to determine if a change has occurred. It is used like this:</p> <p>The Password Sync Status process uses the buffer time as follows:</p> <ul style="list-style-type: none"> ◆ If the time stamp value (password sync time) in DirXML-PasswordSyncStatus for the connected system is older than the last password change time stamp (pwdChangedTime attribute of user object) + password sync buffer time, then the status is considered old and the system continues polling for an updated status for the connected system. ◆ If the time stamp value in DirXML-PasswordSyncStatus for the connected system is newer than the last password change time stamp + password sync buffer time, then the password sync functionality returns the status code or message and displays the updated status of the connected system. ◆ The last password change time stamp is populated to the user object after the user's password change. This functionality is available in NMAS 3.1.3 and higher.
Image Per Row	The number of application images to display per row in the Identity Self-Service Password Sync Status page.
Individual Application Timeout (milliseconds)	The amount of time that the Password Sync Status process waits for a response for each connected application's status before checking for the next one.
All Application Timeout (milliseconds)	This value indicates the amount of time allowed for the entire password sync status process (of all connected systems) to complete. Before this timeout is reached, the password sync process continues to poll until all status values are updated or this timeout is reached. When the timeout status is reached, the system displays an error message to the user that indicates that a timeout condition has been reached.

Configuration Setting	Description
Process Count	The number of times each connected system is checked for the password sync status.
Pass Phrase	If the DirXML-PasswordSyncStatus contains a password hash, then the value entered in this field is compared to that value. If they are not equal, the User Application displays an invalid hash message.
Application Image Size Limit (bytes)	Lets you set the maximum size (in bytes) of the application image that can be uploaded. You specify this image in the Application Image setting described in Table 5-11 .
Show Password Sync Status After Password Change	If this field is set to true, after the user changes a password, the interface presents the Password Sync Status screen. If this field is set to false, the Password Sync Status screen is not displayed after a password change.

The password Sync Status Application Settings are described in [Table 5-11](#).

Table 5-11 Password Sync Status Application Settings

Configuration Setting	Description
Password Synchronization Application Name	<p>The name used to describe the connected application. You can enter the application name in multiple locales.</p> <p>To add a language (locale):</p> <ol style="list-style-type: none"> 1. Click Add Language (+). 2. Type the Application Name for the desired localized languages in the appropriate field. 3. Click Save. <p>If you do not specify localized application names, the value specified in the Password Synchronization Application Name is used.</p>
Application DirXML-PasswordSyncStatus GUID	<p>You can get the driver GUID by browsing the attributes on the driver object in one of two ways:</p> <ul style="list-style-type: none"> ◆ Click the browse button next to this field. This browse button obtains only GUIDs of drivers in the current driverset that the User Application driver resides in. ◆ Use iManager to browse for the driver (use the General - Other tab, used when modifying the object) and manually copy and paste the GUID into this field.
Application Image	<p>The name of the connected application Image to upload. The Application Image size can be configured from the Application Image Size Limit field in the Password Sync Status Client Settings section. Supported file types are .bmp, .jpeg, .jpg, .gif, and .png.</p>

Configuration Setting	Description
Application Filter	<p>Optional. Specify an LDAP filter that allows or prohibits users' viewing the application name on their Check Password Synchronization pages.</p> <p>You can use any standard LDAP filter.</p>
Dependent Driver	<p>Optional. Specify any additional driver this application depends on.</p> <p>If any driver in the dependent driver chain is not visible to the user, the driver specified by Application DirXML-PasswordSyncStatus GUID is also not visible to the user.</p> <p>If any driver in the dependent driver chain fails to check password sync status, the driver specified by Application DirXML-PasswordSyncStatus GUID also fails to check password sync status.</p> <p>You can get the driver GUID by browsing the attributes on the driver object in one of two ways:</p> <ul style="list-style-type: none"> ◆ Use the object selector button beside the Dependent Driver field. <p>This method saves the application driver's fully distinguished name (FDN). When a user checks password sync status, this FDN is compared to the value of the FDN field in the DirXML-Associations attribute of the user object. If the two FDNs do not match, this application is not visible to the user. If there is a match, and if the DirXML-Associations attribute's driver status field is not 0 and the driver data field is not null, this application is visible to the user.</p> ◆ Manually enter the GUID for the dependent driver. <p>Use this method when this application driver is not from the current driverset that the User Application driver resides in. This method does not save an FDN. When a user checks password sync status, FDNs are not compared, and this dependent driver is visible to the user unless you apply an Application Filter that excludes the user.</p>

5.3.5 Configuring Password Hint Change

This self-service page lets users set up or change their password hints, which can be displayed or e-mailed as a clue in forgotten password situations.

Requirements

The Password Hint Change requirements are listed in [Table 5-12](#).

Table 5-12 Password Hint Change Requirements

Topic	Requirements
Universal Password	Does not require Universal Password to be enabled.

Using the Password Hint Change Page

To use the Password Hint Change page, you need to know about the following:

- ♦ “How Password Hint Change Is Used During Login” on page 142
- ♦ “Using Password Hint Change in the User Application” on page 142

How Password Hint Change Is Used During Login

During the login process, the Login page automatically redirects to the Password Hint Change page whenever users need to set up their password hints. For example, the first time a user attempts to log in to the application after an administrator assigns the user to a password policy in iManager, the password policy has forgotten password enabled and has the action set to **Email hint to user** or **Show hint on page**.

Using Password Hint Change in the User Application

By default, the User Application provides users with self-service for changing a password hint.

5.3.6 Configuring Change Password

This self-service page lets users change (reset) their Universal Passwords, according to the assigned password policy. It uses that policy to display the rules that the new password must conform to.

If Universal Password is not enabled, this page changes the user’s eDirectory (simple) password, as permitted in the user’s Password Restrictions.

There are no Password Change configuration settings.

Requirements

The Change Password page requirements are listed in [Table 5-13](#).

Table 5-13 Change Password Requirements

Topic	Requirements
Directory Abstraction Layer configuration	No directory abstraction layer configuration is required for this page.
Password policy	This page does not require a password policy, unless you want to use advanced password rules (with Universal Password enabled).

Topic	Requirements
Universal Password	<p>To use this page for a Universal Password, the setting Allow user to initiate password change must be enabled in the Advanced Password Rules of the user's assigned password policy.</p> <p>To use this page for an eDirectory (simple) password, the setting Allow user to change password must be enabled in the user's Password Restrictions.</p>

Using the Change Password Page

To use the Change Password page, you need to know about the following:

- ♦ [“How Change Password Is Used During Login” on page 143](#)
- ♦ [“Using Change Password in the User Application” on page 143](#)

How Change Password Is Used During Login

During the login process, the Login page automatically redirects to the Change Password page whenever the user needs to reset an invalid password. For example, the first time a user attempts to log in to an application after an administrator implements a password policy that requires users to reset their passwords.

The Forgot Password page also redirects to Change Password automatically if the user's assigned password policy specifies reset password as the action for forgotten password situations.

Using Change Password in the User Application

By default, the User Application provides users with the password change self-service using the Change Password page.

NOTE: On Firefox, if you allow the browser to save passwords, you may see a confusing pop-up message that asks the following question when you confirm a password change: “Would you like to have password manager change the stored password for <user>?”. The user specified in the message may not be the same as the user who logged into the User Application. This message is generated by the Firefox password manager. To turn off this message, you need to disable the password manager in Firefox by deselecting the **Remember passwords sites** checkbox under **Passwords** on the **Tools>Options>Security** page.

6 Page Administration

This section describes how to use the Page Admin page on the **Administration** of the Identity Manager user interface.

6.1 About Page Administration

You use the Page Admin page to control the pages displayed in the Identity Manager User Application and who has permission to access them. The user interface includes two types of pages.

Table 6-1 Page Types

Type of Page	Description
Container	Container pages wrap shared pages with a consistent look and feel, corporate branding, and navigation approach.
Shared	Shared pages provide a coherent set of content that is used for a specific purpose (such as updating a user's profile). They are called shared pages because they offer services used by multiple people.

6.1.1 About Container Pages

This section introduces you to some container pages that play an important role in the Identity Manager user interface:

- ♦ [“GuestContainerPage” on page 145](#)
- ♦ [“DefaultContainerPage” on page 146](#)
- ♦ [“Admin Container Page” on page 146](#)

Keep in mind that you can modify these container pages if necessary. You also have the option of adding your own container pages.

To learn about working with container pages, see [Section 6.2, “Creating and Maintaining Container Pages,” on page 147](#).

GuestContainerPage

By default, when users arrive at the Identity Manager user interface prior to logging in, they see the container page named GuestContainerPage.

The GuestContainerPage layout is divided into three regions, which display the following portlets:

Table 6-2 Layout Regions

Portlet	Description
HeaderPortlet	Displays the header information and top-level controls for the user interface
Shared Page Navigation	Displays a vertical menu from which the user can select a shared page to display
Portal Page Controller	Displays the shared page that the user has currently selected via the Shared Page Navigation portlet

DefaultContainerPage

By default, after users log in to the Identity Manager user interface, they go to the container page named DefaultContainerPage.

The DefaultContainerPage layout is divided into three regions, which display the portlets described in [Table 6-3](#).

Table 6-3 Default Container Page Portlets

Portlet	Description
HeaderPortlet	Displays the header information and top-level controls for the user interface
Shared Page Navigation	Displays a vertical menu from which the user can select a shared page to display
Portal Page Controller	Displays the shared page that the user has currently selected via the Shared Page Navigation portlet
Session Timeout Warning	Displays an alert message whenever a user's session is about to time out

After user login, DefaultContainerPage automatically opens the **Identity Self-Service** in HeaderPortlet.

Admin Container Page

By default, when User Application Administrators (and other authorized users) click the **Administration** tab of the Identity Manager user interface, they go to the container page named Admin Container Page.

The Admin Container Page layout is divided into two regions, which display the portlets described in [Table 6-4](#).

Table 6-4 Default Admin Container Page Portlets

Portlet	Description
HeaderPortlet	Displays the header information and top-level controls for the user interface

Portlet	Description
Admin List Display	Displays a second level of tabs from which the user can select an administration action to perform
Portal Page Controller	Displays a shared page that corresponds to the currently selected by the user via the Admin List Display portlet
Session Timeout Warning	Displays an alert message whenever a user's session is about to time out

6.1.2 About Shared Pages

The Identity Manager user interface includes many shared pages, which provide the major content within its container pages. You can modify these shared pages if necessary. You also have the option of adding your own shared pages.

To learn about working with shared pages, see [Section 6.3, “Creating and Maintaining Shared Pages,” on page 152.](#)

A Typical Shared Page

As an example of one of these shared pages, Organization Chart is the default shared page that DefaultContainerPage displays after users log in to the Identity Manager user interface.

The Organization Chart layout consists of just one region, which displays just one portlet (the Org Chart portlet).

6.1.3 An Exception to Page Usage

In this section, you have seen how these top-level tabs of the Identity Manager user interface are based on pages:

- ◆ The **Identity Self-Service** uses the DefaultContainerPage
- ◆ The **Administration** uses the Admin Container Page

However, the **Work Dashboard** is based on a different architecture and cannot be manipulated through Page Admin.

6.2 Creating and Maintaining Container Pages

The process of creating and maintaining container pages involves the following steps:

- 1 Create a new container page or select an existing container page, as described in [Section 6.2.1, “Creating Container Pages,” on page 148.](#)
- 2 Add content (in the form of portlets) to the page, as described in [Section 6.2.2, “Adding Content to a Container Page,” on page 150.](#)
You can also delete content from the page, as described in [Section 6.2.3, “Deleting Content from a Container Page,” on page 150.](#)
- 3 Choose a portal layout, as described in [Section 6.2.4, “Modifying the Layout of a Container Page,” on page 151.](#)

- 4 Arrange the order and position of content on the selected layout, as described in [Section 6.2.5, “Arranging Content on the Container Page,”](#) on page 151.
- 5 Immediately display the new page by specifying the container page URL in your browser, as described in [Section 6.2.6, “Displaying a Container Page,”](#) on page 152.

You can switch layouts for container pages without losing page contents. When you apply a new layout to a container page, portlets in the page are automatically displayed using the new layout. You might need to fine-tune the content placement in the new layout.

6.2.1 Creating Container Pages

You can create container pages from scratch or by copying existing pages. This section describes both procedures.

To create a container page from scratch:

- 1 On the Page Admin page, select **Maintain Container Pages**.
The Maintain Container Pages panel displays.
- 2 Select the **New** page action (in the bottom left section of the panel).
An untitled, uncategorized container page is created.
- 3 Specify the page properties of the container page:

Property	What to do
Page Link Name (URI)	<p>Specify the URI name for the page (as it is to appear within the user interface URL). For example, if you specify the URI:</p> <p><code>MyContainerPage</code></p> <p>it appears within the URL like this:</p> <p><code>http://myappserver:8080/IDMProv/portal/cn/MyContainerPage</code></p> <p>NOTE: The User Application does not support multibyte characters in the Page Link Name (URI) for a portal page. Multibyte characters are supported in the Page Name.</p>
Page Name	<p>Specify the display name for the page. For example:</p> <p><code>My Container Page</code></p> <p>Click Localize to specify localized versions of this name for other languages.</p>
Navigation Priority	<p>Specify one of the following:</p> <ul style="list-style-type: none"> ◆ None if you don't need to assign a priority to this container page. ◆ Set value to assign a priority to this container page, relative to other container pages. The priority must be an integer between 0 and 9999, where 0 is the lowest priority and 9999 is the highest. <p>Setting priority values is useful if you want to ensure a particular order when pages are listed by priority, or if you want to ensure a particular selection when multiple default pages exist (in the case of a user who belongs to multiple groups).</p>
Default Shared Page	<p>See Section 6.5, "Selecting a Default Shared Page for a Container Page," on page 158.</p>
Assign Categories	<p>Select zero or more of the following categories in which you want the page to belong:</p> <ul style="list-style-type: none"> ◆ Administration ◆ General <p>Assigning categories is useful if you want to ensure proper organization when pages are listed by category, or if you want to ensure an appropriate subset when pages are filtered by category.</p> <p>You Cannot Create New Administration Pages The administrator cannot create new Administration pages. If you attempt to create a new page in the Administration category, the page will not be displayed under Application Configuration.</p>
Description	<p>Type text that describes the page.</p>

4 Click **Save Page** (at the bottom of the page properties section).

To create a container page by copying an existing page:

- 1 On the Page Admin page, select **Maintain Container Pages**.
The Maintain Container Pages panel displays (as shown in the previous procedure).
- 2 In the list of container pages, select the page you want to copy.
If the list is long, you can refine it (by category or starting text) to more easily find the desired page.
- 3 Select the **Copy** page action (in the bottom left section of the panel).
A new container page is created with the name `Copy of OriginalPageName`.
- 4 Specify the page properties of the container page (as described in the previous procedure).
- 5 Click **Save Page** (at the bottom of the page properties section).

6.2.2 Adding Content to a Container Page

After you create a container page, the next step is to add content by selecting portlets to place on the page. You can use prebuilt portlets supplied with the Identity Manager User Application or other portlets you have registered.

To add content to a container page:

- 1 Open a new or existing page on the Maintain Container Pages panel, then click the **Select Content** page task (at the bottom of the panel).
The Content Selector displays in a new browser window.
- 2 If you want to display a specific category of available content, select a category from the **Filter** list.
- 3 Select one or more portlets from the **Available Content** list.
Hold down Control to select multiple non-contiguous portlets from the list; use Shift to make multiple contiguous selections.
- 4 Click **Add** to move your choices to the **Selected Content** list.
- 5 You can click **Content Preferences** to edit the preferences of any portlet you have selected for your container page. The preference values you specify take effect for the instance of the portlet that appears on your page.
- 6 Click **Save Contents**.

Now that you have chosen the content for your container page, you can select a new layout as described in [Section 6.2.4, "Modifying the Layout of a Container Page," on page 151](#), or arrange the content on the current layout as described in [Section 6.2.5, "Arranging Content on the Container Page," on page 151](#).

6.2.3 Deleting Content from a Container Page

In the process of creating container pages, you might want to delete content by removing portlets from a page. You can use the Content Selector or Layout Selector, as described in the following procedures.

To delete content from a container page using the Content Selector:

- 1 Open a page on the Maintain Container Pages panel, then click the **Select Content** page task (at the bottom of the panel).
The Content Selector displays in a new browser window as shown in [Step 1 on page 150](#).

- 2 Select a portlet you want to delete from the **Selected Content** list and click **Remove**.
The portlet is removed from the page.
- 3 Click **Save Contents**.

To delete content from a container page using the Layout Selector:

- 1 Open a page on the Maintain Container Pages panel, then click the **Arrange Content** page task (at the bottom of the panel).
The Layout Selector displays in a new browser window, showing the portlets on that page.
- 2 Click the **X** button for a portlet you want to remove.
- 3 When you're prompted for confirmation, click **OK**.
The portlet is removed from the page.
- 4 Click **Save Layout**.

6.2.4 Modifying the Layout of a Container Page

When you modify the layout of a container page, existing content is shifted to accommodate the new layout. In some cases, you might need to fine-tune the end result.

To modify the layout of a container page:

- 1 Open a page on the Maintain Container Pages panel, then click the **Select Layout** page task (at the bottom of the panel).
The Portal Layouts list displays in a new browser window.
- 2 Scroll through the choices and select the layout you want.
- 3 Click **Select Layout**.

6.2.5 Arranging Content on the Container Page

After you have designated the content and layout for your container page, you can position the content in the selected layout, add other portlets in specific locations, or delete portlets.

- 1 Open a page on the Maintain Container Pages panel, then click the **Arrange Content** page task (at the bottom of the panel).
The Layout Selector displays in a new browser window, showing the portlets on that page.
- 2 To add a portlet to the page:
 - 2a Click **Add Content** in the desired layout frame.
The Portlet Selector displays in a new browser window.
 - 2b If you want to display a specific category of available content, select a category from the **Filter** drop-down list.
 - 2c Select a portlet you want from the **Available Content** list.
 - 2d Click **Select Content**.
The Portlet Selector closes and the portlet you selected appears in the target layout frame of the Layout Selector.
- 3 If you want to move a portlet to a different location in the layout, follow these browser-specific steps:

Browser	What to do
Internet Explorer	<ol style="list-style-type: none"> 1. Move your cursor over the title bar of the portlet until the cursor changes to a hand shape. 2. Hold down the left mouse button and drag the portlet to the desired location in the layout.
Mozilla	<ol style="list-style-type: none"> 1. Click the portlet you want to move. 2. Click inside the destination layout frame. <p>The portlet moves to the destination.</p>

4 If you want to remove a portlet from the layout, follow these steps:

4a Click the **X** button for the portlet you want to remove.

4b When you're prompted for confirmation, click **OK**.

The portlet is removed from the layout.

5 To edit the preferences of a portlet:

5a Click the pencil button for the portlet you want to edit.

The portlet's **Content Preferences** display in your browser.

5b Change preference values, as appropriate.

The preference values you specify take effect for the instance of the portlet that appears on your page.

5c Click **Save Preferences**.

6 Click **Save Layout** to record your changes and close the Layout Selector.

6.2.6 Displaying a Container Page

You can display your page by going to the container page URL in your browser. Specify the following URL in your web browser:

```
http://server:port/IDM-war-context/portal/cn/container-page-name
```

For example, to display the container page named MyContainerPage:

```
http://myappserver:8080/IDMProv/portal/cn/MyContainerPage
```

6.3 Creating and Maintaining Shared Pages

The process of creating and maintaining shared pages involves the following steps:

1 Create a new shared page or select an existing shared page, as described in [Section 6.3.1, "Creating Shared Pages,"](#) on page 153.

2 Add content (in the form of portlets) to the page, as described in [Section 6.3.2, "Adding Content to a Shared Page,"](#) on page 155.

You might also want to delete content from the page, as described in [Section 6.3.3, "Deleting Content from a Shared Page,"](#) on page 156.

3 Choose a portal layout, as described in [Section 6.3.4, "Modifying the Layout of a Shared Page,"](#) on page 156.

- 4 Arrange the order and position of content on the selected layout, as described in [Section 6.3.5, “Arranging Content on the Shared Page,”](#) on page 156.
- 5 Display the new page by entering the shared page URL in your browser, as described in [Section 6.3.6, “Displaying a Shared Page,”](#) on page 157.

Shared Pages and Layouts

Shared pages are not tightly bound to portal layouts. That means you can switch layouts for shared pages without losing any page contents. When a new layout is applied, any portlets that have been added to the page are automatically displayed using the new layout. You might need to fine-tune the content placement in the new layout.

6.3.1 Creating Shared Pages

You can create shared pages from scratch or by copying existing pages. This section describes both procedures.

To create a shared page from scratch:

- 1 On the Page Admin page, select **Maintain Shared Pages**.
- 2 Select the **New** page action (in the bottom left section of the panel).
An untitled, uncategorized shared page is created.
- 3 Specify the page properties of the shared page:

Property	What to do
Page Link Name (URI)	<p>Specify the URI name for the page (as it is to appear within the user interface URL). For example, if you specify the URI:</p> <p><code>MySharedPage</code></p> <p>it appears within the URL like this:</p> <p><code>http://myappserver:8080/IDMProv/portal/cn/MyContainerPage/MySharedPage</code></p> <p>NOTE: The User Application does not support multibyte characters in the Page Link Name (URI) for a portal page. Multi-byte characters are supported in the Page Name.</p>
Page Name	<p>Specify the display name for the page. For example:</p> <p><code>My Shared Page</code></p> <p>You can click Localize to specify localized versions of this name for other languages.</p>
Navigation Priority	<p>Specify one of the following:</p> <ul style="list-style-type: none"> ◆ None if you don't need to assign a priority to this shared page. ◆ Set value to assign a priority to this shared page, relative to other shared pages. The priority must be an integer between 0 and 9999, where 0 is the highest priority and 9999 is the lowest. <p>Setting priority values is useful if you want to ensure a particular order when pages are listed by priority, or if you want to ensure a particular selection when multiple default pages exist (in the case of a user who belongs to multiple groups).</p>
Parent Page	<p>If you want this shared page to be the child of another shared page, click Select Parent. Make sure that both the parent and child pages belong to the same categories (to prevent display problems).</p> <p>At runtime, the end user sees this relationship when using the Shared Page Navigation portlet. When displaying the list of shared pages, it shows children indented under their parents.</p> <p>Child pages do not inherit content, preferences, or settings from their parent pages. Conversely, parent pages do not automatically display the content of child pages along with their own content.</p>

Property	What to do
Assign Categories	<p>Select zero or more of the following categories in which you want the page to belong:</p> <ul style="list-style-type: none"> ◆ Administration ◆ Directory Management ◆ General ◆ Guest Pages ◆ Information Management ◆ Password Management <p>Assigning categories is useful if you want to ensure proper organization when pages are listed by category, or if you want to ensure an appropriate subset when pages are filtered by category.</p> <p>NOTE: Guest Pages is a special category used to identify shared pages that can be displayed prior to user login but not after.</p>
Description	Type text that describes the page.

- 4 Click **Save Page** (at the bottom of the page properties section).

To create a shared page by copying an existing page:

- 1 On the Page Admin page, select **Maintain Shared Pages**.
The Maintain Shared Pages panel displays as shown in [“To create a shared page from scratch:” on page 153](#).
- 2 In the list of shared pages, select the page you want to copy.
If the list is long, you can refine it (by category or starting text) to more easily find the desired page.
- 3 Select the **Copy** page action (in the bottom-left section of the panel).
A new shared page is created with the name Copy of OriginalPageName.
- 4 Specify the page properties of the shared page as described in [“To create a shared page from scratch:” on page 153](#).
- 5 Click **Save Page** (at the bottom of the page properties section).

6.3.2 Adding Content to a Shared Page

After you create a shared page, the next step is to add content by selecting portlets to place on the page. You can use prebuilt portlets supplied with the Identity Manager User Application or other portlets you have registered.

- 1 Open a new or existing page on the Maintain Shared Pages panel, then click the **Select Content** page task (at the bottom of the panel).
- 2 If you want to display a specific category of available content, select a category from the **Filter** drop-down list.
- 3 Select one or more portlets from the **Available Content** list.
Hold down the Ctrl key to select multiple non-contiguous portlets from the list; use the Shift key to make multiple contiguous selections.
- 4 Click **Add** to move your choices to the **Selected Content** list.

- 5 You can click **Content Preferences** to edit the preferences of any portlet you have selected for your shared page. The preference values you specify take effect for the instance of the portlet that appears on your page.
- 6 Click **Save Contents**.

Now that you have chosen the content for your shared page, you can select a new layout as described in [Section 6.3.4, “Modifying the Layout of a Shared Page,” on page 156](#), or arrange the content on the current layout as described in [Section 6.3.5, “Arranging Content on the Shared Page,” on page 156](#).

6.3.3 Deleting Content from a Shared Page

In the process of creating shared pages, you might want to delete content by removing portlets from a page. You can use the Content Selector or Layout Selector, as described in the following procedures.

- 1 Open a page on the Maintain Shared Pages panel, then click the **Select Content** page task (at the bottom of the panel).
The Content Selector displays in a new browser window.
- 2 Select a portlet you want to delete from the **Selected Content** list and click **Remove**.
The portlet is removed from the page.
- 3 Click **Save Contents**.

To delete content from a shared page by using the Layout Selector:

- 1 Open a page on the Maintain Shared Pages panel, then click the **Arrange Content** page task (at the bottom of the panel).
The Layout Selector displays in a new browser window, showing the portlets on that page.
- 2 Click the **X** button for a portlet you want to remove.
- 3 When you're prompted for confirmation, click **OK**.
The portlet is removed from the page.
- 4 Click **Save Layout**.

6.3.4 Modifying the Layout of a Shared Page

When you modify the layout of a shared page, existing content is shifted to accommodate the new layout. In some cases, you might need to fine-tune the end result.

To modify the layout of a shared page:

- 1 Open a page on the Maintain Shared Pages panel, then click the **Select Layout** page task (at the bottom of the panel).
- 2 Scroll through the choices and select the layout you want.
- 3 Click **Select Layout**.

6.3.5 Arranging Content on the Shared Page

After you have designated the content and layout for your shared page, you can position the content in the selected layout, add other portlets in specific locations, or delete portlets.

To arrange content on a shared page:

- 1 Open a page on the Maintain Shared Pages panel, then click the **Arrange Content** page task (at the bottom of the panel).

The Layout Selector displays in a new browser window, showing the portlets on that page.

- 2 If you want to add a portlet to the page:

- 2a Click **Add Content** in the desired layout frame.

The Portlet Selector displays in a new browser window.

- 2b If you want to display a specific category of available content, select a category from the **Filter** drop-down list.

- 2c Select a portlet you want from the **Available Content** list.

- 2d Click **Select Content**.

The Portlet Selector closes and the portlet you selected appears in the target layout frame of the Layout Selector.

- 3 If you want to move a portlet to a different location in the layout, follow these browser-specific steps:

Browser	What to do
Internet Explorer	<ol style="list-style-type: none">1. Move your cursor over the title bar of the portlet until the cursor changes to a hand shape.2. Hold down the left mouse button and drag the portlet to the desired location in the layout.
Mozilla Firefox	<ol style="list-style-type: none">1. Click the portlet you want to move.2. Click inside the destination layout frame.
The portlet moves to the destination.	

- 4 If you want to remove a portlet from the layout:

- 4a Click the **X** button for the portlet you want to remove.

- 4b When you're prompted for confirmation, click **OK**.

The portlet is removed from the layout.

- 5 If you want to edit the preferences of a portlet:

- 5a Click the pencil button for the portlet you want to edit.

The portlet's Content Preferences display in your browser.

- 5b Change preference values, as appropriate.

The preference values you specify take effect for the instance of the portlet that appears on your page.

- 5c Click **Save Preferences**.

- 6 Click **Save Layout** to record your changes and close the Layout Selector.

6.3.6 Displaying a Shared Page

To display your shared page, go to this URL in your Web browser:

`http://server:port/IDM-war-context/portal/pg/shared-page-name`

For example, to display the shared page named *MySharedPage*:

`http://myappserver:8080/IDMProv/portal/pg/MySharedPage`

6.4 Setting Default Pages for Groups

You can assign a default container page and a default shared page for any authorized group of users. These settings affect the container page those users see when they log in and the shared page they see on the container page.

When users belong to multiple groups with default page assignments, Navigation Priority is used in determining which container page and shared page to display.

To assign a default container page or a default shared page to a group:

- 1 Open a page on the Maintain Container Pages panel or the Maintain Shared Pages panel, then click the **Set As Default** page task (at the bottom of the panel).

The Page Defaults dialog box displays in a new browser window.

- 2 Specify values for the following search settings:

Setting	What to do
Search for	Groups is automatically selected.
Starts with	<p>If you want to:</p> <ul style="list-style-type: none">◆ Find all available groups, then make this setting blank.◆ Find a subset of those groups, then enter the starting characters of the CN values you want. (Case is not considered. Wildcards are not supported.) <p>For example, searching for groups that start with <i>s</i> would narrow your search results to something like this:</p> <pre>cn=Sales,ou=groups,o=MyOrg cn=Service,ou=groups,o=MyOrg cn=Shipping,ou=groups,o=MyOrg</pre> <p>Searching for groups that start with <i>se</i> would return:</p> <pre>cn=Service,ou=groups,o=MyOrg</pre>

- 3 Click **Go**.

The results of your search appear in the **Results** list.

- 4 Select the groups for whom this page is to be a default, then click the **Add (>)** button.

Hold down the Ctrl key to make multiple selections.

- 5 Click **Save**, then click **Close**.

6.5 Selecting a Default Shared Page for a Container Page

You can assign a default shared page to each container page you have. The user interface considers this page assignment when determining what to display.

- 1 Open a container page on the Maintain Container Pages panel.
- 2 In the page properties section, look for Default Shared Page and click **Select Default**.

The Choose a Default Shared Page dialog box displays in a new browser window.

- 3 If the shared page list is long, you can refine it by category or starting text to more easily find the desired page.
- 4 Select a shared page to use as the default for the container page or select **None** for no default.
- 5 Click **Save** to accept your selection and close the dialog.
- 6 Click **Save Page** (at the bottom of the page properties section).

7 RBPM Provisioning and Security Configuration

This section describes the tasks that you can perform from the RBPM Provisioning and Security page.

7.1 About RBPM Provisioning and Security Configuration

The Administration tab now provides a new **RBPM Provisioning and Security** tab, which replaces the **Provisioning** and the **Security** tab. This incorporates left navigation options that were previously available on the **Provisioning** and the **Security** tab. In addition, it includes several new left navigation options that give administrators the ability to assign security permissions in accordance with the new consolidated security model.

7.2 Provisioning Configuration

The Provisioning Configuration actions allow you to configure the Delegation and Proxy Service, the Digital Signature Service, the provisioning user interface settings, and the Workflow Engine and clustering.

To access the Provisioning Configuration actions, you need to be a Configuration Administrator.

7.2.1 Configuring Delegation and Proxy Settings

Configuring the Delegation and Proxy Service

To configure the Delegation and Proxy Service:

- 1 Select the **RBPM Provisioning and Security** tab.
- 2 Select **Delegation and Proxy** from the left navigation menu.
The user interface displays the Delegation and Proxy page. To configure the service, you need to make some changes in the Delegation and Proxy Service Settings box.
- 3 Check the **Allow All Requests** option if you want to display the **All** option in the Resource Search Criteria drop-down list for the Team Delegate Assignments action. When the **All** option is available, a delegate assignment can be defined that applies to all resource categories.
- 4 Define the retention period for delegate, proxy, and availability assignments:

Field	Description
Retention time for Delegation assignments	Specifies the number of minutes to retain delegate assignments in the directory after they have expired. The default is 0, which indicates that the assignments will be removed after the expiration time has been reached.
Retention time for Proxy assignments	Specifies the number of minutes to retain proxy assignments in the directory after they have expired. The default is 0, which indicates that the assignments will be removed after the expiration time has been reached.
Retention time for Availability settings	Specifies the number of minutes to retain availability settings in the directory after they have expired. The default is 0, which indicates that the assignments will be removed after the expiration time has been reached.

- 5 Select the email templates you want to use for delegation, proxy, and availability notifications:

Field	Description
Delegation notification template	<p>Specifies the language-independent name for the template to use for delegation email notifications. After the template name has been specified, the notification engine can determine which language-specific template to use at runtime.</p> <p>For details on creating and editing email templates, see Section 10.4, "Working with Email Templates," on page 216.</p>
Proxy notification template	<p>Specifies the language-independent name for the template to use for proxy email notifications. After the template name has been specified, the notification engine can determine which language-specific template to use at runtime.</p> <p>For details on creating and editing email templates, see Section 10.4, "Working with Email Templates," on page 216.</p>
Availability notification template	<p>Specifies the language-independent name for the template to use for availability email notifications. After the template name has been specified, the notification engine can determine which language-specific template to use at runtime.</p> <p>For details on creating and editing email templates, see Section 10.4, "Working with Email Templates," on page 216.</p>

Scheduling Synchronization and Cleanup

To configure the Synchronization and Cleanup Service:

- 1 Select the **RBPM Provisioning and Security** tab.
- 2 In the **Provisioning Configuration** group of actions, select **Delegation and Proxy** from the left navigation menu.
The user interface displays the Delegation and Proxy page. To schedule synchronization and cleanup, you need to make some changes in the Synchronization and Cleanup Service box.
- 3 To specify how often you want to activate the synchronization service, type the activation interval (in minutes) in the **Synchronization Service Activation Interval** field. The default value is 0, which means that the service is not activated.

When the synchronization service runs, any modifications (or deletions) made to delegate assignments are synchronized with the corresponding availability settings for the user.

- 4 To specify how often you want to activate the cleanup service, select **Cleanup Service Activation Interval**, then type the activation interval (in minutes). Alternatively, select **Cleanup Date** and use the calendar tool to specify the date when you want to activate the service. The default value is 0, which means that the service is not activated.

If no cleanup date is specified, the date is set to null. If no cleanup interval is specified, the interval is set to 0. When a cleanup date is specified, the interval is set to be 0. When an interval value other than 0 is specified, the date is set to null. If you check the cleanup interval option without putting in a number (the default is 0), the interface will show the original cleanup date after you submit the page, just as if you had not performed a submit.

When the cleanup service runs, all obsolete proxy and delegate assignments are removed from the system.

If the cleanup service has been activated, the **Last cleanup performed** field indicates when the last cleanup was performed.

7.2.2 Configuring the Digital Signature Service

This section provides details on configuring the Digital Signature Service.

To configure the Digital Signature Service:

- 1 Select the **RBPM Provisioning and Security** tab.
- 2 Select **Digital Signature Service** from the left navigation menu.

The user interface displays the Digital Signature Service panel.

- 3 Perform these steps to configure the Digital Signature Service:

- 3a** Select the **Enable Digital Signature Support** check box.

If this check box is not selected, users will see an error message when they try to access any provisioning resource that requires a digital signature.

Before enabling digital signature support, make sure all of the required JARs are present. If any of the JARs are missing, you will see an error message when you select the check box. For details on which JARs are required for digital signatures, see [Section 2.3, "Digital Signature Configuration," on page 52](#).

- 3b** Select the **Use XML Signature** check box if you want to use an XML Signature.

- 3c** Optionally select the **Enable Signed Document Preview** checkbox to allow users to preview signed documents.

- 3d** Select the **Use Digital Signature as a Service** checkbox to configure digital signatures with the Digital Signature Service.

The service for digital signatures is very easy to configure and manage. Do not deselect this checkbox, unless you are not ready to upgrade from an older digital signature implementation, or require the ability to read digital signature cards, since this support is not available with digital signature as a service.

IMPORTANT: We strongly encourage you to use the digital signature as a service support, since it is the most reliable and easy-to-use configuration for managing digital signatures.

When you select Use Digital Signature as a Service, the user interface automatically displays the name of the provider class for signature verification. You cannot edit the class name if you are using the digital signature service.

- 3e** (Not Required for Digital Signature as a Service) Type the name of the class for your digital signature service in the **Class Name** field.
- 3f** (Not Required for Digital Signature as a Service) Optionally specify an entity key in the **Alternative Certificate Subject Virtual Entity Key** field. The entity key maps to an entity defined in the data abstraction layer. The entity provides a calculated attribute that can be used instead of the LDAP common name to ensure that only authorized users can perform digital signing. In the Designer, you define the entity, giving the key any name you like. On the Digital Signature Service configuration panel, you specify the key for the entity you defined. The alternative subject is an optional feature that you can use to add an extra layer of protection.
- 3g** (Not Required for Digital Signature as a Service) Select **JKS** or **PKCS** as the Key Store Type for your configuration. The choice depends on how your application will store encrypted private keys locally. If you want to use the Java Key Store, then select JKS. If you would prefer to use a language-neutral approach, select PKCS.
- 3h** (Not Required for Digital Signature as a Service) Optionally select the **Certificate Authorization** check box to ensure that the authenticated user matches the user associated with the selected user certificate. When **Certificate Authorization** is enabled, the current user is not permitted to use a certificate on the smart card (or browser) that has been given to a different user.
- 3i** (Not Required for Digital Signature as a Service) Optionally select the **Enable Revocation Check** check box to cause the application to check the certificate revocation list (CRL) before using a certificate to be sure that it is still valid. A certificate might be revoked for several reasons. For example, the certificate authority might determine that a particular certificate was improperly issued. Alternatively, the certificate might be revoked if the private key for the certificate has been lost or stolen.
- 3j** (Not Required for Digital Signature as a Service) Optionally select the **Enable OCSP Query** check box to perform a query against an Online Certificate Status Protocol (OCSP) server before using a certificate. OCSP is an alternative to certificate revocation lists that addresses problems associated with using CRLs in a public key infrastructure (PKI). The OCSP access point for the server is specified in the User Application Configuration utility.
- 4** To view the settings for a previously configured applet, select the applet from the **Signature Applet** dropdown list.
- 5** Perform these steps to add a new signature applet configuration:
- 5a** Click **Add**.
- The user interface makes the fields in the Signature Applet panel editable.
- 5b** Provide a name for this applet configuration in the **Display Name** field.
- 5c** Specify the class ID for the applet in the **Class ID** field.

- 5d Specify the entry of the JAR that contains the applet in the **Archive Name** field.
 - 5e Specify <context root path> of the Web application that contains the applet archive for the **Context Root**. (If the context root points to a different application, always start it with a “/” character.)
 - 5f Specify the callback name in the **Callback Name** field.
 - 5g Specify the XML declaration string in the **Declaration Template** field.
 - 5h Specify the invocation string in the **Invocation Template** field.
 - 5i Specify the callback function in the **Callback Function Template** field.
 - 5j Select the browser type (for example, IE 6.0) in the **Browser Type** select list.
- 6 Click **Save** to save your settings.

7.2.3 Configuring the Provisioning UI Display Settings

This section provides instructions on configuring various user interface settings. Some of the settings control system-wide behavior within the User Application. Others are specific to the Work Dashboard.

To access the Provisioning UI Display Settings:

- 1 Select the **Administration** tab.
- 2 Select the **RBPM Provisioning and Security** tab.
- 3 Select **Provisioning UI Display Settings** from the left navigation menu.

The user interface displays the Provisioning UI Display Settings page. To configure the display settings for the Work Dashboard, you can make changes in the Task Settings and Request Status Settings box, which appear after the General Display Settings.

After you change the settings, you must restart JBoss in order for the changes to take effect.

Configuring the General Display Settings

The **Administration** tab in the User Application provides several settings you can use to control how result sets are processed and displayed on pages within the application. To configure the settings for result sets and pagination:

- 1 On the **Provisioning UI Display Settings** page, scroll down to the **General Display Settings** section of the page.
- 2 Modify any of the following settings, and click **Save**.

Setting	Description
Default number of results displayed per page	<p>Specifies the default number of rows to display in lists shown on the Roles and Resources tab.</p> <p>When a user initiates a query on any of the pages listed above, the User Application caches the data obtained by the query, and returns the number of rows specified for this setting to the browser. Each time the user requests to see the next page, another set of rows is returned from the cache.</p> <p>The default value for this setting is 25.</p>

Setting	Description
Options for number of results displayed per page (use spaces to separate values)	<p>Allows you to specify additional values that the user can select to override the default number of rows displayed on the My Roles, View Request Status, Browse Role Catalog, and Manage Role Relationships pages. The list of values you type must be separated by spaces.</p> <p>Note that the number specified in the Default number of results displayed per page control is always included in the list of values for the user to select.</p> <p>The default value for this setting is 5 10 50 100 500.</p> <p>NOTE: This setting also applies to the Team Tasks page on the Work Dashboard tab and to the Object Selector. The default number of rows displayed on the Team Tasks page and in the Object Selector, however, is not controlled by the Default number of results displayed per page setting. The default number of rows for team tasks is set at 5, and the default number of rows for the Object Selector is set at 10.</p>
Threshold for browser-based sorting and filtering	<p>Specifies the maximum amount of memory (expressed in rows) for the client browser to use for sorting and filtering. If you specify a very high value, client-side sorting and filtering will be very fast, but an excessive amount of memory might be used on the client. If you specify a very low value, the client-side memory usage might be low, but sorting and filtering might also be too slow.</p> <p>This setting applies only if the size of the result set is less than or equal to the threshold value. If the size of the result set is larger than the threshold value specified, sorting and filtering operations are performed on the server.</p> <p>The default value for this setting is 1000.</p>

Configuring the Task Settings

To configure the administrative settings for the Tasks list on the Work Dashboard:

- 1 Scroll down to the **Task Settings** box.
- 2 To specify whether you want the Task List to be displayed when users first open the dashboard, select either the **Yes** or **No** radio button for the **Expand Task List in default view of Work Dashboard** option.
- 3 To set the default sort column for the task list, pick the column in the **Task Notifications List default sort** field. Indicate whether the sort order will be ascending or descending by selecting or deselecting the **Descending** checkbox.

The default sort column is required in the task list display. When you select a default sort column, this column is automatically added to the **User default columns** list.

To allow the user to override the default sort column and sort order, click the the **Allow user to override** checkbox.

- To include a column in the task list, select it in the **Available Columns** list box, and drag them to the **User default columns** list box. To remove a column, select it in the **User default columns** list box and drag it to the **Available Columns** list box. You can select multiple columns to include or exclude by using the Ctrl or Shift key while clicking on the columns.

To allow the user to override the column selections you've made, click the **Allow user to override** checkbox. When you click this checkbox, the user interface displays the **Available columns for User override** list box. Any columns you add to the **Available columns for User override** list box are included in the **Available columns** list that the user sees on the Work Dashboard. To allow the user to override the default column list, select and drag one or more columns to the **Available columns for User override** list box from either the **User default columns** list box or the **Available Columns** list box. When you add a column to the **Available columns for User override** list box, that column is automatically removed from the list box from which you dragged it.

- To specify how the task details should be displayed when the user clicks on a task, select one of the following options:

Option	Description
In line with list	Displays the details within the Task Notifications list, directly under the task selected. This is the default.
In modal dialog	Displays the details in a separate dialog box that must appear on top of the Task Notifications list. After viewing the details for a task, the user needs to close the dialog to see the list again.

- To allow the user to claim a task automatically by simply opening the task details, select **yes** for the **Auto-claim when opening Task Details** option. When this option is set to **no**, the user must explicitly select **Claim** to claim a task.

Configuring the Request Status Settings

To configure the administrative settings for the Request Status list on the Work Dashboard:

- Scroll down to the Request Status Settings box.
- To set the default sort column for the request status list, pick the column in the **Request Status List default sort** field. Indicate whether the sort order will be ascending or descending by selecting or deselecting the **Descending** checkbox.

The default sort column is required in the request status list display. When you select a default sort column, this column is automatically added to the **User default columns** list.

To allow the user to override the default sort column and sort order, click the **Allow user to override** checkbox.

- To include a column in the request status list, select it in the **Available Columns** list box, and drag them to the **User default columns** list box. To remove a column, select it in the **User default columns** list box and drag it to the **Available Columns** list box. You can select multiple columns to include or exclude by using the Ctrl or Shift key while clicking on the columns.

To allow the user to override the column selections you've made, click the **Allow user to override** checkbox. When you click this checkbox, the user interface displays the **Available columns for User override** list box. Any columns you add to the **Available columns for User override** list box are included in the **Available columns** list that the user sees on the Work Dashboard. To allow the user to override the default column list, select and drag one or more columns to the **Available**

columns for User override list box from either the **User default columns** list box or the **Available Columns** list box. When you add a column to the **Available columns for User override** list box, that column is automatically removed from the list box from which you dragged it.

- To specify how the request status details should be displayed when the user clicks on one of the items requested, select one of the following options:

Option	Description
In line with list	Displays the details within the Request Status list, directly under the request selected. This is the default.
In modal dialog	Displays the details in a separate dialog box that must appear on top of the Task Notifications list. After viewing the details for a task, the user needs to close the dialog to see the list again.

7.2.4 Configuring the Workflow Engine and Cluster Settings

This section provides instructions on configuring the Workflow Engine and on configuring cluster settings. These settings apply to all engines in the cluster. When any of these settings are changed, other engines in the cluster will detect these changes in the database and use the new values. The engines check for changes to these settings at the same rate as specified by the pending process interval.

When the workflow engine starts up it checks to see if its engine ID is already in use by another node in the cluster. When this is the case, the workflow engine checks the cluster database to see if the status of the engine is SHUTDOWN or TIMEDOUT. If it is, the workflow engine starts. If the status is STARTING or RUNNING, the workflow engine logs a warning, then waits for a heartbeat time out to occur. If the heartbeat time out occurs, that means that the other workflow engine with the same ID was not shut down properly, so it's safe to start. If the heartbeat timer is updated, that means another workflow engine with the same ID is running in the cluster, so the workflow engine cannot start. You can specify the heartbeat time out (the maximum elapsed time between heartbeats before a workflow engine is considered timed out) by setting **Heartbeat Interval** and **Heartbeat Factor**. For more information about configuring these settings, see [“Configuring the Workflow Cluster” on page 170](#).

The process cache settings and heartbeat settings require a server restart to take effect.

Configuring the Workflow Engine

To configure the Workflow Engine settings:

- Select the **Provisioning** tab.
- Select **Engine and Cluster Settings** from the left navigation menu.

The user interface displays the Workflow Configuration Settings page. To configure the engine, you need to make some changes in the Workflow Engine box.

- To change an engine setting, click the target field for the setting and type the new value. The engine settings are described below:

Engine Setting	Description
Email Notification (per workflow engine)	Enables or disables email notifications for the entire workflow engine. Defaults to enabled.

Engine Setting	Description
Web Service Activity Timeout (minute)	Specifies the default Web Service activity timeout in minutes. The default is 50 minutes.
User Activity Timeout (hour, 0 for no timeout)	Specifies the default user activity timeout. The default is 0 days, which indicates no timeout.
Completed Process Timeout (day)	Specifies the number of days that a completed process state is kept in the workflow database system. The default is 120 days.
Completed Process Cleanup Interval (hour)	Specifies how often the engine checks for and removes completed processes that have been in the workflow database system for longer than the completed process timeout. The default is 12 hours.
Pending Process Interval (second)	User activities that are executed on an engine which the process is not bound to are put into a pending state. This interval specifies how often to check for pending activities in order to continue their execution. The default is 30 seconds.
Retry Queue Interval (minute)	Activities that fail because of suspected database connectivity issues are put on a retry queue. This interval specifies how often the engine attempts to retry these activities. The default is 15 minutes.
Maximum Thread Pool Size	The maximum number of threads that the engine uses to execute activities. The default is 20.
Minimum Thread Pool Size	The minimum number of threads that the engine uses to execute activities. When a thread is requested and fewer than the minimum are in the pool, a new thread will be created even if there are idle threads in the pool. The default is 10.
Initial Thread Pool Size	Number of prestarted threads in the pool when it is created. The default is 5.
Thread Keep Alive Time (second)	If the pool is larger than the minimum size, excess threads that have been idle for more than the keep alive time will be destroyed. The default is 5 minutes.
Process Cache Load Factor	The load factor specifies how full the cache is allowed to get before increasing its capacity. If the number of entries in the cache exceeds the product of the load factor multiplied by the current capacity, then the capacity is increased. The default is 0.75.
Process Cache Initial Capacity	The process cache is backed by a hash map. The capacity is the number of buckets in the hash map. The initial capacity is the number of buckets at the time the cache is created. The default is 700.

Engine Setting	Description
Process Cache Maximum Capacity	<p>Before adding a process to the cache, if the number of processes in the cache equals or exceeds the Process Cache Maximum Capacity, the cache attempts to remove the oldest inactive process from the cache. The maximum capacity is a soft limit, so the number of processes in the cache might exceed the Process Cache Maximum Capacity if there are no inactive processes (only active processes) in the cache.</p> <p>A good value for this setting should be less than product of the Process Cache Initial Capacity and the Process Cache Load Factor. This gives the cache a chance to remove older inactive processes from the cache before having to increase its capacity.</p> <p>Take the following example:</p> <p>Process Cache Initial Capacity = 700;</p> <p>Process Cache Load Factor = .75;</p> <p>Process Cache Maximum Capacity = 500;</p> <p>Number of processes in cache = 500;</p> <p>In this case, the number of processes in the cache that will trigger the cache to grow its capacity and perform a rehash would be 525, because the Initial capacity multiplied by the load factor is equal to 525.</p> <p>In this example, when there are 500 processes in the cache, the cache is approaching the point where it must increase its size and perform a rehash, which is at 525 processes. When another process is added to the cache, the engine attempts to remove the least recently used inactive process instead of letting the cache get closer to 525 processes.</p> <p>The default is 500.</p>
Maximum Engine Shutdown Timeout (minute)	<p>The engine attempts to shutdown gracefully. When shutting down it stops queuing new activities for execution and attempts to complete any activities already queued. This timeout specifies the maximum time that the engine waits for all queued activities and threads executing activities to complete. If this time is exceeded, the engine halts processing of queued activities and attempts to stop all threads executing activities. The default is 1 minute.</p>

Configuring the Workflow Cluster

To configure the Workflow Cluster settings:

- 1 Select the **Provisioning** tab.
- 2 Select **Engine and Cluster Settings** from the left navigation menu.

The user interface displays the Workflow Configuration Settings page. To configure cluster settings, you need to make some changes in the Workflow Cluster box.

Workflow Cluster

Modify any of the settings below to change the current cluster configuration. Review the list of each workflow engine in the cluster for engine ID and engine state. All fields are required.

Heartbeat Interval (second, minimum 60):

Heartbeat Factor (minimum 2):

Engine ID (Read Only)	Engine State (Read Only)
ENGINE	Running

- 3 To change a cluster setting, click the target field for the setting and type the new value. The cluster settings are described below:

Cluster Setting	Description
Heartbeat Interval (second, minimum 60)	<p>Specifies the interval at which the workflow engine's heartbeat is updated.</p> <p>When the workflow engine starts up, it detects if its engine ID is already being used by another node in the cluster and refuses to start if the ID is in use. The User Application database maintains a list of engine IDs and engine states. If an engine crashes and is restarted, its last state in the database indicates that it is still running. The workflow engine therefore uses a heartbeat timer, which writes heartbeats at the specified interval, to determine if an engine with its ID is still running in the cluster. If it's already running, it refuses to start.</p> <p>The minimum value for the heartbeat interval is 60 seconds.</p>
Heartbeat Factor (minimum 2)	<p>Specifies the factor that is multiplied with the heartbeat interval to arrive at the heartbeat timeout.</p> <p>The timeout is the maximum elapsed time permitted between heartbeats before an engine will be considered timed out.</p> <p>The minimum value for the heartbeat factor is 2.</p>

7.3 Administrator Assignments

The Administrator Assignments page allows you to assign users, groups, and containers to administrative roles. An administrator assignment specifies a domain type (Security, Provisioning, Role, Resource, Configuration, and Compliance), as well as a set of permissions for the assignment.

The Administrator Assignments page is accessible to the following users:

Table 7-1 User Access to the Administrator Assignments Page

User	Capabilities
Security Administrator	Can perform all operations on the Administrator Assignments page.
Other Domain Administrators	Can view administrator assignments and request assignments (or revoke assignments) for the domain over which this user has authority. He/she cannot view assignments or request assignments within another domain.

Delegated administrators of a domain have no access to this page.

The permissions for an administrator assignment define the actions that administrators can take on a particular scope of object instances within the domain type selected. For example, if you select the Role domain as the domain type for an assignment, the permissions determine what actions the administrators can take on the set of role instances selected as the scope for the assignment. These permissions might specify, for the selected scope of roles, that administrators can perform actions such as assigning roles to users, viewing role assignments, and reporting on role assignments.

Changing the Default Administrator Assignments The default administrator assignment settings are established at the time the User Application driver is initialized. After the driver has been initialized, you can change the default settings on the Administrator Assignments page, as long as your “admin” user account still exists. If the account has been deleted, deactivated, or moved to a different location, you will not be able to login to make the new assignments. In this case, you need to reset the values in the configupdate utility and delete the initialization property in the User Application Driver. Here are the basic steps you need to follow to do this:

1. Change the administrator assignment values in the configupdate utility.
2. Delete the initialization parameter in the User Application Driver.
3. Restart the User Application Driver and the Roles and Resources Driver.
4. Restart the User Application.

For complete details, see [Section 2.11, “Changing the Default Administrator Assignments After Installation,” on page 72](#).

7.3.1 Viewing Administrator Assignments

To view existing administrator assignments:

- 1 Select **Administrator Assignments** on the **RBPM Provisioning & Security** tab.

The Administrator Assignments page displays the list of administrator assignments currently defined.

7.3.2 Creating New Assignments

To define a new administrator assignment:

- 1 Click the **Assign** button at the top of the Administrator Assignments display.

The **New Administrator Assignment** dialog displays.

2 Select one of the following domains:

- ◆ The **Compliance** domain defines rights to launch attestation requests and view the status of attestation requests.
- ◆ The **Configuration** domain defines rights to configure access to User Application header tabs and navigation items.
- ◆ The **Provisioning** domain defines rights to launch and retract process requests, manage addressee tasks, and configure delegate, proxy, and availability settings.
- ◆ The **Reports** domain defines report permissions, which include the ability to use the Identity Manager Reporting tool.

NOTE: To access the existing role and resource reports, a user who is a Report Administrator must be assigned as Role or Resource Manager, and be given the **Report on Role** or **Report on Resource** permission for a specific role or resource, or for all roles or resources.

- ◆ The **Resource** domain defines rights to manage resources, assign, revoke, and report on resources, as well as rights to configure resource settings and bind entitlements.
- ◆ The **Role** domain defines rights to manage roles and SoDs, assign, revoke, and report on roles, as well as rights to configure role settings.
- ◆ The **Security** domain defines rights to manage User Application security, such as assign and revoke domain administrators, domain managers, and teams.

The domain determines what types of objects the administrator can act on. An administrator assignment can only be associated with a single domain.

NOTE: If a particular user has been designated as a manager of a team, NetIQ recommends that this user should not also be designated as a domain administrator for the domain associated with the team.

To see a description of a particular domain, click the Info icon to the right of the **Domain** list.

3 Specify one of the following choices for the **Type of Assignment**:

- ◆ **User**
- ◆ **Group**
- ◆ **Container**
- ◆ **Role**

4 Select the users (or groups, containers, or roles) in the **Select Users** field.

The label for the control, and the objects available for selection, vary according to the type of assignment you've specified.

5 Select an **Effective Date** for the assignment. This date (and time) determines when the permissions are enabled for the assignment.

6 Select an **Expiration Date** for the assignment. This date (and time) determines when the permissions are disabled for the assignment.

7 To give the administrator full permissions for the selected domain, click the **All Permissions** checkbox.

When the **All Permissions** checkbox is checked, the assignment creates a Domain Administrator. When it is unchecked, the assignment creates a Delegated Administrator.

When the domain selected is **Security**, **Configuration**, **Compliance**, or **Reports**, the assignment automatically gives full permissions for the selected domain, and the **All Permissions** checkbox is not displayed.

NOTE: When a user is assigned a Compliance Administrator role, the user interface shows two rows in the Administrator Assignments page, one for the Compliance Administrator role, and one for a Provisioning Manager role with no permissions visible. Note that this latter row should not be removed. If the row is removed, the user assigned to be Compliance Administrator will not be able to launch attestation requests successfully. The Compliance Administrator role is automatically given rights to initiate and retract attestation provisioning requests. For this reason, the Provisioning Manager role is required.

- 8 Click **Save** to preserve your administrator assignment settings.

If the domain for the assignment is **Provisioning**, **Role**, or **Resource** domain, and you've unchecked the **All Permissions** checkbox, the **Permissions** section is added to the page.

- 9 To define the permissions, click **New**.

This interface shows controls that apply to the domain selected for the assignment. These controls allow you to specify which objects are within the scope of the assignment and which permissions administrators have with respect to these objects.

- 10 Follow these steps to define permissions for an assignment that uses the **Provisioning** domain:

10a To include all provisioning request definitions, click the **All Provisioning Request Definition** button.

10b To select provisioning request definitions individually, choose the **Select Provisioning Request Definition** radio button, and use the Object Selector to pick one provisioning request definition at a time.

NOTE: If you select All Provisioning Request Definitions, and define a permission at this level, and then try to define the same permission for a particular provisioning request definition, the Administration Assignment page will not create the permission for the provisioning request definition, since it has already been defined at a higher level. In general, a permission will not be set on a lower level object if it has been already defined for a higher level object. However, if it is defined on a lower level object first, the same permission can be set on a higher level set of objects as well.

- 10c** Once you've defined the scope, choose the permissions you want to allow for each object by selecting the object and picking the desired permissions in the list on the right side of the dialog.

Permission	Description
Initiate PRD	Allows the user to initiate the selected provisioning requests. NOTE: The Initiate PRD permission has no effect on the behavior of the NetIQ-installed PRDs for resources, roles, and attestation within the User Application, since these PRDs cannot be initiated directly from the User Application. However, this permission does control whether these PRDs can be initiated from a SOAP call.
Retract PRD	Allows the user to retract the selected provisioning requests when they are in progress.
View Running PRD	Allows the user to view the selected provisioning requests when they are in progress.

Permission	Description
Configure Delegate	Allows the user to configure delegate assignments for the selected provisioning requests.
Manage Addressee Task	Allows the user to manage tasks associated with the selected provisioning requests that have been addressed to other users. When this permission is enabled, Domain and Delegated Administrators can manage tasks for all users, including addresses and recipients. Team Managers are able to manage tasks for addressees, but not for recipients.
Configure Availability	Allows the user to configure availability for tasks associated with the selected provisioning requests.

- 10d** In the **Add User Application Driver Permissions** section of the page, optionally select the **Configure Proxy** permission to allow the selected user(s) to configure proxy assignments. This setting applies to the driver as a whole.
- 10e** Click **Save** to save the permissions for the selected objects or containers.
To delete a permission, select the permission and click **Delete**.
To refresh the list of permissions, click **Refresh**.
- 11** Follow these steps to define permissions for an assignment that uses the **Role** domain:
- 11a** To include all roles in all levels in the roles hierarchy, choose **All Role Levels** in the **Role Level** control.
To include all roles at a particular level in the role hierarchy, choose one of the following levels:
- ◆ **Business Role**
 - ◆ **IT Role**
 - ◆ **Permission Role**
- To include all roles in a particular sub container under the selected role level, use the Object Selector to select the sub container.
- 11b** To select roles individually, choose **Select Roles** radio button, and use the Object Selector to pick one or more roles.
- 11c** Once you've defined the role scope, choose the permissions you want to allow for each object by selecting the object and picking the desired permissions in the list on the right side of the dialog.

Permission	Description
Create Role	Allows the user to create roles. This permission is hidden when a particular role is selected.

Permission	Description
Delete Role	<p>Allows the user to delete the selected roles.</p> <p>This setting applies only at the container level.</p> <p>At installation time, no user has the ability to delete system roles. However, the administrator may grant user access to the system roles. The permission to delete roles should not be given for the RoleConfig, Level20, and System roles containers. Also, in general, you should not set permissions on those containers, because permissions on these containers will be propagated to the system roles. Instead, you should create role subcontainers under the role level container and set permissions on each subcontainer.</p>
Update Role and Role Relationship	<p>Allows the user to update the selected roles and modify role relationships.</p> <p>This setting applies only at the container level.</p>
View Role	<p>Allows the user to view the selected roles.</p> <p>This setting applies only at the container level.</p>
Assign Role To User	<p>Allows the user to assign users to the selected roles.</p> <p>IMPORTANT: Only the Security Administrator can assign system roles on the Work Dashboard tab and the Roles and Resources tab.</p>
Revoke Role From User	<p>Allows the user to revoke user assignments for the selected roles.</p>
Assign Role To Group And Container	<p>Allows user to assign groups and containers to the selected roles.</p>
Revoke Role From Group And Container	<p>Allows the user to revoke group and container assignments for the selected roles.</p>
Report On Role	<p>Allows the user to generate reports that provide information about the selected roles.</p>

- 11d** To include all separation of duties constraints, choose **All Separation of Duties Constraints** radio button.
- 11e** To select separation of duties constraints individually, choose **Select Separation of Duties Constraint** radio button, and use the Object Selector to pick one or more constraints.
- 11f** Once you've defined the separation of duties scope, choose the permissions you want to allow for each object by selecting the object and picking the desired permissions in the list on the right side of the dialog.

Permission	Description
Create SoD	Allows the user to create separation of duties constraints. This permission is hidden when a particular SoD constraint is selected.
Update SoD	Allows the user to update the selected separation of duties constraints.
Delete SoD	Allows the user to delete the selected separation of duties constraints.
View SoD	Allows the user to look at the selected separation of duties constraints.
Report On SoD	Allows the user to generate reports that provide information about the selected separation of duties constraints.

- 11g** In the **Add Role Configuration Permissions** section of the page, optionally select the **Configure Roles Settings** permission for the configuration object.

This setting controls access to the **Configure Role and Resource Settings** page on the **Roles and Resources** tab. To access this page, the user must have the **Configure Roles Settings** permission as well as the **Configure Resource Settings** permission, which is given through a Resource Manager (or Resource Administrator) assignment. If a user does not have both of these permissions, the **Configure Roles and Resource Settings** page displays read-only information, and cannot be edited.

- 11h** Click **Save** to save the permissions for the selected objects or containers.

To delete a permission, select the permission and click **Delete**.

To refresh the list of permissions, click **Refresh**.

- 12** Follow these steps to define permissions for an assignment that uses the **Resource** domain:

- 12a** To include all resources, click the **All Resources** button.

- 12b** To select resources individually, choose the **Select Resources** radio button, and use the Object Selector to pick one or more resources.

- 12c** Once you've defined the resource scope, choose the permissions you want to allow for each object by selecting the object and picking the desired permissions in the list on the right side of the dialog.

Permission	Description
Create Resource	Allows the user to create resources. This permission is hidden when a particular resource is selected.
Delete Resource	Allows the user to delete the selected resources.
Update Resource	Allows the user to update the selected resources.
View Resource	Allows the user to view the selected resources.
Assign Resource	Allows the user to assign users to the selected resources.

Permission	Description
Revoke Resource	Allows the user to revoke user assignments for the selected resources.
Report On Resource	Allows the user to generate reports that provide information about the selected resources.

- 12d** To include all drivers for entitlements, click the **All Drivers** radio button.
- 12e** To select drivers individually, choose the **Select Driver** radio button, and use the Object Selector to pick a resource.
- 12f** Once you've defined the driver scope, optionally select the **Bind Entitlement** permission to allow the selected user(s) to bind resources to entitlements. To allow the user to generate reports on entitlements, optionally select the **Report On Entitlement** permission.
- 12g** In the **Add Resource Configuration Permissions** section of the page, optionally select the **Configure Resources Settings** permission for the configuration object.
- This setting controls access to the **Configure Role and Resource Settings** page on the **Roles and Resources** tab. To access this page, the user must have the **Configure Resources Settings** permission as well as the **Configure Roles Settings** permission, which is given through a Role Manager (or Role Administrator) assignment. If a user does not have both of these permissions, the **Configure Roles and Resource Settings** page displays read-only information, and cannot be edited.
- 12h** Click **Save** to save the permissions for the assignment.
- To delete a permission, select the permission and click **Delete**.
- To refresh the list of permissions for the assignment, click **Refresh**.
- 13** Click **Save** to save the assignment and permissions.

7.3.3 Editing an Existing Assignment

To edit an existing administrator assignment:

- 1 Select a previously defined assignment and click **Edit**.
- 2 Make your changes to the administrator settings and click **Save**.

7.3.4 Deleting Assignments

To delete an assignment:

- 1 Select a previously defined assignment and click **Edit**.

7.3.5 Refreshing the Assignment List

To refresh the list of administrator assignments:

- 1 Click **Refresh**.

7.4 Team Configuration

The Team Configuration page allows you to create teams and define permissions for these teams. A team definition specifies a domain type (Provisioning, Role, or Resource), as well as a set of team members and managers. The Team Configuration page is accessible to the following users:

Table 7-2 User Access to the Team Configuration Page

User	Capabilities
Security Administrator	Can perform all operations on the Team Configuration page.
Other Domain Administrators	Can define a team for the domain over which the administrator has authority.
Team Manager	Can view a team definition for which he/she is configured to be the manager. When a team manager edits a team, the team definition itself is read-only, because the team manager cannot modify the team configuration.

The members of a team can be specified individually as a set of users, groups, or containers, or can be defined based on a business relationship, such as the Manager-Employee relationship. Alternatively, the team member list can include all users within the container.

When a team definition includes a container or group in its membership list, the User Application expands the list within the container or group to show the users within the container or group. Therefore, the User Application only allows the team manager to specify a particular user within the container or group as the recipient for a team request; the team manager is not permitted to specify a container or group as the recipient for a team request.

The managers for a team can be a one or more users or groups. When you define a team, you can specify whether you want the team managers to also be members of the team.

The permissions for a team define the actions that team members can take on a particular scope of object instances within the domain type selected for a team. For example, if you select the Role domain as the domain type for a team, the team permissions determine what actions the members can take on the set of role instances selected as the scope for the team. These permission might specify, for the selected scope of roles, that members can perform actions such as assigning roles to users, viewing role assignments, and reporting on role assignments.

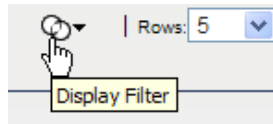
7.4.1 Viewing Team Configurations

To view existing team configurations:

- 1 Select **Team Configuration** on the **RBPM Provisioning and Security** tab.
The Team Configuration page displays a list of team configurations currently defined.

Filtering the Team List

- 1 Click the Display Filter button in the upper right corner of the Resource Catalog display.



- 2 Specify a filter string for the team name or description in the Filter dialog, or select a particular domain, and click **Filter**:
- 3 To remove the current filter, click **Reset**.

Setting the Maximum Number of Rows on a Page

- 1 Click on the Rows dropdown list and select the number of rows you want to be displayed on each page:.

Scrolling within the Team List

- 1 To scroll to another page in the resource list, click on the Next, Previous, First or Last button at the bottom of the list.

Sorting the Team List

To sort the team list:

- 1 Click the header for the column you want to sort on.
The pyramid-shaped sort indicator shows you which column is the new sort column. When the sort is ascending, the sort indicator is shown in its normal, upright position.
When the sort is descending, the sort indicator is upside down.
The default sort column is the Resource Name column.

If you override the default sort column, your sort column is added to the list of required columns. Required columns are indicated with an asterisk (*).

When you modify the sort order for the task list, your preference is saved in the Identity Vault along with your other user preferences.

7.4.2 Creating New Teams

To define a new team:

- 1 Click the **New** button at the top of the Team Configuration display.
The **New Team** dialog displays.
- 2 Select one of the following domains:
 - ◆ **Provisioning Domain**
 - ◆ **Role Domain**
 - ◆ **Resource Domain**

The domain determines what types of objects the team members can act on. A team can only be associated with a single domain.

NOTE: If a particular user has been designated as a domain administrator, NetIQ recommends that this user should not also be designated as a manager of a team for the same domain for which the user is a domain administrator.

- 3 Provide a name and description for the team.
- 4 In the **Managers** control, select the users and groups that will be managers of the team.
- 5 In the **Members** control:
 - 5a Indicate whether the managers will also be members of the team by selecting or deselecting the **Also include selected managers in members list** checkbox.
 - 5b Define the members of the team by selecting one of the following radio buttons:

Option	Description
All Users	Includes all users in the container.
Relationship	Includes all users that have a relationship with the users in the Managers list. For example, if you select the Manager-Employee relationship, the members report directly to the users in the Managers list.
Select Members	Includes the users, groups, and containers you select.

- 6 Click **Save** to preserve your team configuration settings.

Once you've saved a team, the **Permissions** section is added to the page, and the Team Permissions Configuration interface is displayed.

The Team Permissions Configuration interface includes buttons for adding new permissions, deleting permissions and refreshing the display. The Permissions section of the page does not include an **Edit** button because the details associated with each permission are shown in the Permissions list. If a particular team permission is not properly defined, you can simply delete the permission and add a new one in its place.
- 7 To define the permissions for the team, click **New**.

This interface shows controls that apply to the domain selected for the team. These controls allow you to specify which objects are within the scope of the team and which permissions team members have with respect to these objects.
- 8 Follow these steps to define permissions for a team that uses the **Provisioning** domain:
 - 8a To include all provisioning request definitions, click the **All Provisioning Request Definition** button.
 - 8b To select provisioning request definitions individually, choose the **Select Provisioning Request Definition** radio button, and use the Object Selector to pick one or more provisioning request definitions:.
 - 8c Once you've defined the scope for the team, choose the permissions you want to allow for each object by selecting the object and picking the desired permissions in the **Permissions** control.

The provisioning permissions are the same for team configurations as for RBPM administrator assignments. See [Step 10c on page 174](#) for details on the provisioning permissions.

- 8d** To define permissions that apply to the User Application driver as a whole, open the **Add User Application Driver Permissions** section of the page and select the permissions you want to allow with this assignment.
- 8e** Click **Save** to save the permissions for the selected objects or containers.
To delete a permission, select the permission and click **Delete**.
To refresh the list of permissions for the team, click **Refresh**.
- 9** Follow these steps to define permissions for a team that uses the **Role** domain:
- 9a** To include all roles in all levels in the roles hierarchy, choose **All Role Levels** in the **Role Level** control.
To include all roles at a particular level in the role hierarchy, choose one of the following levels:
- ◆ **Business Role**
 - ◆ **IT Role**
 - ◆ **Permission Role**
- To include all roles in a particular sub container under the selected role level, use the Object Selector to select the sub container.
- 9b** To select roles individually, choose **Select Roles** radio button, and use the Object Selector to pick one or more roles.
- 9c** Once you've defined the role scope for the team, choose the permissions you want to allow for each object by selecting the object and picking the desired permissions in the **Permissions** control.
The following role permissions are supported in team configurations:
- ◆ View Role
 - ◆ Assign Role
 - ◆ Revoke Role
 - ◆ Assign Role to Group and Container
 - ◆ Revoke Role from Group and Container
- These role permissions have the same behavior as for RBPM administrator assignments. See [Step 11c on page 175](#) for details on these role permissions.
- 9d** Click **Save** to save the permissions for the selected objects or containers.
To delete a permission, select the permission and click **Delete**.
To refresh the list of permissions for the team, click **Refresh**.
- 10** Follow these steps to define permissions for a team that uses the **Resource** domain:
- 10a** To include all resources, click the **All Resources** button.
- 10b** To select resources individually, choose the **Select Resources** radio button, and use the Object Selector to pick one or more resources.
- 10c** Once you've defined the resource scope for the team, choose the permissions you want to allow for each object by selecting the object and picking the desired permissions in the **Permissions** control.
The following resource permissions are supported in team configurations:
- ◆ View Resource
 - ◆ Assign Resource
 - ◆ Revoke Resource

These resource permissions have the same behavior as for RBPM administrator assignments. See [Step 12c on page 177](#) for details on these resource permissions.

10d Click **Save** to save the permissions for the team.

To delete a permission, select the permission and click **Delete**.

To refresh the list of permissions for the team, click **Refresh**.

11 Click **Save** to save the team configuration and team permissions.

7.4.3 Editing an Existing Team

To edit an existing team:

1 Select a previously defined team and click **Edit**.

When a team manager edits a team, the team definition itself is read-only, because the team manager cannot modify the team configuration.

2 Make your changes to the team settings and click **Save**.

7.4.4 Deleting Teams

To delete an existing team:

1 Select a previously defined team and click **Delete**.

7.4.5 Refreshing the Team List

To refresh the list of teams:

1 Click **Refresh**.

7.5 Navigation Access Permissions

The Navigation Access Permissions page allows you to set the access permissions for some of the navigation items within the User Application. It allows you to control access to three of the main header tabs with the application: Roles and Resources tab, Identity Self-Service tab, and Work Dashboard tab. In addition, it allows you to define permissions for lower-level navigation items within the Provisioning and Security, Roles and Resources, and Work Dashboard areas of the application.

NOTE: The Compliance and Administration tabs cannot be configured through the Navigation Access Permissions page. The Compliance tab is only visible to Compliance Administrators, and the Administration tab is only visible to Security Administrators, Domain Administrators (such as the Role Domain Administrator and Resource Domain Administrator), and Configuration Administrators.

To define navigation access permissions:

1 Select **Navigation Access Permissions** on the **RBPM Provisioning & Security** tab.

The Navigation Access Permissions page displays.

2 Click on the **Name** drop-down list to see the navigation items for which you can define permissions.

The navigation areas appear in bold. Within each area, you can see the items.

3 Select the navigation item for which you want to define permissions.

- 4 Select one or more trustees for the navigation item. When a trustee logs on to the User Application, the navigation item is displayed. Otherwise, the navigation item is hidden. You can add users, groups, roles, and containers as trustees.

Each navigation item has a set of default trustees that is suitable for the services that can be accessed through the navigation item. Most of the navigation items listed are self-explanatory. For those items that require additional explanation, you can find details below.

Make a Process Request By default, the Make a Process Request navigation item is shown on the Work Dashboard. To hide the Make a Process Request item, remove all trustees for this item. If you remove all trustees, only Configuration Administrators will be able to see the item. To show the Make a Process Request item on the Work Dashboard again, select **Make a Process Request** and choose the users, groups, roles, or containers that you want to be able to access the item.

Assign Resource Controls whether you see the **Assign** button on the **Resource Assignments** section of the Work Dashboard. The root container is specified as the default trustee for this permission at installation time.

Remove Resource Assignments Controls whether you see the **Remove** button on the **Resource Assignments** section of the Work Dashboard. The root container is specified as the default trustee for this permission at installation time.

Assign Role Controls whether you can see the **Assign** button on the **Role Assignments** section of the Work Dashboard. The root container is specified as the default trustee at installation time.

Remove Role Assignments Controls whether you can use the **Remove** button on the **Role Assignments** section of the Work Dashboard. The root container is specified as the default trustee at installation time.

- 5 To make the currently selected navigation item the default for the navigation area, select **Check to make this the default navigation item for selected area**.

The **Check to make this the default navigation item for selected area** control is not available for navigation items within the Work Dashboard area.

- 6 Click **Save**.

If you add a user as a trustee for a navigation item, and this user is a member of a container that was previously added as a trustee, this user will have access to the navigation item, but will not be added to the list of trustees.

NOTE: If a user does not have access to the default tab (or to the default menu item within a navigation area), the User Application will attempt to display a tab (or menu item) for which the user has authorization. If the user has not been given authorization for any tab or menu item, the default page will display. If the user is not authorized for the default page, or if the user goes directly to an unauthorized bookmark, an error message is displayed indicating that the user does not have the proper authorization.

If the user has been authorized to access a tab, but nothing under the tab, the page will still show and an error message will be displayed indicating that the user does not have the proper authorization. Conversely, if the tab has not been authorized, the tab will not show. However, if the user is authorized to access menu items under the tab, the user will be able to access these menu items by using bookmarks.

Proxy Mode When a user is in proxy mode, the navigation access permissions for menu items on the Dashboard will show the proxied user's permissions, not the permissions for the logged in user. For all other navigation, the menu items will be controlled by the permissions set for the logged in user. The **Manage** control (for selecting a user, group, role, or container) is not available in proxy mode, even if a user is proxying for a user that is a Domain Administrator or Domain Manager.

IV **Configuring and Managing Provisioning Workflows**

These sections describe how to configure and manage provisioning requests and workflows:

8 Configuring the User Application Driver to Start Workflows

This section describes the User Application driver and how to configure it to automatically trigger a workflow based on an event in the Identity Vault.

8.1 About the User Application Driver

The User Application driver is responsible for starting provisioning workflows and for notifying the User Application of changes in the Identity Vault (for example, when you make changes to the directory abstraction layer using the Designer for Identity Manager). Only the Subscriber channel is used in this driver. The driver processes messages from the Identity Vault to the User Application running on an application server. Although there are events that occur in the User Application that are reported back to the Identity Vault, these events do not flow through the Publisher channel of the User Application driver.

When the application server is started, the driver establishes a session with the application server. The driver sends messages to the User Application running on the application server (for example, “retrieve a new set of virtual directory definitions”).

The following table lists the source components of the driver.

File Name	Description	Windows Installation Directory	Linux Installation Directory
ComposerDriverShim.jar	The Composer Driver Shim	\Netiq\NDS\lib	/usr/lib/dirxml/classes
srvprvUAD.jar	The Application Driver Shim	\Netiq\NDS\lib	/usr/lib/dirxml/classes
UserApplicationDriver.xml	A file that contains configuration data for setting up the new driver.	\Tomcat\webapps\nps\DirXML.Drivers	/opt/netiq/eDirectory/lib/dirxml/rules/ or /var/opt/netiq/iManager/nps/DirXML.Drivers

The User Application driver components are installed when you install Identity Manager. Before you can run the Identity Manager User Application, you must add the User Application driver to a new or existing driver set, and activate the driver.

Depending on your work environment, very little configuration of the User Application driver might be required, or you might want to implement a complex set of business rules in the driver policies. The User Application driver provides the same flexible mechanisms for data synchronization as other Identity Manager drivers.

8.2 Setting Up Workflows to Start Automatically

Workflows are automatically started when a user starts a provisioning request by requesting a resource. In addition, the Identity Manager User Application driver listens for events in the Identity Vault and, when configured to do so, responds to events by starting the appropriate provisioning workflows. For example, you can configure the User Application driver to automatically start a provisioning workflow if a new user is added to the Identity Vault. You configure the User Application driver to automatically start workflows using Identity Manager policies and rules.

8.2.1 About Policies

You can use filters and policies with the User Application driver in the same way that you can with other Identity Manager drivers. When an event occurs in the Identity Vault, Identity Manager creates an XML document that describes the event. The XML document is passed along the channel to the connected system (in this case, the connected system is the User Application). Filters and policies associated with a driver allow you to define how to respond to the event, and in the process transform that XML document to the format that is expected by the connected system. Identity Manager provides several categories of policies (for example, Event Transformation, Command Transformation, Schema Mapping, Output Transformation) that you can apply, in a prescribed order, to transform the XML document.

This section provides an example of starting a workflow based on events in the Identity Vault. Although any of the policies can be used to trigger a workflow, the example presented in this section demonstrates the easiest and most useful method.

When you create a User Application driver, an Event Transformation Policy is created for use by the driver. The Event Transformation Policy is responsible for creating the XML document that is processed by the remaining Subscriber channel policies.

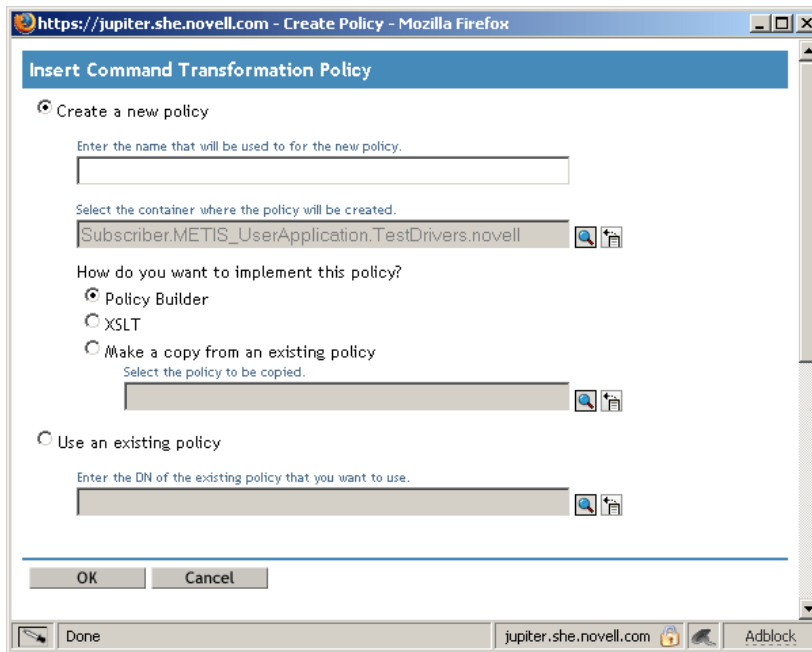
NOTE: Do not change the Event Transformation policy that was created when the User Application driver was created. The DN of this policy begins with `Manage.Modify.Subscriber`. Changing this policy might cause the workflow process to fail.

An empty Schema Mapping Policy is also created. You can use this policy as a starting point for triggering a workflow, based on events in the Identity Vault.

8.2.2 Using the Policy Builder

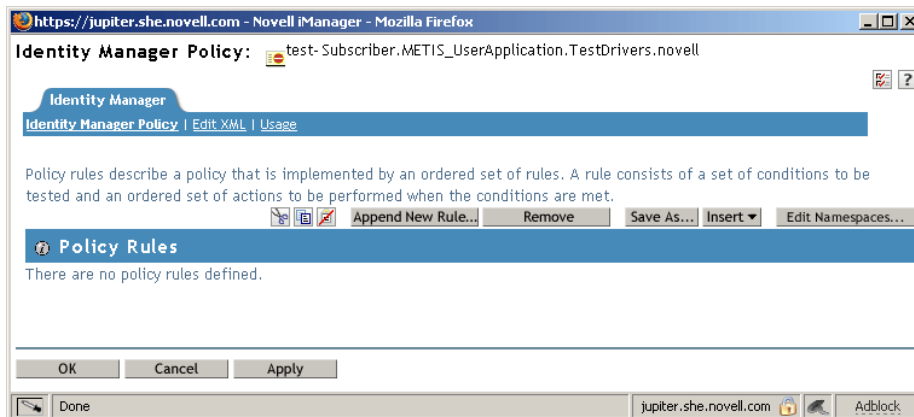
The easiest way to automatically start a workflow based on an Identity Vault event is to use the Policy Builder. The Policy Builder provides a Start Workflow action that simplifies the process of setting up a workflow to start automatically.

- 1 In iManager, expand the **Identity Manager Role**, then click **Identity Manager Overview**.
- 2 Specify a driver set.
- 3 Click the driver for which you want to manage policies. The **Identity Manager Driver Overview** opens.
- 4 Click the policy that you want to edit.
- 5 Click **Insert** to open the Policy Builder.

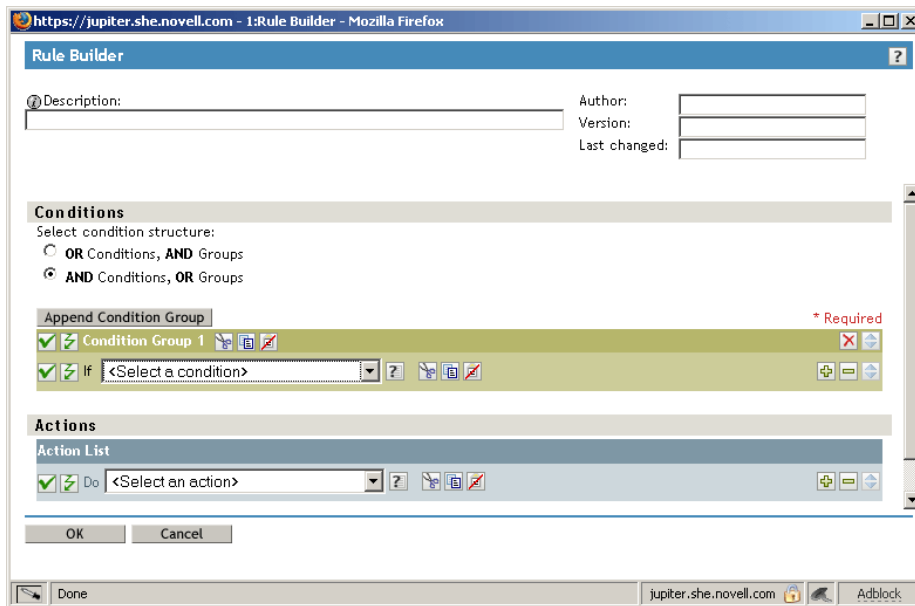


- 6 Click **Create a new policy**.
- 7 Type a name for the policy.
- 8 Click **Policy Builder**.
- 9 Click **OK**.

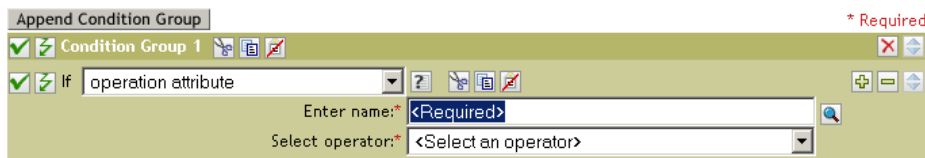
iManager displays a screen that lists defined policy rules.



- 10 Click **Append New Rule**.
- iManager displays the Rule Builder.

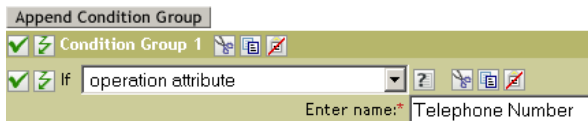


- 11 Type a **Description** for the rule.
- 12 Select **operation attribute** for the **If** condition in **Condition Group 1**.

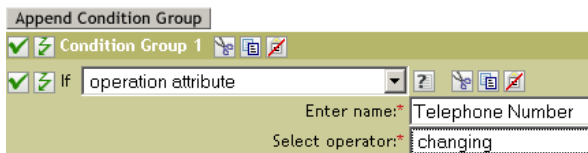


- 13 Use the **Browse attributes** button for the **Enter name** field to specify the Identity Vault attribute that you want to use to start the workflow.

For example, to start a workflow when a telephone number changes, select the **Telephone Number** attribute.



- 14 Use the **Select Operator** list to select the operator to use to test the specified attribute.
- For example, to start a workflow when a telephone number changes, select **changing**.



- 15 Select **start workflow** from the **Action** list.

Append Condition Group

Condition Group 1

If operation attribute

Enter name:* Telephone Number

Select operator:* changing

Actions

Action List

Do start workflow

- 16 Use the Object Selector in the **Enter provisioning request DN** field to select the provisioning request definition that you want to be executed when the if condition is true.

Append Condition Group

Condition Group 1

If operation attribute

Enter name:* Telephone Number

Select operator:* changing

Actions

Action List

Do start workflow

Enter provisioning request DN:* CN=TestSingleApprovalTA,CN=RequestDefs,CN=AppConfig,CN=METIS_UserApplic

The **Enter user application URL** and **Enter authorized user DN** fields are filled in automatically.

- 17 Type the password for the User Application administrator in the **Enter authorized user password** field.

Append Condition Group

Condition Group 1

If operation attribute

Enter name:* Telephone Number

Select operator:* changing

Actions

Action List

Do start workflow

Enter provisioning request DN:* CN=TestSingleApprovalTA,CN=RequestDefs,CN=AppConfig,CN=METIS_UserApplic

Enter user application URL:* http://164.99.26.207:8080/ADMPROV

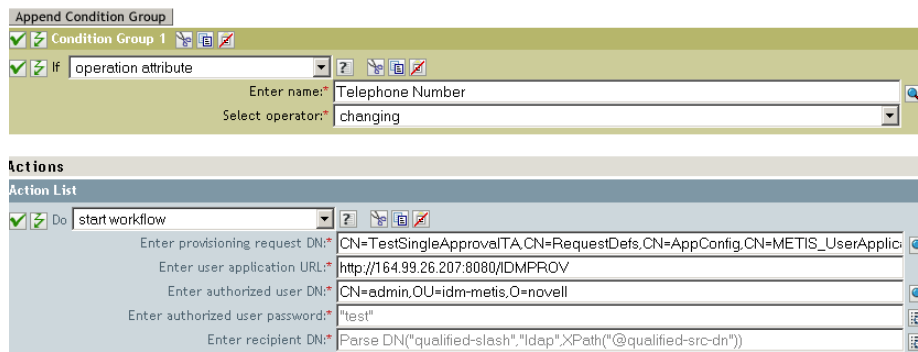
Enter authorized user DN:* CN=admin,OU=idm-metis,O=novell

Enter authorized user password:* "test"

We recommend using a named password, because typing a password in clear text is a security risk.

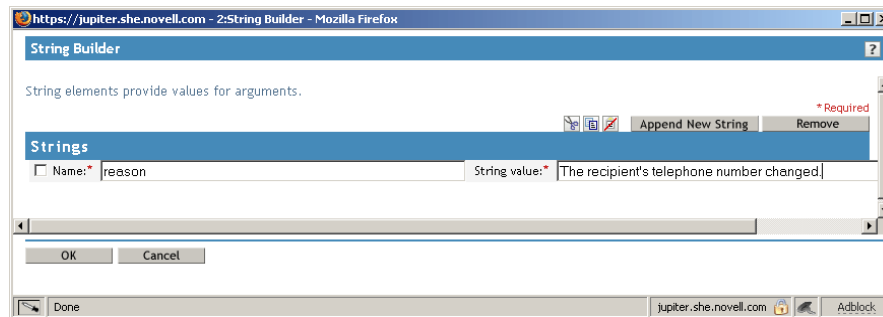
- 18 In the **Enter recipient DN** field, specify the DN of the recipient of the workflow in LDAP format. The expression for the recipient DN must evaluate to a DN that conforms to RFC 2253 format (in other words, cn=user,ou=organizational unit,o=organization). For example, you can click the **Argument Builder** button in the **Enter recipient DN** field to create the following expression to pass the recipient's DN to the workflow:

```
Parse DN("qualified-slash", "ldap", XPath("@qualified-src-dn"))
```



- 19 Specify the arguments for the workflow in the **Enter additional arguments** field.

You must use this field to specify the **reason** attribute, which is required by the workflow. You can click the **String Builder** button in the **Enter additional arguments** field to specify the **reason** attribute and create a value for the attribute (for example, “the recipient’s telephone number has changed”).



- 20 Click **OK** to close the Rule Builder.
- 21 Click **OK** to close the Policy Builder.
- 22 Click **OK** to close the Policies screen.
- 23 Make sure that you add any attributes needed by the workflow to the filter.

In the example described in this procedure, you would need to add **Telephone Number** and **CN** to the filter.

8.2.3 Using the Schema Mapping Policy Editor

The Schema Mapping Policy Editor provides an alternative method of starting a workflow automatically, by mapping Identity Vault attributes to workflow runtime data. To get you started, the User Application driver provides an empty policy to edit. Workflow runtime data is available from the workflow definition template described in [Chapter 9, “Managing Provisioning Request Definitions,” on page 201](#).

When a workflow is created, the following global attributes are created in the Identity Vault:

- ◆ `<workflowName>_StartWorkflow`. This attribute starts a workflow.
- ◆ `<workflowName>_recipient`. This attribute accepts runtime data needed by the workflow from the Identity Vault.
- ◆ `<workflowName>_reason`. This attribute accepts runtime data needed by the workflow from the Identity Vault.

Two other attributes always exist and accept runtime data needed by the workflow from the Identity Vault:

- ◆ AllWorkflows:reason
- ◆ AllWorkflows:recipient

Ensure you have the following information before you set up a workflow to start based on an event in the Identity Vault:

- ◆ The name of the Identity Vault attribute that you want to use as a trigger for the workflow
- ◆ The name of the workflow that you want to start. All workflows include a special attribute named `<workflowName>_StartApprovalFlow`. You configure a workflow to start automatically based on an event in the Identity Vault by mapping the desired eDirectory attribute to the `<workflowName>_StartApprovalFlow` attribute for the workflow.

To set up a workflow to start based on an event in the Identity Vault:

- 1 In iManager, click the **Identity Manager Overview** link under Identity Manager in the iManager navigation tree.



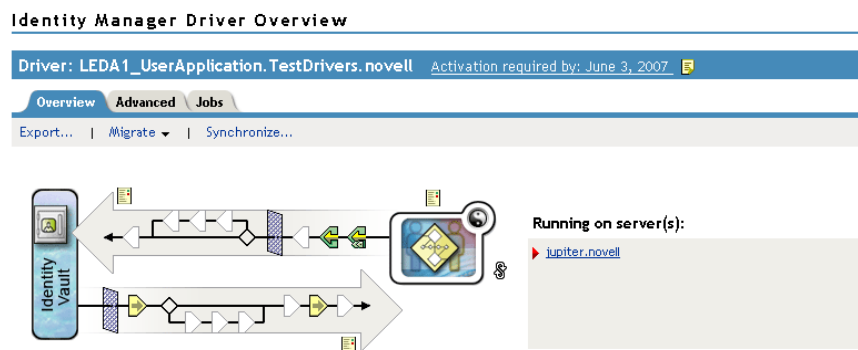
The Identity Manager Overview page displays. This page prompts you to select a driver set.

- 2 Click **Search Entire Tree**; then click **Search**. The Identity Manager Overview page displays, with a graphic that depicts the drivers in the currently selected driver set.
- 3 Click the large driver icon for the User Application driver:



UserApplication

The Identity Manager Driver Overview displays:



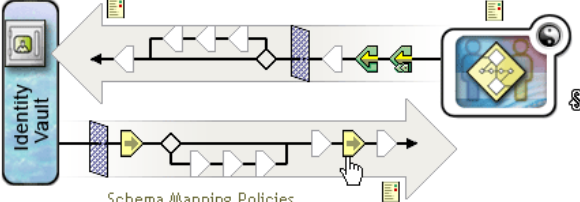
The top horizontal arrow represents the Publisher channel (which is not used in the User Application driver) and the bottom horizontal arrow represents the Subscriber channel. As you pass the mouse pointer over an object in the graphic, a description of the object displays:

Identity Manager Driver Overview

Driver: LEDA1_UserApplication.TestDrivers.novell Activation required by: June 3, 2007

Overview **Advanced** Jobs

Export... | Migrate ▾ | Synchronize...

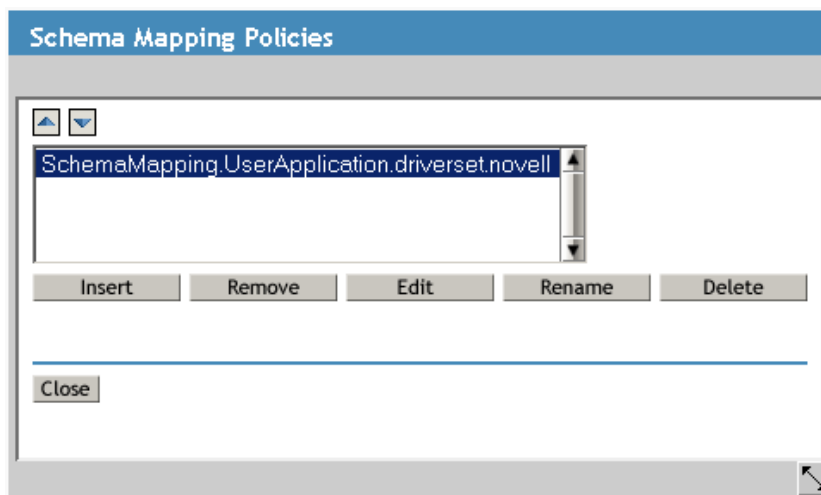


Running on server(s):

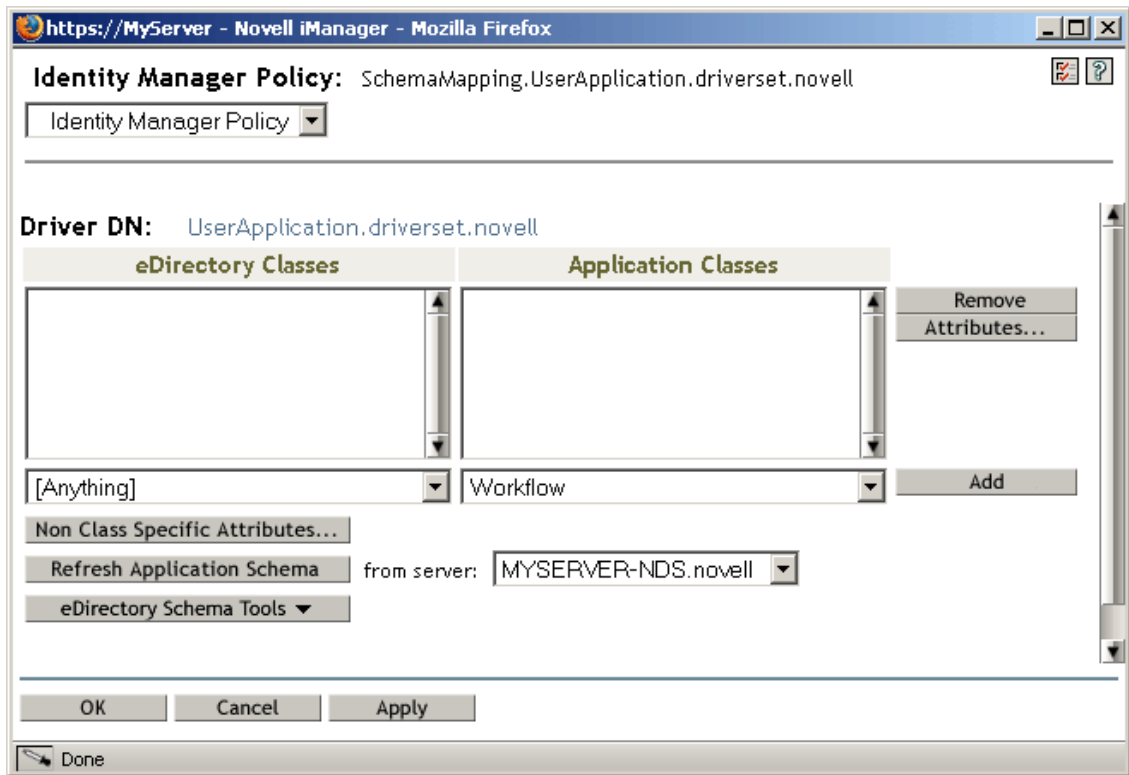
- jupiter.novell

The diagram shows a flow from Identity Vault to Schema Mapping Policies, which then connects to a server icon. A hand cursor is pointing at the Schema Mapping Policies icon.

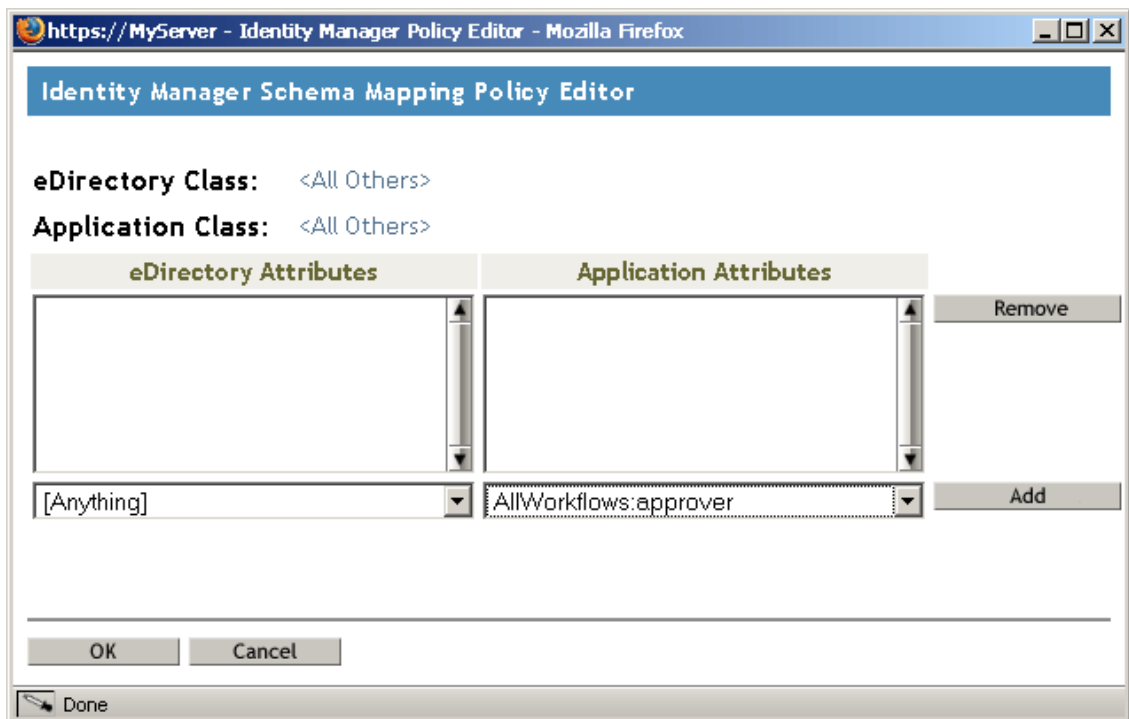
- 4 Click the **Schema Mapping Policies** icon. The **Schema Mapping Policies** dialog box displays:



- 5 Click **Edit**. The Identity Manager Policy dialog box displays. (This dialog box maps Identity Vault classes to application classes, but this procedure uses it to map eDirectory attributes to global User Application attributes.)



- 6 Click **Refresh Application Schema**. A message displays informing you that the driver must be stopped in order to read the schema, then restarted. It might take about 60 seconds to refresh the schema. This step reads the latest set of workflow information in preparation for the following step, which specifies the information to move from the Identity Vault to the workflow that will be started.
- 7 Click **OK** to refresh the schema. A message displays when the schema refresh is completed.
- 8 Click **OK** to close the schema refresh message. You are returned to the Identity Manager Policy dialog box.
- 9 Click **Non Class Specific Attributes**. The Identity Manager Schema Mapping Policy Editor displays.



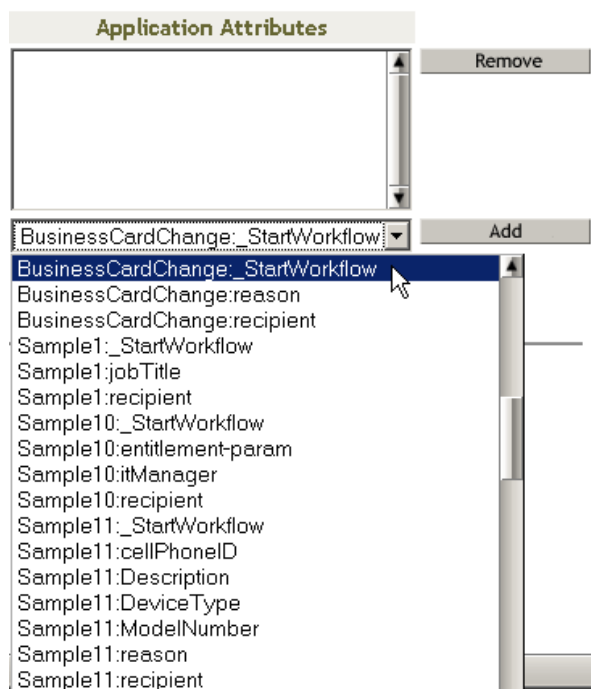
The **eDirectory Attributes** drop-down list contains all eDirectory attributes.

The **Application Attributes** drop-down list contains the attributes in all active Workflows. Attributes in the list are prefaced with either `AllWorkflows` (meaning that the attribute applies to all workflows) or the name of a specific workflow. If you want the same eDirectory attribute (for example `manager`) to be mapped to the `manager` attribute for all workflows, map `manager` to `Allworkflows:manager`. If you want a different eDirectory attribute (for example, `HRmanager`) to be used for a specific workflow, map the eDirectory attribute to the specific workflow attribute (for example `BusinessCardChange:manager`).

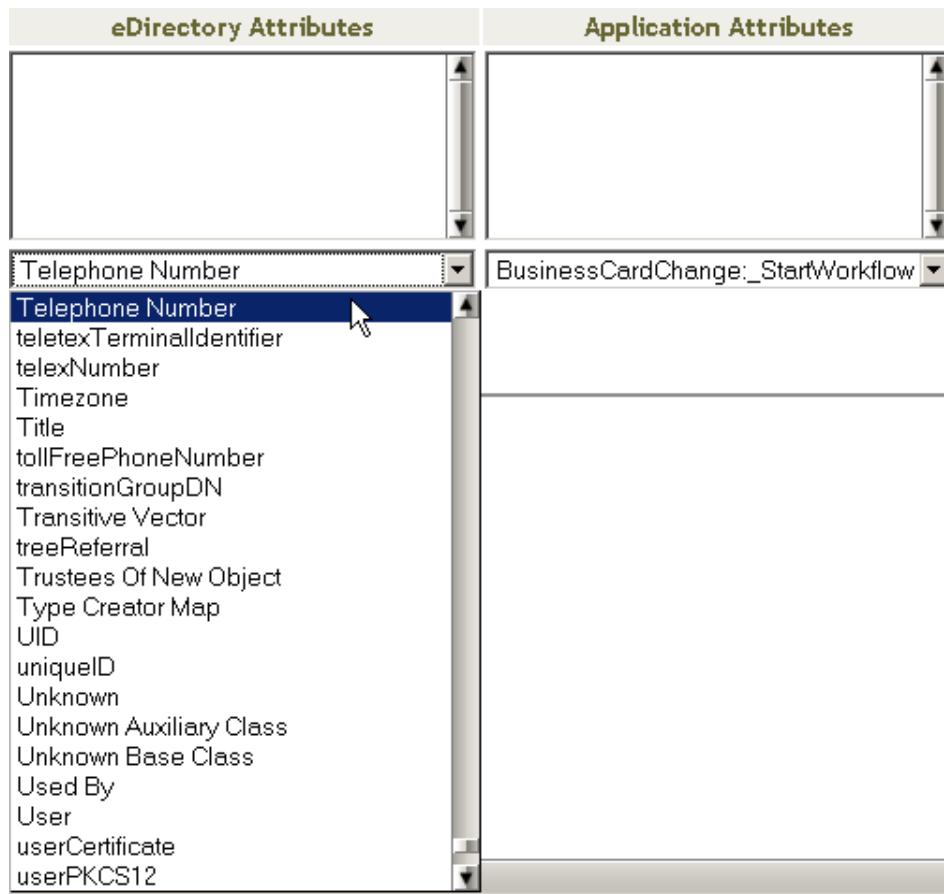
Attributes that have been mapped are displayed side-by-side in the **eDirectory Attributes** and **Application Attributes** columns.

In the following steps, map the eDirectory attribute that you want to use to start the workflow to the `_StartWorkflow` attribute for that workflow. If additional eDirectory attributes are expected by the workflow, you should also map those attributes. For example, if an eDirectory `Address` attribute is the trigger for a workflow, the workflow can also require attributes like `City` and `State`. Alternatively, these attributes can be mapped in policies.

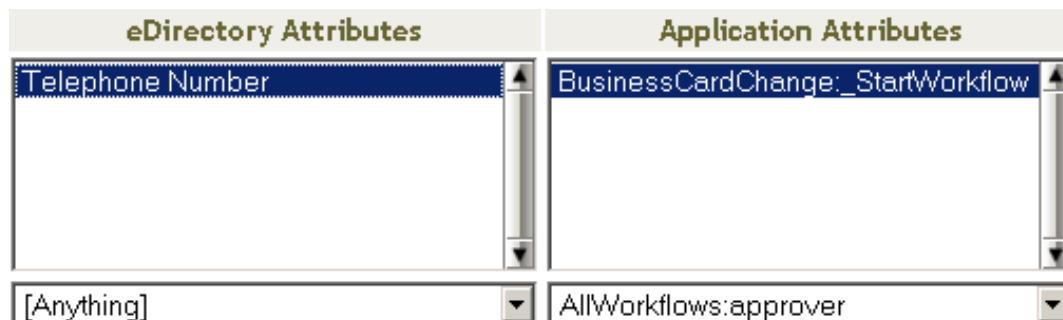
- 10 In the **Application Attributes** list, select the `_StartWorkflow` attribute for the workflow that you want to configure. The following example shows the `_StartWorkflow` attribute for a `BusinessCardChange` workflow (`BusinessCardChange_StartWorkflow`).



- 11 In the **eDirectory Attributes** list, select the eDirectory attribute that you want to use to start the workflow when that attribute changes. In the following example, the Telephone attribute is selected. This means that the BusinessCardChange workflow starts whenever an employee's telephone number changes.



- Click **Add**. The eDirectory attribute is mapped to the Application attribute.



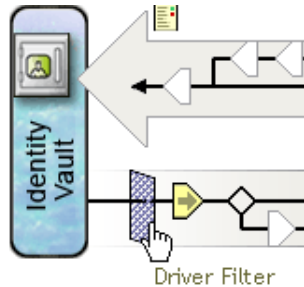
- Repeat [Step 10](#) through [Step 12](#) to map eDirectory attributes to the workflow `_reason` and `_recipient` attributes.

- If additional eDirectory attributes are needed by the workflow, repeat [Step 10](#) through [Step 12](#) until you have mapped all of the attributes that you need to map.

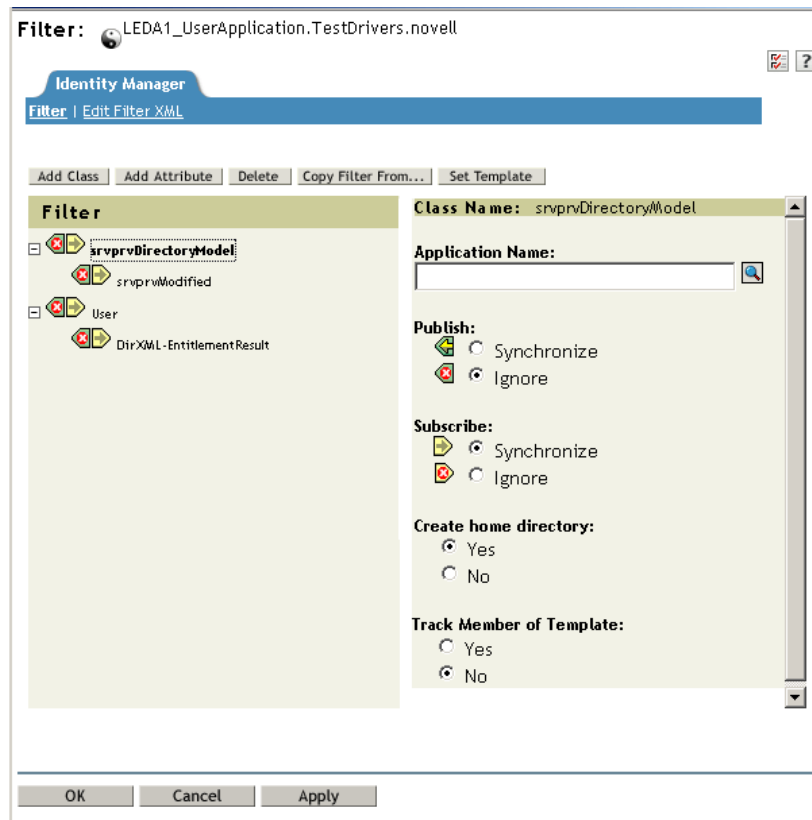
The workflow starts automatically when a change occurs in the eDirectory attribute that is mapped to an application `_StartApprovalFlow` attribute. However, the eDirectory attribute only reaches the Schema Mapping policy if the eDirectory attribute is included in the Driver Filter. In the following steps, add the eDirectory attribute to the Driver Filter.

- Click **OK** to close the Schema Mapping Policy Editor.
- Click **OK** to close the Identity Manager Policy dialog box.
- Click **Close** to close the Schema Mapping Policies dialog box.

- 18 Click the **Driver Filter** icon.



The filter window displays:

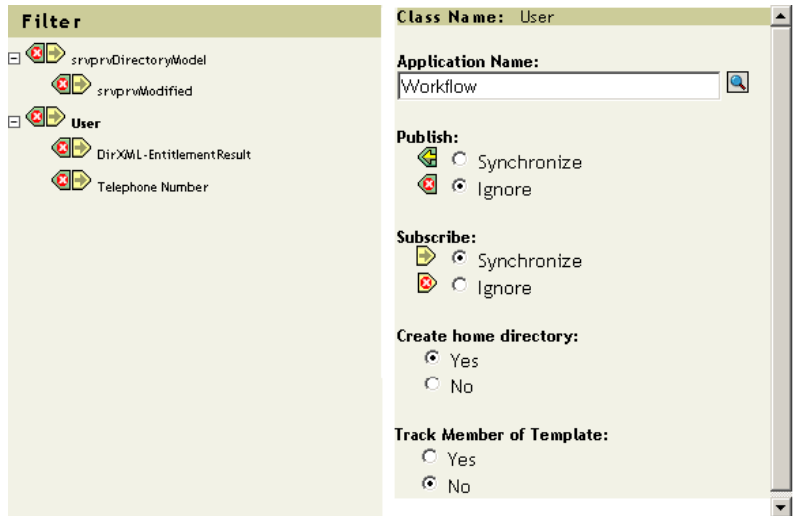


Event filters specify the object classes and the attributes for which the Identity Manager engine processes events. The read-only **Filter** list on the left shows the attributes of the class. The **Class Name** list on the right displays options associated with the target object.

- 19 Click the name of the class to which the attribute that you want to add to the filter belongs (for example, **User**).
- 20 Click **Add Attribute**. A list of attributes displays.
- 21 Select an attribute, then click **OK**. The attribute is added to the **Filter** list.



- 22 Click the attribute name. The synchronization options for the attribute are displayed on the panel on the right.



- 23 Under **Subscribe**, click **Synchronize**.



- 24 Specify any other attributes for the filter. Select **Synchronize** for an attribute if you want changes to attribute values to be reported and synchronized. Select **Ignore** if you do not want changes to attribute values to be reported and synchronized.
- 25 Click **OK**. A message displays asking you if you would like the driver to be restarted to put the changes into effect.
- 26 Click **OK**. You are returned to the Identity Manager Driver Overview page.

9 Managing Provisioning Request Definitions

This section provides instructions for managing provisioning request definitions.

9.1 About the Provisioning Request Configuration Plug-in

You can use the Provisioning Request Configuration plug-in to iManager to view a read-only display of a provisioning request definition that was created in the Designer for Identity Manager. This plug-in allows you to delete, activate, inactivate and retire existing provisioning request definitions.

NOTE: The Provisioning Request Configuration plug-in to iManager does not allow you to create or edit provisioning request definitions. To create or edit a provisioning request definition, you need to use the Designer for Identity Manager.

You can find the Provisioning Request Configuration plug-in in the Identity Manager category in iManager. The plug-in includes the Provisioning Requests task in the Provisioning Configuration role. The Provisioning Requests task consists of the panels described in [Table 9-1](#).

Table 9-1 Provisioning Requests Task: Panels

Panel	Description
Provisioning Driver Selection	Gives you the opportunity to select an Identity Manager User Application driver. The driver contains a set of predeployed provisioning request definitions, so you need to pick a driver before you can begin configuring your provisioning requests.
Provisioning Request Configuration	Provides tools that let you: <ul style="list-style-type: none">◆ Browse the available provisioning request definitions and select one to configure◆ Create a new provisioning request definition based on an existing definition◆ Set the properties of a provisioning request definition◆ Assign the provisioning request definition to a provisioned resource◆ Edit the addressee and timeout settings for each activity in the associated workflow <p>When you choose to create a new provisioning request or edit an existing one, the plug-in runs the Provisioning Request Configuration Wizard.</p>

9.2 Working with the Installed Templates

You can define provisioning request definitions from scratch in the Designer for Identity Manager. Alternatively, you can define provisioning requests by modeling them after the provisioning request templates that ship with the product. To use the templates, you define new objects based on the installed templates and customize these objects to suit the needs of your organization.

The installed templates let you determine the number of approval steps required for the request to be fulfilled. You can configure a provisioning request to require:

- ◆ No approvals
- ◆ One approval step
- ◆ Two approval steps
- ◆ Three approval steps
- ◆ Four approval steps
- ◆ Five approval steps

You can also specify whether you want to support sequential or parallel processing, and whether you want to approve or deny the request in the event that the workflow times out during the course of processing.

Identity Manager ships with the templates listed in [Table 9-2](#).

Table 9-2 *Templates for Provisioning Requests*

Template	Description
Self Provision Approval	Allows a provisioning request to be fulfilled without any approvals.
One Step Approval (Timeout Approves)	Requires a single approval for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item forwards to the next activity.
Two Step Sequential Approval (Timeout Approves)	Requires two approvals for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item forwards to the next activity. This template supports sequential processing.
Three Step Sequential Approval (Timeout Approves)	Requires three approvals for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item forwards to the next activity. This template supports sequential processing.
Four Step Sequential Approval (Timeout Approves)	Requires four approvals for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item forwards to the next activity. This template supports sequential processing.

Template	Description
Five Step Sequential Approval (Timeout Approves)	<p>Requires five approvals for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item forwards to the next activity.</p> <p>This template supports sequential processing.</p>
One Step Approval (Timeout Denies)	<p>Requires a single approval for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request.</p> <p>This template supports sequential processing.</p>
Two Step Sequential Approval (Timeout Denies)	<p>Requires two approvals for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request.</p> <p>This template supports sequential processing.</p>
Three Step Sequential Approval (Timeout Denies)	<p>Requires three approvals for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request.</p> <p>This template supports sequential processing.</p>
Four Step Sequential Approval (Timeout Denies)	<p>Requires four approvals for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request.</p> <p>This template supports sequential processing.</p>
Five Step Sequential Approval (Timeout Denies)	<p>Requires five approvals for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request.</p> <p>This template supports sequential processing.</p>
Two Step Parallel Approval (Timeout Approves)	<p>Requires two approvals for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item forwards to the next activity.</p> <p>This template supports parallel processing.</p>
Three Step Parallel Approval (Timeout Approves)	<p>Requires three approvals for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item forwards to the next activity.</p> <p>This template supports parallel processing.</p>
Four Step Parallel Approval (Timeout Approves)	<p>Requires four approvals for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item forwards to the next activity.</p> <p>This template supports parallel processing.</p>

Template	Description
Five Step Parallel Approval (Timeout Approves)	Requires five approvals for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item forwards to the next activity. This template supports parallel processing.
Two Step Parallel Approval (Timeout Denies)	Requires two approvals for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request. This template supports parallel processing.
Three Step Parallel Approval (Timeout Denies)	Requires three approvals for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request. This template supports parallel processing.
Four Step Parallel Approval (Timeout Denies)	Requires four approvals for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request. This template supports parallel processing.
Five Step Parallel Approval (Timeout Denies)	Requires five approvals for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request. This template supports parallel processing.

Workflows and provisioned resources. When you create a new provisioning request definition, you bind it to a provisioned resource. You can change the provisioned resource associated with the request definition, but not the workflow or its topology.

Categories for provisioning requests. Each provisioning request template is also bound to a category. Categories provide a convenient way to organize provisioning requests for the end user. The default category for all provisioning request templates is **Entitlements**. The category key, which is the value of the `srvprvCategoryKey` attribute, is **entitlements** (lowercase).

You can create your own categories by using the directory abstraction layer editor. When you create a new category, make sure the category key (the value of `srvprvCategoryKey`) is lowercase. This is necessary to ensure that categories work properly in the Identity Manager User Application.

For details on creating provisioning categories, see the *Identity Manager User Application: Design Guide*.

9.3 Configuring a Provisioning Request Definition

Before configuring a provisioning request definition, you need to select the Identity Manager User Application driver that contains the definition. Having selected the driver, you can create a new provisioning request definition or edit an existing definition. You can also delete provisioning request definitions, change the status of a request definition, or define rights for a request definition.

9.3.1 Selecting the Driver

To select an Identity Manager User Application driver:

- 1 Select the **Identity Manager** category in iManager.
- 2 Open the **Provisioning Request Configuration** role.
- 3 Click the **Provisioning Requests** task.

iManager displays the User Application Driver panel.

- 4 Specify the driver name in the **User Application Driver** field, then click **OK**.

iManager displays the Provisioning Request Configuration panel. The Provisioning Request Configuration panel displays a list of available provisioning request definitions.

The installed templates appear in dark text with a status of **Template**. Request definitions that are templates do not display hypertext links because they are read only.

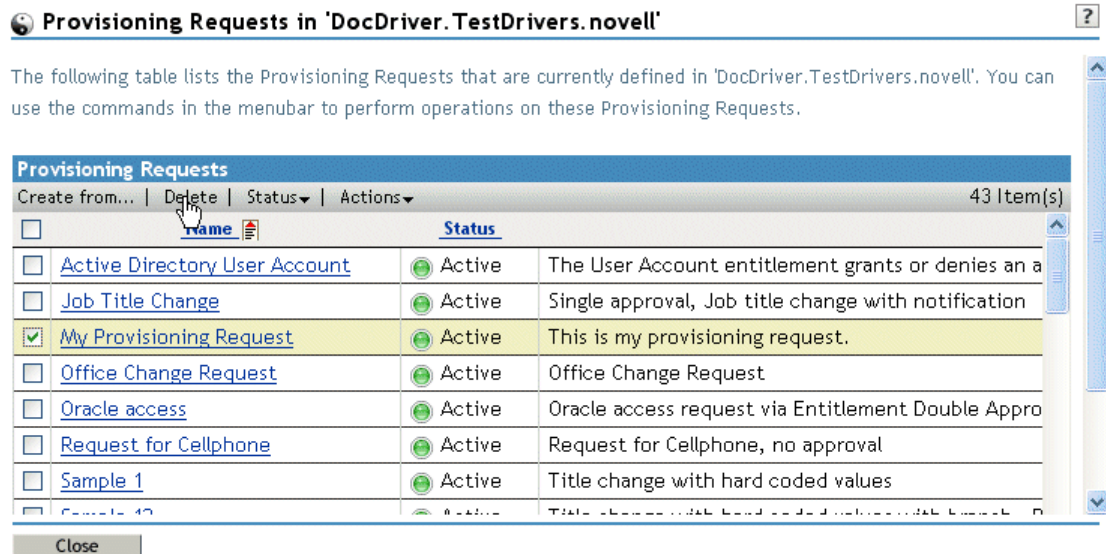
NOTE: If the request definitions were configured to use localized text, the names and descriptions for these definitions show text that is suitable for the current locale.

Changing the driver. When you have selected a driver, the driver selection remains in effect for the duration of your iManager session, unless you select a new driver. To select a new driver, click the **Actions** command, then choose **Select User Application Driver** from the **Actions** menu.

9.3.2 Deleting a Provisioning Request

To delete a provisioning request:

- 1 Select the provisioning request you want to delete by clicking the check box next to the name. You are not permitted to delete a provisioning request that is a template.
- 2 Click the **Delete** command in the Provisioning Request Configuration panel.



Provisioning Requests in 'DocDriver.TestDrivers.novell'

The following table lists the Provisioning Requests that are currently defined in 'DocDriver.TestDrivers.novell'. You can use the commands in the menubar to perform operations on these Provisioning Requests.

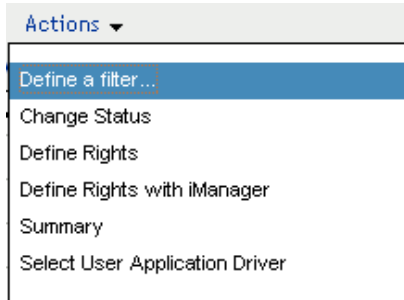
<input type="checkbox"/>	Name	Status	
<input type="checkbox"/>	Active Directory User Account	Active	The User Account entitlement grants or denies an a
<input type="checkbox"/>	Job Title Change	Active	Single approval, Job title change with notification
<input checked="" type="checkbox"/>	My Provisioning Request	Active	This is my provisioning request.
<input type="checkbox"/>	Office Change Request	Active	Office Change Request
<input type="checkbox"/>	Oracle access	Active	Oracle access request via Entitlement Double Appro
<input type="checkbox"/>	Request for Cellphone	Active	Request for Cellphone, no approval
<input type="checkbox"/>	Sample 1	Active	Title change with hard coded values
<input type="checkbox"/>	Sample 12	Active	Title change with hard coded values with hard c

Close

9.3.3 Filtering the List of Requests

To filter the list of requests:

- 1 Click the **Actions** command in the Provisioning Request Configuration panel.
- 2 Click the **Define a Filter** command on the **Actions** menu.



Specify the filter characteristics:

Choice	Description
Turn off filtering	Disables any existing filtering for the list.
Filter for status equals	Filters based on the status. You can filter the list based on any of the following status codes: Active Inactive Template Retired
Filter for category equals	Filters based on category. Select any of the defined categories.
Filter for description contains	Allows you to search for text in the request description. Type the string you want to search for.

9.3.4 Changing the Status of an Existing Provisioning Request

To change the status of an existing provisioning request:

- 1 Select the provisioning request for which you want to change status by clicking the check box beside the name.
- 2 Click the **Actions** command in the Provisioning Request Configuration panel.
- 3 Click the **Change Status** command on the **Actions** menu.

Provisioning Requests in 'DocDriver.TestDrivers.novell'



The following table lists the Provisioning Requests that are currently defined in 'DocDriver.TestDrivers.novell'. You can use the commands in the menubar to perform operations on these Provisioning Requests.

Name	Status	Description
<input type="checkbox"/> Active Directory User Account		Account entitlement grants or denies an account.
<input type="checkbox"/> Job Title Change		Job title change with notification.
<input checked="" type="checkbox"/> My Provisioning Request		Provisioning request.
<input type="checkbox"/> Office Change Request	Active	Office Change Request
<input type="checkbox"/> Oracle access	Active	Oracle access request via Entitlement Double Approval.
<input type="checkbox"/> Request for Cellphone	Active	Request for Cellphone, no approval.
<input type="checkbox"/> Sample 1	Active	Title change with hard coded values.

- 4 Click the status in the **Status** menu:

Status	Description
Active	Available for use.
Inactive	Temporarily unavailable for use.
Retired	Permanently disabled.

- 5 Click the button for the correct action (**Grant** or **Revoke**).
- 6 Click **Finish**.

9.3.5 Defining Rights on an Existing Provisioning Request

To define rights on an existing provisioning request:

- 1 Select the provisioning request for which you want to define rights by clicking the check box beside the name.
- 2 Click the **Actions** command in the Provisioning Request Configuration panel.
- 3 Click the **Define Rights** command on the **Actions** menu.

Provisioning Requests in 'DocDriver.TestDrivers.novell'



The following table lists the Provisioning Requests that are currently defined in 'DocDriver.TestDrivers.novell'. You can use the commands in the menubar to perform operations on these Provisioning Requests.

Provisioning Requests					
Create from...	Delete	Refresh	Status	Actions	26 Item(s)
<input type="checkbox"/>	Name		Status		Description
<input type="checkbox"/>	My Provisioning Request		Active	Change Status Define Rights Define Rights with iManager Select User Application Driver	Provisioning Request
<input type="checkbox"/>	Sample 1		Active		Assignment with hard coded values
<input type="checkbox"/>	Sample 10		Active		Assignment
<input type="checkbox"/>	Sample 2		Active		Title Change with IDVault and XPATH examples
<input type="checkbox"/>	Sample 3		Inactive	Software	Assignment of a laptop computer - Related Item
<input type="checkbox"/>	Sample 4		Active	Software	Entitlement with hard coded values
<input type="checkbox"/>	Sample 5		Active	Software	Entitlement with timer example
<input type="checkbox"/>	Sample 6		Active	Software	Title Change
<input type="checkbox"/>	Sample 7		Active	Software	Single approval of title change with many data
<input type="checkbox"/>	Sample 8		Active	Category1	Eight sample
<input type="checkbox"/>	Sample 9		Active	Human Resources	Nine sample
<input type="checkbox"/>	tdbtest		Active	Software	this is a test
<input type="checkbox"/>	tdbtestOneStepApproval		Active	Entitlements	TDB One Step Approval (Timeout Approves)
<input type="checkbox"/>	Template2ParallelApproval_TA		Template	Entitlements	Two Step Parallel Approval (Timeout Approves)
<input type="checkbox"/>	Template2ParallelApproval_TD		Template	Entitlements	Two Step Parallel Approval (Timeout Deny)
<input type="checkbox"/>	Template2SerialApproval_TA		Template	Entitlements	Two Step Sequential Approval (Timeout Appro

Close

4 Specify the rights for the request.

To define rights on a provisioning request with iManager:

- 1 Select the provisioning request for which you want to define rights by clicking the check box beside the name.
- 2 Click the **Actions** command in the Provisioning Request Configuration panel.
- 3 Click the **Define Rights with iManager** command on the **Actions** menu.

10 Managing Provisioning Workflows

This section provides instructions for managing provisioning workflows at runtime. It also provides instructions for configuring email notification for provisioning workflows.

10.1 About the Workflow Administration Plug-in

The Workflow Administration plug-in to iManager provides a browser-based interface that lets you view the status of workflow processes, reassign activities within a workflow, or terminate a workflow in the event that it is stopped and cannot be restarted.

You can find the Workflow Administration plug-in in the Identity Manager category in iManager. The plug-in includes the **Workflows** task in the **Workflow Administration** role.

The Workflow Administration role also includes the **Email Templates** and **Email Server Options** tasks. These tasks are shortcuts to other tasks listed under the Passwords role.

The Workflows task comprises the panels listed in [Table 10-1](#).

Table 10-1 Workflows Task: Panels

Panel	Description
Workflows	<p>Provides the primary user interface for administering provisioning workflows. The interface lists workflows currently being processed and lets you perform various actions on these workflows.</p> <p>When you first start the Workflows task, the Workflows panel requires that you select an Identity Manager User Application driver. The driver points to a workflow server. You need to select a driver before you can log in to the server and begin workflow administration.</p> <p>When you have selected a driver, you can specify search criteria for selecting the workflows to manage.</p>
Workflow Detail	<p>Provides a read-only user interface for viewing the details about a specific workflow.</p>

10.2 Managing Workflows

This section includes procedures for managing provisioning workflows using the Workflow Administration plug-in.

10.2.1 Connecting to a Workflow Server

Before you can begin managing workflows, you need to connect to a workflow server. If the User Application driver is bound to a single workflow server, you can simply specify the name of the driver to use. If the driver is associated with multiple workflow servers, you need to select the target workflow server.

To connect to a workflow server:

- 1 Select the Identity Manager category in iManager.
- 2 Open the **Workflow Administration** role.
- 3 Click the **Workflows** task.
iManager displays the Workflows panel.
- 4 If you accessed the target workflow server previously, you can select the server from the **Previously accessed servers** drop-down list.
iManager fills in the remaining fields on the panel.
- 5 If you have not yet accessed a workflow server, specify the driver name in the **User Application Driver** field, then click **OK**.
iManager fills in the **Workflow server URI** and **User** fields.
- 6 Type the password for the user in the **Password** field.
- 7 Click *Login*.

The Workflow Administration plug-in displays a page that allows you to specify a filter for finding workflows:

10.2.2 Restricting Access to Workflows

When you install the Roles Based Provisioning Module with the default settings, all users can view all workflows when searching for them. To restrict access to workflows to only the workflow trustees, use NetIQ eDirectory to add an Inheritance Rights Filter (IRF) on the `AppConfig` container, which is located under the User Application driver. For information about adding an IRF, see the [eDirectory Administration Guide](#).

10.2.3 Finding Workflows that Match Search Criteria

If the target workflow server is running a large number of workflow processes, you might want to filter the list of workflows you see in iManager. To do this, you can specify search criteria.

- 1 Select **Show Workflows with**.
By default, **Show all Workflows** is selected. Do not change the default if you want to see the complete list of workflows on the server.
- 2 Select the attribute for which you want to specify criteria.

Attribute	Description
Creation time	Time that the workflow was initiated.
Initiator	Username of the requestor.
Recipient	Username of the recipient.
Process Status	Status of the workflow process as a whole (Completed, Running, or Terminated).
Approval status	Status of the approval process (Approved, Denied, or Retracted).
Entitlement status	Status of the entitlement initiated by the provisioning request (Error, Fatal, Success, Unknown, or Warning).

- 3 Select an operator:

Operator	Comment
Equals	Supported for all attributes.
Before	Only supported for the Creation time attribute.
After	Only supported for the Creation time attribute.
Between	Only supported for the Creation time attribute.

4 Specify a value in the field below the attribute and operator.

For **Creation time**, you can use the **Date and time** control to select the value. For **Initiator** and **Recipient**, you can use **Object History** or **Object Selector** to specify a value. For all other attributes, select the value from the drop-down list.

5 Click **OK**.

iManager displays the workflows you have selected on the Workflows panel.

Changing the target server and filter. When you have selected a workflow server, this selection remains in effect for the duration of your iManager session, unless you select a new server. To select a new server, click the **Actions** command, then choose **Select Server** from the **Actions** menu.

To specify different search criteria, choose *Define Filter* on the **Actions** menu.

10.2.4 Controlling the Active Workflows Display

The Workflows panel lists the workflows that match the search criteria you specified. In addition to filtering the list, you can control the display. For example, you can specify how often to refresh the list and sort the list on a particular column.

Refreshing the List of Workflows

When the workflow server is very busy, the list of active workflows can change very frequently. In this case, you should refresh the list of active workflows running on the server.

1 Click the **Refresh** command in the Workflows panel.

2 Specify the refresh interval you want to use by selecting one of these options from the **Refresh** menu:

- ◆ Refresh Off
- ◆ Refresh Now
- ◆ 10 seconds
- ◆ 30 seconds
- ◆ 60 seconds
- ◆ 5 minutes

3 Click **OK**.

Using Quick Filters to Control the Display

Sometimes you might want to show or hide workflows that have a particular status.

- 1 Click the **Quick Filters** command in the Workflows panel.
- 2 Select one of the following choices to filter the items in the list:

Choice	Description
Show all workflows	Disables all previous filters and displays all workflows in process.
Hide/show completed workflows	Hides or shows workflows that have completed processing.
Hide/show terminated workflows	Hides or shows workflows that have been terminated.
Hide/show stopped workflows	Hides or shows workflows that have been stopped by user action.
Hide/show running workflows	Hides or shows workflows that are still running.

Sorting the List of Workflows

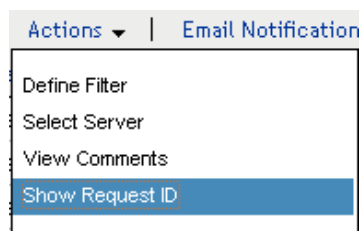
If you have a large number of request definitions, you might want to sort the list by a particular column, such as **Name** or **Description**.

- 1 Click the heading for the sort column.

Displaying the Process Request ID

You can display and sort data based on the internal process ID for a request.

- 1 Click the **Actions** command in the Workflows panel.
- 2 Click **Show Request ID** on the **Actions** menu.



Depending on your display, you might need to scroll to the right to see the **Request ID** column.

To sort the data based on the process request ID, click the heading for the **Request ID** column.

10.2.5 Terminating a Workflow Instance

If you do not want a workflow instance to continue its processing, you can terminate the workflow.

- 1 Select the workflow in the Workflows panel by clicking the check box next to the workflow name.
- 2 Click the **Terminate** command in the Workflows panel.

10.2.6 Viewing Details about a Workflow Instance

When you have displayed a set of running workflows on a particular server, you can select a workflow instance to see more details about the running process.

NOTE: If a workflow instance uses a serial processing design pattern, the display shows a single activity as current because only one user can act on the work item at any point in time. However, if the workflow handles parallel processing and branching, there might be multiple current activities for a workflow instance.

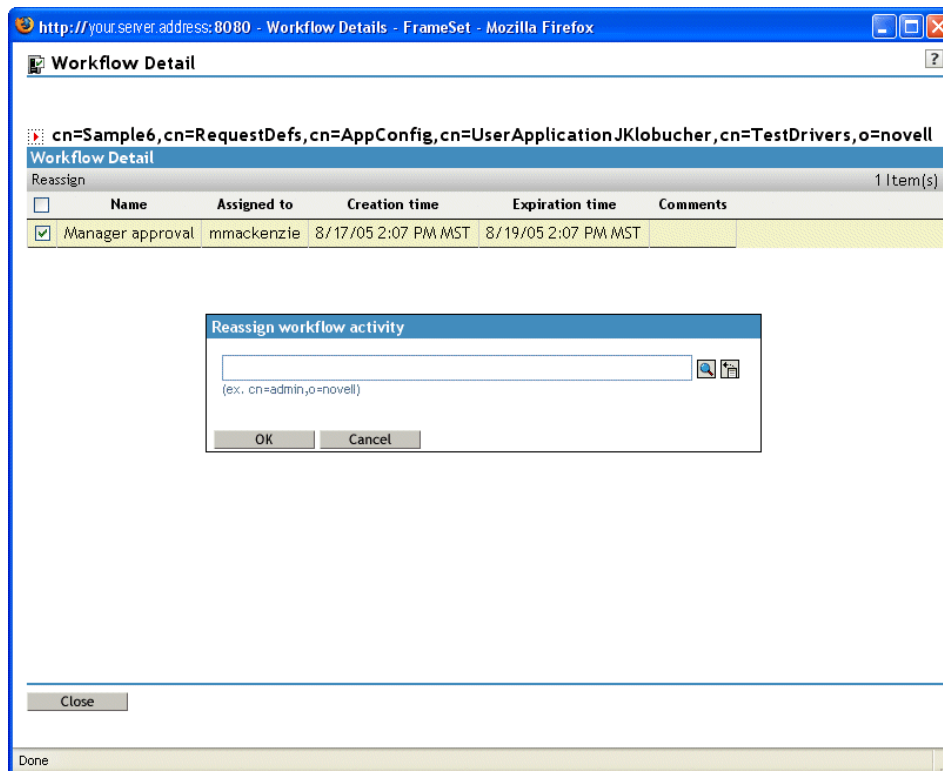
To view details about a particular workflow instance:

- 1 Click the name of the workflow instance in the Workflows panel.
iManager displays the Workflow Detail panel.

10.2.7 Reassigning a Workflow Instance

If a workflow instance has stopped and cannot be restarted, you can reassign the work item to another user or group.

- 1 Select the current activity associated with the workflow by clicking the check box next to the name in the Workflow Detail panel.
- 2 Click the **Reassign** command in the Workflow Detail panel.



- 3 Select the user or group to which you want to reassign the work item.

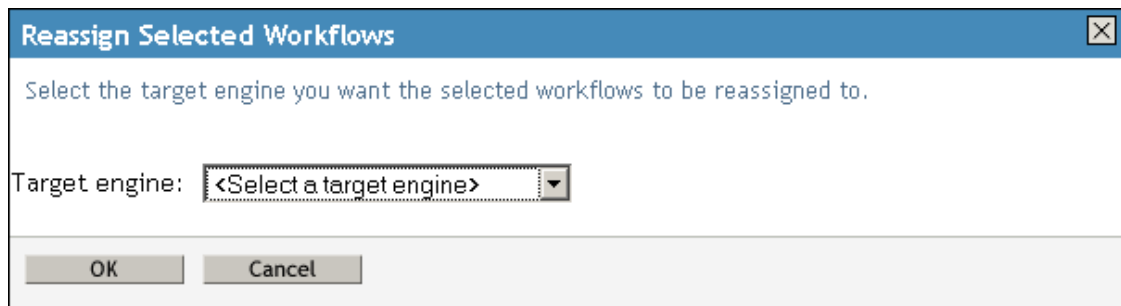
10.2.8 Managing Workflow Processes in a Cluster

You can use the Workflows screen to reassign processes from one workflow engine to another. For example, you could use this feature to reassign processes back to a failed workflow engine when the workflow engine is brought back online, or you could redistribute processes to other engines when an engine is permanently removed from the cluster.

The source engine(s) must be in a SHUTDOWN or TIMEDOUT state. The target engine must be restarted in order to restart the processes that were reassigned to that engine.

Reassigning a Process from One Workflow Engine to Another

- 1 In the Workflows panel, select the workflow that you would like to reassign by clicking the check box next to the workflow name.
- 2 Select **Actions > Reassign**.

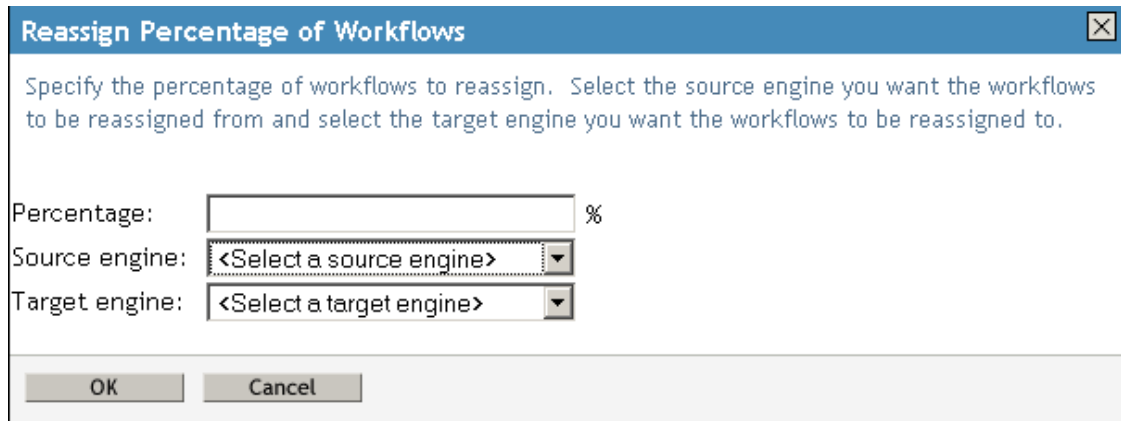


The dialog box titled "Reassign Selected Workflows" has a blue header bar with a close button (X) in the top right corner. Below the header, the text reads: "Select the target engine you want the selected workflows to be reassigned to." There is a label "Target engine:" followed by a dropdown menu containing the text "<Select a target engine>". At the bottom of the dialog, there are two buttons: "OK" and "Cancel".

- 3 Select the workflow engine to which you want to reassign the workflow process from the **Target Engine** list.
- 4 Click **OK**.

Reassigning a Percentage of Processes from One Workflow Engine to Another

- 1 In the Workflows panel, select the workflow that you would like to reassign by clicking the check box next to the workflow name.
- 2 Select **Actions > Reassign Percentage**.



The dialog box titled "Reassign Percentage of Workflows" has a blue header bar with a close button (X) in the top right corner. Below the header, the text reads: "Specify the percentage of workflows to reassign. Select the source engine you want the workflows to be reassigned from and select the target engine you want the workflows to be reassigned to." There are three input fields: "Percentage:" followed by a text box and a "%" symbol; "Source engine:" followed by a dropdown menu containing the text "<Select a source engine>"; and "Target engine:" followed by a dropdown menu containing the text "<Select a target engine>". At the bottom of the dialog, there are two buttons: "OK" and "Cancel".

- 3 In the **Percentage** field, type the percentage of workflow processes that you would like to reassign from one workflow engine to another.
- 4 Use the **Source engine** list to select the workflow engine from which you want to reassign processes.
- 5 Use the **Target engine** field to select the workflow engine to which you want to reassign processes.
- 6 Click **OK**.

Reassigning All Processes from One Workflow Engine to Another

- 1 In the Workflows panel, select the workflow that you would like to reassign by clicking the check box next to the workflow name.
- 2 Select **Actions > Reassign All**.

Reassign All Workflows

Select the source engine you want the workflows to be reassigned from. Select the target engines you want the workflows to be reassigned to. If multiple target engines are selected, the workflows from the source engine will be spread across the target engines.

Source engine:

Target engines:

- ENGINE
- gw-Engine1
- gw-Engine2

OK Cancel

- 3 Use the **Source engine** list to select the workflow engine from which you want to reassign processes.
- 4 Select the workflow engines to which you would like to reassign processes by clicking the check box next to the name of the workflow engine.

If you select multiple target engines, the processes from the source engine will be evenly distributed to the target engine.
- 5 Click **OK**.

10.3 Configuring the Email Server

A workflow process often sends email notifications at various points in the course of its execution. For example, an email might be sent when a user assigns a workflow activity to a new addressee.

Before you can take advantage of the email notification capabilities of Identity Manager, you need to configure the SMTP email server. To do this, you need to use the **Email Server Options** task within the Workflow Administration role in iManager.

NOTE: This task is a shortcut to the **Email Server Options** task under the Passwords role.

To configure the email server:

- 1 Select the Identity Manager category in iManager.
- 2 Open the **Workflow Administration** role.
- 3 Click on the **Email Server Options** task.
iManager displays the Email Server Options panel.
- 4 Type the name (or IP address) of the host server in the **Host Name** field.
- 5 Type the email address for the sender in the **From** field.

When the recipient opens the email, this text is displayed in the **From** field of the email header. Depending on your mail server settings, the text in this field might need to match a valid sender in the system in order to allow the mail server to do reverse lookups or authentication. An example is helpdesk@company.com instead of descriptive text such as The Password Administrator.

- 6 If your server requires authentication before sending email, select the **Authenticate to server using credentials** check box and specify the username and password.
- 7 When you are finished, click **OK**.

10.4 Working with Email Templates

Identity Manager includes email notification templates that are designed specifically for workflow-based provisioning. These email templates include the following.

- ◆ **New Provisioning Request** (Provisioning Notification)
- ◆ **Availability Setting Notification** (Availability)
- ◆ **Delegate Assignment Notification** (Delegate)
- ◆ **Provisioning Approval Notification** (Provisioning Approval Completed Notification)
- ◆ **Reminder - A request is waiting on your approval** (Provisioning Reminder)
- ◆ **Proxy Assignment Notification** (Proxy)
- ◆ **New Role Request** (Role Request Notification)
- ◆ **Role Request Approval Notification** (Role Request Approval Completed Notification)
- ◆ **Compliance Task** (Attestation Notification)
- ◆ **New Resource Request** (Resource Request Notification)
- ◆ **Resource Request Approval Notification** (Resource Request Approval Completed Notification)

The subject lines are listed first above. The template names (as they appear in iManager and Designer) are given in parentheses.

You can edit the templates to change the content and format of email messages. You can also create new templates. If you create new templates, you need to follow these naming conventions.

- ◆ The language-independent version of the Provisioning Notification template can have any name you like. The default template for notification email messages is called:

Provisioning Notification

- ◆ The language-independent version of the Provisioning Reminder template can have any name you like. The default template for reminder email messages is called:

Provisioning Reminder

- ◆ Each delegation template must have a name that begins with the word:

delegate

The language-independent name can be followed by one or more characters that describe the purpose or content of the template.

- ◆ Each proxy template must have a name that begins with the word:

proxy

The language-independent name can be followed by one or more characters that describe the purpose or content of the template.

- ◆ Each availability template must have a name that begins with the word:

availability

The language-independent name can be followed by one or more characters that describe the purpose or content of the template.

Each language-specific version of a template must have a suffix that provides a language code (for example, `_fr` for French, `_es` for Spanish, and so forth).

To create or edit an email template, use the **Email Templates** task within the Workflow Administration role in iManager.

NOTE: This task is a shortcut to the **Edit Email Templates** task under the Passwords role.

You also can create and edit email templates in Designer.

When you create a User Application driver in iManager or Designer, any email notification templates that are missing from the standard set of email notification templates are replaced. Existing email notification templates are not updated. This is to prevent overwriting email notification templates that you have customized. You can update existing email notification templates manually using Designer (see the section “About Email Notification Templates” in the *Identity Manager User Application: Design Guide* (<http://www.netiq.com/documentation/idm402/index.html>)). For more information about email notification templates, see “Setting up Email Notification Templates” in the *NetIQ Designer for Identity Manager Administration Guide*.

NOTE: When you use a localized email template in a provisioning request definition, the preferred locale setting of the recipient of the notification is ignored. For example, the Provisioning Notification of a request using a localized email notification template of Spanish will only send a Spanish email, regardless of the preferred locale setting for the user.

10.4.1 Default Content and Format

This section shows you what the content of the email templates looks like after you install the product. It also describes the replacement tags that can be used in the email template.

New Provisioning Request

This template identifies the provisioning request definition that triggered the email message. In addition, it includes a URL that redirects the addressee to the task that requires approval, as well as a URL that displays the complete list of tasks pending for that user.

Hi ,

A new provisioning request has been submitted that requires your approval.

Request name: \$requestTitle\$
Submitted by: \$initiatorFullName\$
Recipient: \$recipientFullName\$

Please review the details of this request at \$PROTOCOL\$://\$HOST\$: \$PORT\$/\$TASK_DETAILS\$ to take the appropriate action.

You can review a list of all requests pending your approval at \$PROTOCOL\$://\$HOST\$: \$PORT\$/\$TASKLIST_CONTEXT\$.

Table 10-2 *New Provisioning Request Template: Replacement Tags*

Tag	Description
\$userFirstName\$	The first name of the addressee.
\$requestTitle\$	The display name of the provisioning request definition.
\$initiatorFullName\$	The full name of the initiator.
\$recipientFullName\$	The full name of the recipient.
\$PROTOCOL\$	The protocol for URLs included in the email message.
\$SECURE_PROTOCOL\$	The secure protocol for URLs included in the email message.
\$HOST\$	The host for the JBoss application server that is running the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$PORT\$	The port for the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$SECURE_PORT\$	The secure port for the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$TASKLIST_CONTEXT\$	The page that displays the list of all requests pending for the addressee.

Tag	Description
\$TASK_DETAILS\$	The page that displays details for the request for which this email message was generated.

Availability Setting Notification

This template identifies a user whose availability has been updated. It includes the start time and expiration time of the period for which the user is unavailable, and the resources for which the user is unavailable.

Hi,

\$submitterFirstName\$ \$submitterLastName\$ has updated availability settings for \$userFirstName\$ \$userLastName\$.

This user has \$operation\$ an availability setting that applies to the following resources:

\$resources\$

This setting indicates that \$userFirstName\$ \$userLastName\$ is unavailable to work on these resources during the timeframe outlined below:

Start time: \$startTime\$

Expiration time: \$expirationTime\$

When a user is unavailable, any delegates assigned may handle resource requests for that user.

You can review a list of your availability settings at \$PROTOCOL\$://\$HOST\$: \$PORT\$/\$AVAILABILITY_CONTEXT\$.

Table 10-3 Availability Setting Notification Template: Replacement Tags

Tag	Description
\$submitterFirstName\$	The first name of the user who updated the availability setting.
\$PROTOCOL\$	The protocol for URLs included in the email message.
\$PORT\$	The port for the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$startTime\$	The start time of the workflow for this provisioning request.
\$resources\$	The resources (provisioning requests) for which the addressee is unavailable.
\$SECURE_PROTOCOL\$	The secure protocol for URLs included in the email message.
\$expirationTime\$	The time at which the availability will expire.

Tag	Description
\$submitterLastName\$	The last name of the user who updated the availability setting.
\$SECURE_PORT\$	The secure port for the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$userFirstName\$	The first name of the user to whom this availability setting applies.
\$userLastName\$	The last name of the user to whom this availability setting applies.
\$HOST\$	The host for the JBoss application server that is running the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$ASSIGNMENT_LIST_CONTEXT\$	The context or path of the URL to the provisioning User Application.

Delegate Assignment Notification

This template notifies a user when a provisioning request has been submitted that requires the user’s approval. It includes the name of the request, the user who submitted the request, and the full name of the recipient. It includes links for viewing the provisioning request and for viewing all provisioning requests awaiting the user’s approval.

Hi,

A new provisioning request has been submitted that requires your approval.

Request name: \$requestTitle\$
Submitted by: \$initiatorFullName\$
Recipient: \$recipientFullName\$

Please review the details of this request at \$PROTOCOL\$://\$HOST\$: \$PORT\$/\$TASK_DETAILS\$ to take the appropriate action.

You can review a list of all requests pending your approval at \$PROTOCOL\$://\$HOST\$: \$PORT\$/\$TASKLIST_CONTEXT\$.
_SUBJECT

Table 10-4 Delegate Assignment Notification: Replacement Tags

Tag	Description
\$submitterFirstName\$	The first name of the user who assigned the delegate.
\$PROTOCOL\$	The protocol for URLs included in the email message.

Tag	Description
\$PORT\$	The port for the Identity Manager User Application For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$resources\$	The resources (provisioning requests) for which the delegate is available.
\$SECURE_PROTOCOL\$	The secure protocol for URLs included in the email message.
\$fromUsers\$	The users for which the assigned delegate is authorized to handle resource requests.
\$relationship\$	The relationship defined in the directory abstraction layer that was selected for this delegate assignment.
\$expirationTime\$	The time at which the delegate assignment will expire.
\$fromContainers\$	The containers for which the assigned delegate is authorized to handle resource requests.
\$fromGroups\$	The groups for which the assigned delegate is authorized to handle resource requests.
\$submitterLastName\$	The last name of the user who assigned the delegate.
\$SECURE_PORT\$	The secure port for the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$userFirstName\$	The first name of the user who has been assigned as a delegate.
\$userLastName\$	The last name of the user who has been assigned as a delegate.
\$HOST\$	The host for the JBoss application server that is running the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$ASSIGNMENT_LIST_CONTEXT\$	The context or path of the URL to the provisioning User Application.

Provisioning Approval Notification

This template notifies a user when an approval process for a provisioning request submitted by the user has been completed.

Hi,

The approval process of your provisioning request has completed.

Request name: \$requestTitle\$
Request id: \$requestId\$
Submitted by: \$initiatorFullName\$
Submitted on: \$requestSubmissionTime\$
Recipient: \$recipientFullName\$

Status: \$requestStatus\$

Table 10-5 Provisioning Approval Notification: Replacement Tags

Tag	Description
\$initiatorFullName\$	The full name of the initiator.
\$requestSubmissionTime\$	The time at which the request was submitted.
\$requestTitle\$	The display name of the provisioning request definition.
\$requestId	The ID of the provisioning request.
\$recipientFullName\$	The full name of the recipient.

Reminder - A Request Is Waiting on Your Approval

This template reminds a user that a provisioning request that requires the user's approval is waiting in a queue for approval. It includes the name of the request, the user who submitted the request, and the recipient. It includes links for viewing the provisioning request and for viewing all provisioning requests awaiting the user's approval.

Hi,

This is a reminder that a provisioning request is sitting in your queue waiting on your approval.

Request name: \$requestTitle\$
Submitted by: \$initiatorFullName\$
Recipient: \$recipientFullName\$

Please review the details of this request at \$PROTOCOL\$://\$HOST\$: \$PORT\$/
\$TASK_DETAILS\$ to take the appropriate action.

You can review a list of all requests pending your approval at \$PROTOCOL\$://
\$HOST\$: \$PORT\$/\$TASKLIST_CONTEXT\$.

Table 10-6 Reminder - A request is waiting on your approval: Replacement Tags

Tag	Description
\$TASKLIST_CONTEXT\$	The page that displays the list of all requests pending for the addressee.

Tag	Description
\$PROTOCOL\$	The protocol for URLs included in the email message.
\$PORT\$	The port for the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$SECURE_PROTOCOL\$	The secure protocol for URLs included in the email message.
\$initiatorFullName\$	The full name of the initiator.
\$recipientFullName\$	The full name of the recipient.
\$TASK_DETAILS\$	The page that displays details for the request for which this email message was generated.
\$SECURE_PORT\$	The secure port for the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$userFirstName\$	The first name of the addressee.
\$HOST\$	The host for the JBoss application server that is running the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$requestTitle\$	The display name of the provisioning request definition.

Proxy Assignment Notification

This template notifies the recipient that a proxy has been assigned. The user who has been assigned as a proxy is identified, as are the users, groups, and containers for which the user is authorized to act as proxy. It includes links for viewing the recipient’s list of proxy assignments.

Hi,

A proxy assignment that authorizes a user to act as proxy for one or more users, groups, or containers was \$operation\$ by: \$submitterFirstName\$ \$submitterLastName\$.

Unlike delegate assignments, proxy assignments are independent of resource requests, and therefore apply to all work and settings actions.

The user selected as proxy is:

\$userFirstName\$ \$userLastName\$

The assigned proxy is authorized to handle all work for these users, groups, and containers:

Users: \$fromUsers\$

Groups: \$fromGroups\$

Containers: \$fromContainers\$

This proxy assignment expires at:

\$expirationTime\$

You can review a list of your proxy assignments at \$PROTOCOL\$://\$HOST\$: \$PORT\$/\$PROXY_CONTEXT\$.

Table 10-7 Proxy Assignment Notification: Replacement Tags

Tag	Description
\$submitterFirstName\$	The first name of the user who assigned the proxy.
\$PROTOCOL\$	The protocol for URLs included in the email message.
\$PORT\$	The port for the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, "Modifying Default Values for the Template," on page 229.
\$resources\$	The resources (provisioning requests) for which the proxy is available.
\$SECURE_PROTOCOL\$	The secure protocol for URLs included in the email message.
\$fromUsers\$	The users for which the assigned proxy is authorized to handle resource requests.
\$expirationTime\$	The time at which the proxy assignment will expire.
\$fromContainers\$	The containers for which the assigned proxy is authorized to handle resource requests.
\$fromGroups\$	The groups for which the assigned proxy is authorized to handle resource requests.
\$submitterLastName\$	The last name of the user who assigned the proxy.

Tag	Description
\$SECURE_PORT\$	The secure port for the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$userFirstName\$	The first name of the user who has been assigned as a proxy.
\$userLastName\$	The last name of the user who has been assigned as a proxy.
\$HOST\$	The host for the JBoss application server that is running the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$ASSIGNMENT_LIST_CONTEXT\$	The context or path of the URL to the provisioning User Application.

New Role Request

This template identifies the provisioning request definition that triggered the email message. In addition, it includes a URL that redirects the addressee to the task that requires approval, as well as a URL that displays the complete list of tasks pending for that user.

Hi,

A new role request has been submitted that requires your approval.

Request name: \$requestTitle\$
 Submitted by: \$initiatorFullName\$
 Recipient: \$recipientFullName\$

Please review the details of this role request at \$PROTOCOL\$://\$HOST\$: \$PORT\$/\$TASK_DETAILS\$ to take the appropriate action.

You can review a list of all role requests pending your approval at \$PROTOCOL\$://\$HOST\$: \$PORT\$/\$TASKLIST_CONTEXT\$.

Table 10-8 *New Role Request Template: Replacement Tags*

Tag	Description
\$userFirstName\$	The first name of the addressee.
\$requestTitle\$	The display name of the request definition.
\$initiatorFullName\$	The full name of the initiator.
\$recipientFullName\$	The full name of the recipient.
\$PROTOCOL\$	The protocol for URLs included in the email message.
\$SECURE_PROTOCOL\$	The secure protocol for URLs included in the email message.

Tag	Description
\$HOST\$	The host for the JBoss application server that is running the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$PORT\$	The port for the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$SECURE_PORT\$	The secure port for the Identity Manager User Application. For information about setting the value for this parameter, see Section 10.4.3, “Modifying Default Values for the Template,” on page 229.
\$TASKLIST_CONTEXT\$	The page that displays the list of all requests pending for the addressee.
\$TASK_DETAILS\$	The page that displays details for the request for which this email message was generated.

Role Request Approval Notification

This template notifies a user when an approval process for a role request submitted by the user has been completed.

Hi,

The approval process of your role request has completed.

Request name: \$requestTitle\$
 Request id: \$requestId\$
 Submitted by: \$initiatorFullName\$
 Submitted on: \$requestSubmissionTime\$
 Recipient: \$recipientFullName\$

Status: \$requestStatus\$

Table 10-9 Role Request Approval Notification: Replacement Tags

Tag	Description
\$initiatorFullName\$	The full name of the initiator.
\$requestSubmissionTime\$	The time at which the request was submitted.
\$requestTitle\$	The display name of the provisioning request definition.
\$requestId	The ID of the role request.
\$recipientFullName\$	The full name of the recipient.

Compliance Task

This template notifies an attester when an attestation process has assigned a task to the attester.

Hi,

A new compliance activity has been submitted that requires your attention.

Request name: \$requestTitle\$
Submitted by: \$initiatorFullName\$

Please review the details of this compliance activity request at \$PROTOCOL\$://\$HOST\$: \$PORT\$/\$TASK_DETAILS\$ to take the appropriate action.

You can review a list of all requests pending your action at \$PROTOCOL\$://\$HOST\$: \$PORT\$/\$TASKLIST_CONTEXT\$.

Table 10-10 Compliance Task: Replacement Tags

Tag	Description
\$initiatorFullName\$	The full name of the initiator.
\$requestTitle\$	The display name of the attestation request.

New Resource Request

This template identifies the resource request definition that triggered the email message. In addition, it includes a URL that redirects the addressee to the task that requires approval, as well as a URL that displays the complete list of tasks pending for that user.

Hi,

A new resource request has been submitted that requires your approval.

Request name: \$requestTitle\$
Submitted by: \$initiatorFullName\$
Recipient: \$recipientFullName\$

Please review the details of this role request at \$PROTOCOL\$://\$HOST\$: \$PORT\$/\$TASK_DETAILS\$ to take the appropriate action.

You can review a list of all resource requests pending your approval at \$PROTOCOL\$://\$HOST\$: \$PORT\$/\$TASKLIST_CONTEXT\$.

Table 10-11 New Resource Request Template: Replacement Tags

Tag	Description
<code>\$userFirstName\$</code>	The first name of the addressee.
<code>\$requestTitle\$</code>	The display name of the request definition.
<code>\$initiatorFullName\$</code>	The full name of the initiator.
<code>\$recipientFullName\$</code>	The full name of the recipient.
<code>\$PROTOCOL\$</code>	The protocol for URLs included in the email message.
<code>\$SECURE_PROTOCOL\$</code>	The secure protocol for URLs included in the email message.
<code>\$HOST\$</code>	The host for the JBoss application server that is running the Identity Manager User Application.
<code>\$PORT\$</code>	The port for the Identity Manager User Application.
<code>\$SECURE_PORT\$</code>	The secure port for the Identity Manager User Application.
<code>\$TASKLIST_CONTEXT\$</code>	The page that displays the list of all requests pending for the addressee.
<code>\$TASK_DETAILS\$</code>	The page that displays details for the request for which this email message was generated.

Resource Request Approval Notification

This template notifies a user when an approval process for a resource request submitted by the user has been completed.

Hi,

The approval process of your resource request has completed.

Request name: `$requestTitle$`
Request id: `$requestId$`
Submitted by: `$initiatorFullName$`
Submitted on: `$requestSubmissionTime$`
Recipient: `$recipientFullName$`

Status: `$requestStatus$`

Table 10-12 Role Request Approval Notification: Replacement Tags

Tag	Description
<code>\$initiatorFullName\$</code>	The full name of the initiator.
<code>\$requestSubmissionTime\$</code>	The time at which the request was submitted.
<code>\$requestTitle\$</code>	The display name of the provisioning request definition.
<code>\$requestId</code>	The ID of the role request.
<code>\$recipientFullName\$</code>	The full name of the recipient.

10.4.2 Editing Email Templates

You can change the content or format of the supplied email templates. For information about creating email templates, see “Configuring Email Notification” in the *NetIQ Identity Manager Administration Guide*.

To edit a template:

- 1 Select the **Identity Manager** category in iManager.
- 2 Open the **Workflow Administration** role.
- 3 Click the **Email Templates** task.
iManager displays the **Edit Email Templates** panel.
- 4 Click the name of the email template that you would like to edit.
iManager displays the **Modify Email Message** screen.
- 5 Make your changes in the **Message Body** box.
- 6 If necessary, copy one or more of the supplied tags in the **Replacement Tags** list to include dynamic text in the message body.
For a description of the replacement tags, see [Section 10.4.1, “Default Content and Format,” on page 218](#).
- 7 When you are finished, click **OK**.

10.4.3 Modifying Default Values for the Template

At installation time, you can set default values for several of the replacement tags used in email templates. After you have completed the installation, you can also modify these values by using the User Application Configuration tool.

- 1 Run the `configupdate.sh` script in the `idm` folder.

```
./configupdate.sh
```

On Windows, run `configupdate.bat`.

- 2 Make changes as necessary to any of the following fields:

Field	Description
Email Notify Host	Used to replace the <code>\$HOST\$</code> token in email templates used in approval flows. If left blank, computed by the server.
Email Notify Port	Used to replace the <code>\$PORT\$</code> token in email templates used in approval flows.
Email Notify Secure Port	Used to replace the <code>\$SECURE_PORT\$</code> token in email templates used in approval flows.

- 3 Click **OK** to confirm your changes.

10.4.4 Adding Localized Email Templates

To add localized email templates:

- 1 Select the **Identity Manager** category in iManager.
- 2 Open the **Workflow Administration** role.
- 3 Click the **Email Templates** task.
iManager displays the **Edit Email Templates** panel.
- 4 Identify the email template (without any locale in the name) that you want to copy.
 - 4a Write down the template name to use in [Step 5](#).
 - 4b Click the template subject to open the template and view its message subject, body, and replacement tags.
 - 4c Copy the message subject, body (to be translated), and replacement tags that you want to use in your new template.
 - 4d Click **Cancel**.
- 5 Click **Create**, then enter the template name with a locale extension. For example, to create a Forgot Hint template in German, enter the name Forgot Hint_de, where _de signifies Deutsch (German).

If you use a two-letter language and two-letter country code, this works fine. If you attempt to use a locale with a variant such as en_US_TX, only the variant and language are considered. Do not use locale variants when naming email templates.
- 6 Click **OK**.
- 7 In the template list, click the newly created template, for example Forgot Hint_de, and enter the translated subject and message body. Be sure to preserve the replacement tags surrounded by the dollar (\$) sign in the message body.
- 8 If necessary, copy one or more of the supplied tags in the **Replacement Tags** list to include dynamic text in the message body.

For a description of the replacement tags, see [Section 10.4.1, "Default Content and Format," on page 218](#).
- 9 Click **Apply**.
- 10 Click **OK**.

NOTE: Email templates only send localized content if the preferred locale is set for the user (to whom the mail is sent).

10.5 Allowing a Named Password to be Retrieved over LDAP

You can add a boolean definition to the User Application driver to allow a named password to be retrieved over LDAP from a workflow. To take advantage of this feature, you need to create a global configuration value "allow-fetch-named-passwords".

Here's a sample definition:

```

<definitions>
  <definition display-name="Allow Named Password to be retrieved over LDAP"
name="allow-fetch-named-passwords" type="boolean">
    <value>>false</value>
    <description>Allow Named Password to be retrieved over LDAP. If the
value is true, then the named password value can be fetched using the LDAP
extension
com.novell.nds.dirxml.ldap.GetNamedPasswordRequest/
com.novell.nds.dirxml.ldap.GetNamedPasswordResponse.</description>
  </definition>
</definitions>

```

If the global configuration is not present, the runtime functions as if the definition is present and the value is set to false. If you then try to use the GCV script method `getValueForNamedPassword(String valueKey)`, an exception is thrown since the permission is set to false. If you want to be able to use the method, then the value for `allow-fetch-named-passwords` variable must be true.

If the gcv variable `allow-fetch-named-passwords` does not exist, you have to create the variable and set it to true. If it already exists, you can simply need to set the value to true.

NOTE: To retrieve a named password, you must use the GCV script method `getValueForNamedPassword` on a GCV of the `password-ref` type, which points to the named password. You cannot use the `get` script method.

To add the GCV value for the `allow-fetch-named-passwords` option:

- 1 In iManager, double click on the User Application driver.
- 2 Click on the Global Configuration Values tab.
- 3 Click on the *Add* button.
- 4 Fill out the definition, as described below:
 - 4a Specify `allow-fetch-named-passwords` as the name for the global configuration definition.
 - 4b Specify `Allow Named Password to be retrieved over LDAP` as the display name.
 - 4c Provide a description for the definition.
 - 4d Specify **boolean** as the Type.
- 5 Click *OK*.
- 6 Set the value to true or false and click *Apply*.
- 7 Create a named password in your User Application driver.
- 8 Create a GCV of the type `password-ref` that points to the named password you want to be able to read.
- 9 In your workflow, use the function `getValueForNamedPassword` to retrieve the value of the named password, using the following syntax:

```
GCV.getValueForNamedPassword('PasswordRefGCV')
```

V Web Service Reference

These sections describe the Web Service endpoints provided for the User Application.

11 Provisioning Web Service

This section describes the Provisioning Web Service, which allows SOAP clients to access Provisioning functionality.

11.1 About the Provisioning Web Service

The Identity Manager User Application includes a workflow system that executes approval flows. A workflow process is based on a provisioning request definition, which is an XML document stored in the Identity Vault. The provisioning request definition describes an arbitrary topology using activities and links. For example, a provisioning request to grant an entitlement might have a workflow that collects approvals from relevant users and writes the entitlement to the directory.

To support access by third-party software applications, the provisioning workflow system includes a Web service endpoint. The endpoint offers all provisioning functionality (for example, allowing SOAP clients to start a new approval flow, or list currently executing flows). The Web service is built using the NetIQ Web Service SDK (WSSDK), which supports the WS-I Basic Profile, thus guaranteeing interoperability with other standards based SOAP implementations.

This Appendix describes the provisioning Web service in detail and shows how to access it using the Web or by writing a Java or C# client. We provide an overview of the operations in the SOAP endpoint and describe how to use the Web interface. We show how to develop a Java client using the SOAP toolkit included with Identity Manager provisioning, followed by how to write a C# client using Mono. The sample source code a the Java client and associated ANT build file is provided.

11.1.1 Provisioning Web Service Overview

Identity Manager is composed of two main systems: the Identity Vault and the workflow application. The Identity Vault is capable of connecting to a large number of different systems such as databases, financial systems, and other enterprise applications, and keep these systems synchronized. The rules for synchronizing the remote systems can be very complex and the Identity Vault engine supports a sophisticated scripting language for expressing the rules.

The workflow application is composed of several subsystems. The User Application provides a user-interface for workflows. The User Application is a Web application for requesting and managing approval flows. The Web application runs in a portal, which also includes administration portlets. The workflow application contains a security layer, a directory abstraction layer and a logging subsystem, which can send log events to NetIQ Sentinel. The workflow subsystem is responsible for executing approval flows. The User Application runs on an application server (for example, JBoss) and uses a database (for example, Oracle) for persistence.

The Web service for the workflow system is only used by the User Application driver, which is capable of listening to certain events emitted by the Identity Vault engine and convert these events into an appropriate SOAP message. For example, when a specific attribute in the Identity Vault changes, the Identity Vault engine emits an event, which the User Application picks up from the subscriber channel. The User Application driver then sends a SOAP message to the provisioning Web service to start a new approval flow.

11.1.2 Removing Administrator Credential Restrictions

By default, the requirement for invoking the public interfaces for the SOAP services is that the HTTP session logged in user must have administrator credentials. The Provisioning and Directory Services require Provisioning Administrator credentials. The Role Service and Resource Service require Role Administrator and Resource Administrator credentials respectively. The restrictions can be removed to allow a session with a logged in user who does not have administrator credentials to invoke the methods for the services by changing the configuration settings for the service. In order to do this, you must extract the configuration files from the User Application war, make the appropriate changes, and import the files back into the User Application WAR. The details for changing the Provisioning Service follow. Instructions for the other SOAP services are provided with the documentation for these services.

To remove the administrator credential restriction for the Provisioning Service:

- 1 Open the `ism-configuration.properties` file, located by default in the `/netiq/idm/apps/tomcat/conf` directory.
- 2 Change `WorkflowService/SOAP-End-Points-Accessible-By-ProvisioningAdminOnly` to `false`.
- 3 Save and close the file.

These are the methods that can be invoked by users without Provisioning Administrator credentials if the `WorkflowService/SOAP-End-Points-Accessible-By-ProvisioningAdminOnly` property is set to `false`:

- ◆ `getAllProvisioningRequests(String)`
- ◆ `getDataItems(String workId)`
- ◆ `getDefinitionById(String definitionID, String recipient)`
- ◆ `getProvisioningCategories()`
- ◆ `getProvisioningRequests(String recipient, String category, String operation)`
- ◆ `getWork(String workId)`
- ◆ `getWorkEntries(T_WorkEntryQuery query, int maxRecords)`
- ◆ `start(String processId, String recipient, DataItemArray items)`
- ◆ `startAsProxy(String processId, String recipient, DataItemArray items, String proxyUser)`
- ◆ `startAsProxyWithDigitalSignature(String processId, String recipient, DataItemArray items, String digitalSignature, SignaturePropertyArray digitalSignaturePropertyArray, String proxyUser)`
- ◆ `startWithCorrelationId(String processId, String recipient, DataItemArray items, String digitalSignature, SignaturePropertyArray digitalSignaturePropertyArray, String proxyUser, String correlationId)`
- ◆ `startWithDigitalSignature(String processId, String recipient, DataItemArray items, String digitalSignature, SignaturePropertyArray digitalSignaturePropertyArray)`

All other methods for this service always require Provisioning Administrator credentials independent of whether the `WorkflowService/SOAP-End-Points-Accessible-By-ProvisioningAdminOnly` property is set to `false`.

11.1.3 Provisioning Web Service Method Categories

The methods provided by the provisioning Web service endpoint are divided into six categories:

Table 11-1 Provisioning Web Service Operation Categories

Category	Description
Comments	Methods for retrieving comments and for adding a comment to a pending user activity
Configuration	Methods for getting and setting configuration parameters for the workflow system (for example, timeouts, thread pool settings).
Miscellaneous	Several unrelated methods (for example, for getting a JPG with a provisioning request's topology, for getting the XML definition of a provisioning request, and for getting the XML for the request form).
Processes	Methods for getting information about running and completed workflow processes.
Provisioning Requests	Methods for working with provisioning requests (for example, listing available provisioning requests, listing provisioning categories)
Work Entries	Methods for retrieving and manipulating work entries (items awaiting approval).

The methods provided by the provisioning Web service are described in detail in [Section 11.3, "Provisioning Web Service API,"](#) on page 247.

11.2 Developing Clients for the Provisioning Web Service

11.2.1 Web Access to the Provisioning Web Service

A SOAP-based Web service is usually accessed by inserting a SOAP message in the body of an HTTP Post request. The Web service toolkit used to build the provisioning Web service also supports access using HTTP GET. In other words, you can open the URL of the Web service endpoint in a browser and interact with the Web service. In particular, the provisioning Web service lets you invoke each of its operations.

Accessing the Test Page

You can access the provisioning Web Service endpoint using a URL similar to the following:

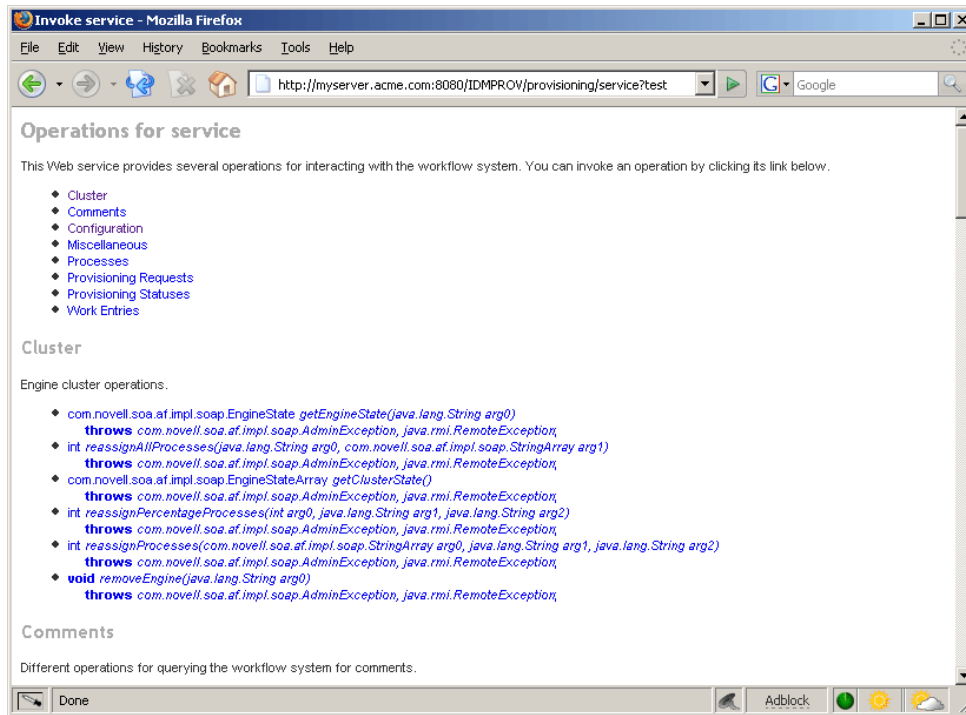
```
http://server:port/warcontext/provisioning/service?test
```

For example, if your server is named "myserver", your User Application is listening on port 8080, and your User Application war file is named "IDMPROV", the URL would be:

```
http://myserver:8080/IDMPROV/provisioning/service?test
```

The following page is displayed:

Figure 11-1 Web Service Test Page



You can also access the SOAP endpoint by going to the **Administration** within the User Application. To do this, you need to select the **Application Configuration** tab, then select **Web Services** from the left-navigation menu. After selecting **Web Services**, pick the Web Service endpoint you want from the list.

WARNING: The test page is disabled by default. Since some of the methods allow data to be updated, the test page presents a potential security vulnerability and should not be allowed in a production environment. For details on enabling the test page, see the instructions provided for the Role Service in “[Enabling the Test Page](#)” on page 359.

Entering Arguments for Operations

To see an example of an operation that is particularly useful to invoke from the browser, scroll down to the **Miscellaneous** section and click **getGraph**.

NOTE: The Graphviz program must be installed on the computer where the application server and the IDM User Application is running. For more information about Graphviz, see [Graphviz \(http://www.graphviz.org\)](http://www.graphviz.org).

A page is displayed that allows you to enter the parameters for the `getGraph` method.

Figure 11-2 Parameters for getGraph Method

Enter Parameters to Invoke getGraph

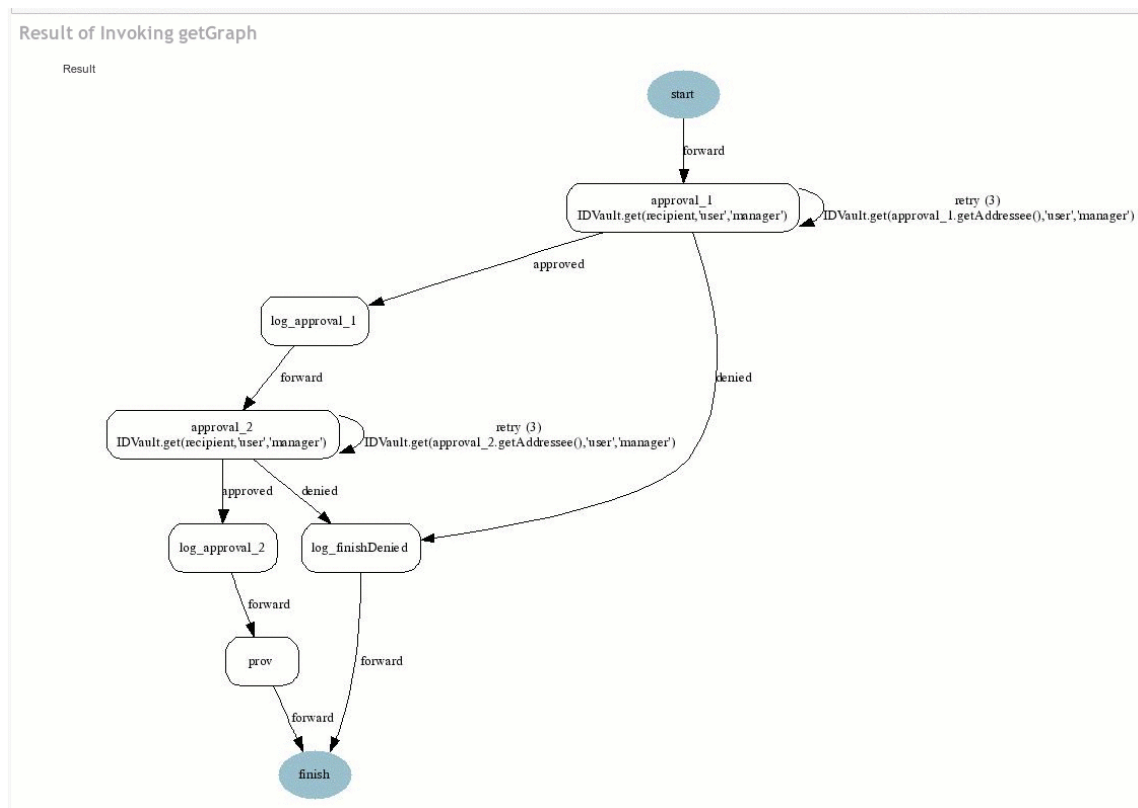
Get JPG image for workflow.

`processId` (java.lang.String):

[Back to home.](#)

The method takes one argument, which is the distinguished name of a provisioning request. Enter the DN, and the underlying workflow is displayed as a JPG file..

Figure 11-3 Output of getGraph



11.2.2 A Java Client for the Provisioning Web Service

This section describes how to develop a simple Java client for the provisioning Web service, which lists all the processes in the workflow system. For complete source code for the client, see “[Sample Code for the Java Client](#)” on page 243.

Prerequisites

To develop a Java client you must install a supported Java Developer’s Kit. Also, a client program needs the following JAR files:

activation.jar
commons-httpclient.jar
IDMfw.jar
IDMrbac.jar
log4j.jar
saaj-api.jar
wssdk.jar
commons-codec-1.3.jar
commons-logging.jar
jaxrpc-api.jar
mail.jar
workflow.jar
xpp3.jar

Developing a Java Client

Developing a client that accesses a Web service consists of two steps:

- ◆ Get the stub, which is the object that represents the remote service
- ◆ Invoke one or more of the operations available in the remote service

The Java programming model for Web services is very similar to RMI. The first step is to lookup the stub using JNDI:

```
InitialContext ctx = new InitialContext();
ProvisioningService service = (ProvisioningService)
ctx.lookup("xmlrpc:soap:com.novell.soa.af.impl.soap.ProvisioningService");
Provisioning prov = service.getProvisioningPort();
```

The first line of code creates the initial context for JNDI lookups. The second line looks up the service object, which is a kind of factory that can be used to retrieve the stub for the provisioning Web service. The last line gets the provisioning stub from the service.

Before invoking an operation on the provisioning stub, it is necessary to set some properties, including the credentials used for authentication on the service, as well as the endpoint URL.

```
Stub stub = (Stub) prov;
// set username and password
stub._setProperty(Stub.USERNAME_PROPERTY, USERNAME);
stub._setProperty(Stub.PASSWORD_PROPERTY, PASSWORD);
// set the endpoint URL
stub._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY, url);
```

These and other stub properties are described in more detail in [“Frequently Used Stub Constants” on page 241](#). Now that we have a fully configured stub, we can invoke the `getAllProcesses` operation and dump information about each of the processes returned on the console:


```

// invoke the getAllProcesses method
ProcessArray array = prov.getAllProcesses();
Process[] procs = array.getProcess();
// print process array
System.out.println("list of all processes:");
if (procs != null) {
for (int i = 0; i < procs.length; i++) {
System.out.println(" process with request identifier " +
procs[i].getRequestId());
System.out.println(" initiator = " + procs[i].getInitiator());
System.out.println(" recipient = " + procs[i].getRecipient());
System.out.println(" processId = " + procs[i].getProcessId());
procs[i].getCreationTime().getTime());
if (null != procs[i].getCompletionTime()) {
System.out.println(" completed = " +
procs[i].getCompletionTime().getTime());
}
System.out.println(" approval status = " +
procs[i].getApprovalStatus());
System.out.println(" process status = " +
procs[i].getProcessStatus());
if (i != procs.length - 1)
System.out.println();
}
}
}

```

A method invocation on the stub results in a SOAP message being sent using the HTTP transport to the provisioning Web service. For operations that have arguments, the stub takes care of marshaling those Java objects into XML. The Web service returns a SOAP message, and the stub unmarshals the XML, in this case converting it into a ProcessArray Java object.

Running the Client

The sample ANT build file has a target for running the client (see [“Sample Ant File” on page 246](#)). The client needs the JAR files described in [“Prerequisites” on page 239](#) to be in the CLASSPATH. You can change the code to have a different default address for the provisioning Web service SOAP endpoint, or simply specify it as a command line argument. For example:

```
ant -Durl=http://www.company.com:80/IDMProv/provisioning/service run
```

Frequently Used Stub Constants

The `com.novell.soa.ws.portable.Stub` class (which is part of WSSDK) supports several properties that can be used to configure a stub instance (for example, to fine-tune aspects of the HTTP communication). The following table lists a small subset of these properties, which are frequently used:

Table 11-2 Provisioning Web Service Stub Constants

Property	Type	Description
ENDPOINT_ADDRESS_PROPERTY	java.lang.String	The URL of the Web service. The URL protocol scheme can be HTTP or HTTPS depending on the requirements of the server. The path portion should be: <i>/IDMProv/provisioning/service</i>

Property	Type	Description
HTTP_HEADERS	java.util.Map	Additional HTTP headers as String name/value pairs.
HTTP_TIME_OUT	java.lang.Integer	The number of milliseconds to wait to establish a connection to the host before timing out.
HTTP_MAX_TOTAL_CONNECTIONS	java.lang.Integer	The number of concurrent connections that this client program can establish to all server hosts it accesses. The default limit is 20.
HTTP_MAX_HOST_CONNECTIONS	java.lang.Integer	The number of concurrent connections this client program can establish to an individual server host. The default limit is 2. This value may not exceed that of <code>HTTP_MAX_TOTAL_CONNECTIONS</code> , so if a client requires more than 20 connections to the server, it must also set <code>HTTP_MAX_TOTAL_CONNECTIONS</code> to the desired value.
USERNAME	java.lang.String	The user ID for HTTP authentication.
PASSWORD	java.lang.String	The password for HTTP authentication.
HTTP_PROXY_HOST	java.lang.String	The host DNS name of a proxy. Setting this property requires setting <code>HTTP_PROXY_PORT</code> as well.
HTTP_PROXY_PORT	java.lang.Integer	The port to use on a proxy. Setting this property requires setting <code>HTTP_PROXY_HOST</code> as well.
HTTP_PROXY_AUTH_SCHEME	java.lang.Integer	The authentication scheme (Basic or Digest) to use for a proxy.
HTTP_PROXY_USERNAME	java.lang.String	The user ID for HTTP authentication using a proxy.
HTTP_PROXY_PASSWORD	java.lang.String	The password for HTTP authentication via proxy.

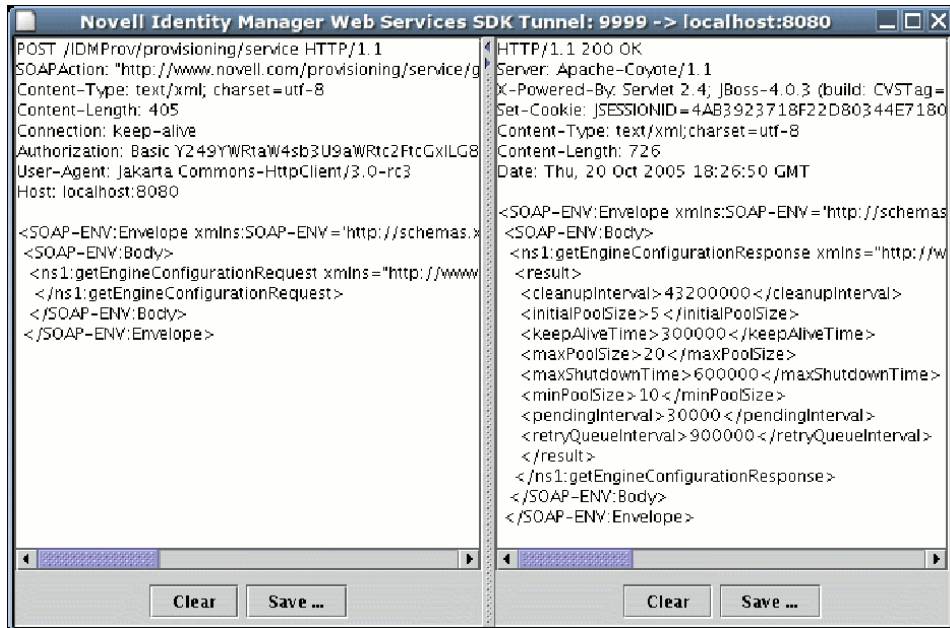
The TCP Tunnel

The TCP Tunnel is a useful tool for looking at the SOAP messages that are exchanged between a client and a server. The ANT build file (see [“Sample Ant File” on page 246](#)) has a target for starting the tunnel. Once the tunnel starts you need to enter the port on which the tunnel will listen, and the host/port of the remote Web service. The default settings cause the tunnel to listen on port 9999 and connect to a service running on localhost port 8080. The client program (see [“Developing a Java Client” on page 240](#)) uses the first command line parameter to set the `ENDPOINT_ADDRESS_PROPERTY`. Using the default values, you can run the client using the following command, after starting the tunnel:

```
ant -Durl=http://localhost:9999/IDMProv/provisioning/service run
```

[Figure 11-4](#) shows the TCP tunnel with a request SOAP message in the left panel and the message in the right panel.

Figure 11-4 TCP Tunnel



Sample Code for the Java Client

The following is the code for the Java client for listing all processes in the workflow system

```

package com.novell.examples;
import javax.naming.InitialContext;
import com.novell.soa.af.impl.soap.AdminException;
import com.novell.soa.af.impl.soap.Process;
import com.novell.soa.af.impl.soap.ProcessArray;
import com.novell.soa.af.impl.soap.Provisioning;
import com.novell.soa.af.impl.soap.ProvisioningService;
import com.novell.soa.ws.portable.Stub;
public class Client
{
    private static final String USERNAME = "admin";
    private static final String PASSWORD = "test";
    public static void main(String[] args)
    {
        try {
            String url = args.length > 0 ? args[0] :
                "http://localhost:8080/IDMProv/provisioning/service";
            listProcesses(url);
        } catch (AdminException ex) {
            System.out.println("command failed: " + ex.getReason());
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
    private static void listProcesses(String url)
    throws Exception
    {
        // get the stub
        InitialContext ctx = new InitialContext();
        ProvisioningService service = (ProvisioningService)
            ctx.lookup("xmlrpc:soap:com.novell.soa.af.impl.soap.ProvisioningService");
    }
}

```


Generating the Stub

Compared to the Java client developed in [“Developing a Java Client” on page 240](#), there is one additional step required when building the C# client. Since the stub for accessing the Web service SOAP endpoint is not provided, you must generate the stub from the WSDL document. Mono includes a compiler called `wSDL` that processes the WSDL file and creates the stub. You can download the WSDL file from your User Application server by accessing the following URL:

```
http://myserver:8080/IDMProv/provisioning/service?wsdl
```

Replace “myserver” with the name of your server, and “IDMProv” with the name of your User Application war file.

Compile the WSDL file using the following command:

```
wSDL Provisioning.wsdl
```

This will generate a C# file called `ProvisioningService.cs`, which you need to compile into a DLL using the following Mono C# compiler command:

```
mcs /target:library /r:System.Web.Services.dll ProvisioningService.cs
```

Compared to the Java client, the resulting `ProvisioningService.dll` file is the equivalent of `workflow.jar`, which contains the stub code and supporting classes for accessing the provisioning Web service. The following is the source code for the simple C# client that sets the flow retention time and displays the new value on the console:

```
using System;
using System.Net;
class provclient {
public static void Main(string [] args) {
// create the provisioning service proxy
ProvisioningService service = new ProvisioningService();
// set the credentials for basic authentication
service.Credentials = new NetworkCredential("admin", "test");
service.PreAuthenticate = true;
// set the value for completed request retention to 30 days
setCompletedProcessTimeoutRequest req = new
setCompletedProcessTimeoutRequest();
req.arg0 = 30;
service.setCompletedProcessTimeout(req);
// display the new value on the console
getCompletedProcessTimeoutResponse res = service.getCompletedProcessTimeout(new
getCompletedProcessTimeoutRequest());
Console.WriteLine(res.result);
}
}
```

You need to edit the file using the administrator credentials on your deployed Identity Manager system. Compile the client using the following command:

```
mcs /r:ProvisioningService.dll /r:System.Web provclient.cs
```

This generates the `provclient.exe` file.

Running the Client

Use the following command to run the client:

```
mono provclient.exe
```

11.2.4 Sample Ant File

The sample Ant file includes useful targets for extracting the necessary JAR files from the Identity Manager installation, compiling and running the Java client, and for launching the TCP Tunnel.

```
<?xml version="1.0"?>
<project name="client" default="all" basedir=".">
<target name="all" depends="clean, extract, compile"></target>
<!-- main clean target -->
<target name="clean">
<delete quiet="true" dir="classes"/>

<delete quiet="true" dir="lib"/>
</target>
<!-- init sets up the build environment -->
<target name="init">
<mkdir dir="classes"/>
<copy todir="${basedir}/lib">
<fileset dir="${basedir}" includes="log4j.properties"/>
</copy>
<!-- classpath -->
<path id="CLASSPATH">
<pathelement location="${basedir}/classes"/>
<fileset dir="${basedir}/lib" includes="*.jar"/>
</path>
</target>
<!-- extract -->
<target name="extract">
<property name="idm.home" value="/opt/netiq/idm3"/>
<property name="jboss.lib" value="${idm.home}/jboss-4.0.3/server/IDMProv/lib"/>
<mkdir dir="lib"/>
<unzip src="${idm.home}/IDMProv.war" dest="${basedir}/lib">
<patternset>
<include name="WEB-INF/lib/commons-codec-1.3.jar"/>
<include name="WEB-INF/lib/commons-httpclient.jar"/>
<include name="WEB-INF/lib/commons-logging.jar"/>
<include name="WEB-INF/lib/jaxrpc-api.jar"/>
<include name="WEB-INF/lib/saaj-api.jar"/>
<include name="WEB-INF/lib/xpp3.jar"/>
<include name="WEB-INF/lib/workflow.jar"/>
<include name="WEB-INF/lib/wssdk.jar"/>
<include name="WEB-INF/lib/IDMfw.jar"/>
</patternset>
</unzip>
<move todir="${basedir}/lib">
<fileset dir="${basedir}/lib/WEB-INF/lib" includes="*.jar"/>
</move>
<delete quiet="true" dir="${basedir}/lib/WEB-INF"/>
<copy todir="${basedir}/lib">
<fileset dir="${jboss.lib}" includes="activation.jar, mail.jar, log4j.jar"/>
</copy>
</target>
<!-- tunnel -->
<target name="tunnel" depends="init">
<java classname="com.novell.soa.ws.impl.tools.tcptunnel.Tunnel" fork="true"
spawn="true">
<classpath refid="CLASSPATH"/>
</java>
</target>
<!-- compile -->
```

```

<target name="compile" depends="init">
<javac srcdir="${basedir}" destdir="classes"
includes="Client.java">
<classpath refid="CLASSPATH"/>
</javac>
</target>
<!-- run -->
<target name="run" depends="init">
<property name="url" value="http://localhost:8080/IDMProv/provisioning/service"/>
<java classname="com.novell.examples.Client" fork="true">
<arg line="${url}"/>
<classpath refid="CLASSPATH"/>
</java>
</target>
</project>

```

11.2.5 Sample Log4J File

The following log4j file sets the default log level to "error":

```

log4j.rootCategory=ERROR, R
log4j.appender.R=org.apache.log4j.ConsoleAppender
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%-5p: %m%n

```

11.3 Provisioning Web Service API

This section provides details about the Provisioning Web service methods.

All of the methods throw `com.novell.soa.af.impl.soap.AdminException` and `java.rmi.RemoteException`. To improve readability, the throws clause has been omitted from the method signatures.

11.3.1 Processes

This section provides reference information for each Processes method.

getProcessesByQuery

Used to get information about processes.

Method Signature

```

com.novell.soa.af.impl.soap.ProcessArray
getProcessesByQuery(com.novell.soa.af.impl.soap.T_ProcessInfoQuery query, int
maxRecords)

```

Example

```
//  
  
// Query information about processes for a user that are running and  
// have not been approved yet.  
String logic = "AND";  
T_ProcessInfoOrder order = T_ProcessInfoOrder.APPROVAL_STATUS;  
int CHOICE_SIZE = 4;  
Integer approvalStatusInteger = new Integer(ProcessConstants.PROCESSING);  
Integer processStatusInteger = new Integer(ProcessConstants.RUNNING);  
//  
// Setup the query with the above params  
T_ProcessInfoQueryChoice [] choice = new  
T_ProcessInfoQueryChoice[CHOICE_SIZE];  
choice[0] = new T_ProcessInfoQueryChoice();  
choice[0].setApprovalStatus(approvalStatusInteger);  
choice[1] = new T_ProcessInfoQueryChoice();  
choice[1].setProcessStatus(processStatusInteger);  
choice[2] = new T_ProcessInfoQueryChoice();  
choice[2].setRecipient(recipient);  
choice[3] = new T_ProcessInfoQueryChoice();  
choice[3].setRequestId(requestId);  
  
int maxRecords = -1;  
T_ProcessInfoQuery processInfoQuery =  
    new T_ProcessInfoQuery(T_Logic.fromString(logic), order, choice);  
ProcessArray processArray = stub.getProcessesByQuery(processInfoQuery,  
maxRecords);
```

getProcessesByStatus

Used to get information about processes with a specified status (for example, running processes).

Method Signature

```
public com.novell.soa.af.impl.soap.ProcessArray  
getProcessesByStatus(com.novell.soa.af.impl.soap.T_ProcessStatus status)
```

Example

```
T_ProcessStatus processStatus = T_ProcessStatus.Running;  
//  
// Get processes by status  
ProcessArray processArray = stub.getProcessesByStatus(processStatus);  
Process [] process = processArray.getProcess();
```

getProcesses

Used to get information about processes, specified by processID.

Method Signature

```
com.novell.soa.af.impl.soap.ProcessArray getProcesses(java.lang.String id, long  
time, com.novell.soa.af.impl.soap.T_Operator op, java.lang.String initiator,  
java.lang.String recipient)
```


Parameters

Parameter	Description
processId	The process Id (java.lang.String).
creationTime	The time at which the process was started (long).
op	The operator to use. The operators are: EQ - equals LT - less than LE - less than or equal to GT - greater than GE - greater than or equal to
initiator	The initiator of the workflow.
recipient	The recipient of the approval activity.

Example

```
int processMatchCount = 0;
T_Operator operator = T_Operator.GT;
long currentTimeInMillis = System.currentTimeMillis();
String [] requestIds = requestIdArray.getString();
//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable Active Directory
Account (Mgr Approve-No Timeout)");
//
// Start request
// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap, null);
sleep(5);

Process process = stub.getProcess(requestId);
if(process != null)
{
    String processId = process.getProcessId();
    String initiator = process.getInitiator();

    ProcessArray processArray = stub.getProcesses(processId,
currentTimeInMillis, operator, initiator, recipient);
}
```

getAllProcesses

Used to get information about all running and completed provisioning requests.

Method Signature

```
com.novell.soa.af.impl.soap.ProcessArray getAllProcesses()
```

Example

```
ProcessArray array = stub.getAllProcesses();
Process [] processes = array.getProcess();
if(_process != null)
{
    sb = new StringBuffer();
    sb.append("\nProcess List:");
    for(int index = 0; index < _process.length; index++)
    {
        String processId = _process[index].getProcessId();
        String approvalStatus = _process[index].getApprovalStatus();
        Calendar completionTime = _process[index].getCompletionTime();
        Calendar creationTime = _process[index].getCreationTime();
        String engineId = _process[index].getEngineId();
        String proxy = _process[index].getProxy();
        String initiator = _process[index].getInitiator();
        String processName = _process[index].getProcessName();
        String processStatus = _process[index].getProcessStatus();
        String p_recipient = _process[index].getRecipient();
        String p_requestId = _process[index].getRequestId();
        int valueOfapprovalStatus = _process[index].getValueOfApprovalStatus();
        int valueOfprocessStatus = _process[index].getValueOfProcessStatus();
        String version = _process[index].getVersion();
    }
}
```

getProcessesArray

Used to limit the number of processes returned. If the limit you specify is less than the system limit, the number you specify is returned. If you exceed the system limit, the Workflow Engine returns the system limit. If the limit you specify is less than or equal to 0, the Workflow Engine returns all processes.

Method Signature

```
com.novell.soa.af.impl.soap.ProcessArray getProcessesArray(int maxRecords);
```

Example

```
/**
 * Method to augment the getAllProcesses() method that impose limits
 * on the number of processes returned.
 * @throws TestProgramException
 */
public void adding_Limits_To_getProcessArray_TestCase()
throws TestProgramException
{
    String recipient =
ServiceUtils.getInstance().getLoginData().getUsername(LoginData.RECIPIENT_TYPE);
    String requestNameToStart =
provUtils.getProvisioningResourceNameForRecipient(recipient,
"Enable Active Directory");

    //
    // Get the stub
    Provisioning stub = ServiceUtils.getInstance().getProvisioningStub();
    try
    {
        //
    }
}
```

```

// Start multiple requests
final int NUMBER_OF_REQUESTS_TO_START = 2;

Map map = MapUtils.createAndSetMap(new Object[] {
    Helper.RECIPIENT, recipient,

IProvisioningConstants.PROVISIONING_REQUEST_TO_START, requestNameToStart});
//
// Start request(s)
StringArray requestIdArray =
    provUtils.startMultipleProvisioningRequests(map, null,
NUMBER_OF_REQUESTS_TO_START);
    LoggerUtils.sleep(3);
    LoggerUtils.sendToLogAndConsole("Started " +
NUMBER_OF_REQUESTS_TO_START + " provisioning requests");
//
// New method to limit the number of processes returned
//
// Test Results : maxProcesses <= 0 returns all processes
//                 maxProcesses up to system limit returns maxProcess
count
//                 maxProcesses > system limit returns system limit
int maxProcesses = 10;
ProcessArray processArray = stub.getProcessesArray(maxProcesses);
Process [] processes = processArray.getProcess();
if(processes != null)
{
    LoggerUtils.sendToLogAndConsole("Process count returned: " +
processes.length);
    Assert.assertEquals("Error: Processes returned shouldn't exceed max
count.",
        maxProcesses, processes.length);
}
}
catch(AdminException error)
{
    RationalTestScript.logError(error.getReason() );
    throw new TestProgramException(error.getReason() );
}
catch(RemoteException error)
{
    RationalTestScript.logError(error.getMessage() );
    throw new TestProgramException(error.getMessage() );
}
}
}

```

getProcessesById

Used to get information about a specific process, specified by the Process Id.

Method Signature

```
com.novell.soa.af.impl.soap.ProcessArray getProcessesById(java.lang.String id)
```

Example

```
Process [] allProcesses = stub.getAllProcesses().getProcess();
if(allProcesses != null)
{
    String processId = allProcesses[0].getProcessId;
    ProcessArray array = stub.getProcessesById(processId);
    Process [] processes = array.getProcess();
}
```

terminate

Used to terminate a running provisioning request.

Method Signature

```
void terminate(java.lang.String requestId,
com.novell.soa.af.impl.soap.T_TerminationType state, java.lang.String comment)
```

Parameters

Parameter	Description
requestId	The Id of the provisioning request.
state	The reason for terminating the process. The choices are: RETRACT ERROR
comment	Adds a comment about the terminate action.

Example

```
//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable Active Directory
Account (Mgr Approve-No Timeout)");
//
// Start request
// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap, null);
sleep(5);
//
// Now retract the request
T_TerminationType terminationType = T_TerminationType.RETRACT;
stub.terminate(requestId, terminationType, terminationType.getValue() + " the
request");
```

getProcess

Used to get information about a running or completed provisioning request, specified by Request ID.

Method Signature

```
com.novell.soa.af.impl.soap.Process getProcess(java.lang.String requestId)
```

Example

```
//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable Active Directory
Account (Mgr Approve-No Timeout)");
//
// Start request

// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap, null);
sleep(5);

Process process = stub.getProcess(requestId);
if(process != null)
{
    boolean bMatchProcess = false;
    if( (recipient.compareTo(process.getRecipient()) == 0) &&
(requestId.compareTo(process.getRequestId()) == 0) )
    {
        bMatchProcess = true;
    }
    if(bMatchProcess)
    {
        String msg = "Found process with requestId : " + requestId;
        LoggerUtils.sendToLogAndConsole(msg);
    }
    //
    // Assert if we could not find a match
    Assert.assertTrue("Could not find process with request id: " + requestId,
bMatchProcess);
}
```

getProcessesByCreationTime

Used to get information about processes created between the current time and the time at which the workflow process was created.

Method Signature

```
com.novell.soa.af.impl.soap.ProcessArray getProcessesByCreationTime(long time,
com.novell.soa.af.impl.soap.T_Operator op)
```

Parameters

Parameter	Description
creationTime	The time at which the process was started.
op	The operator to use. The operators are: EQ - equals LT - less than LE - less than or equal to GT - greater than GE - greater than or equal to

Example

```
T_Operator operator = T_Operator.GT;
//
// Get processes with operator relative to the current time
long currentTime = System.currentTimeMillis();//currentDateTime.getTime();
ProcessArray processArray = stub.getProcessesByCreationTime(currentTime,
operator);
```

getProcessesByApprovalStatus

Used to get information about processes with a specified approval status (Approved, Denied, or Retracted).

Method Signature

```
com.novell.soa.af.impl.soap.ProcessArray
getProcessesByApprovalStatus(com.novell.soa.af.impl.soap.T_ApprovalStatus status)
```

Example

```
T_ApprovalStatus approvalStatus = T_ApprovalStatus.Approved;
//
// Get all the processes based upon approval status above
ProcessArray processArray = stub.getProcessesByApprovalStatus(approvalStatus);
Process [] processes = processArray.getProcess();
```

getProcessesByRecipient

Used to get information about processes that have a specific recipient Id.

Method Signature

```
com.novell.soa.af.impl.soap.ProcessArray getProcessesByRecipient(java.lang.String
recipient)
```

Example

```
String recipient = "cn=ablake,ou=users,ou=idmsample-komodo,o=netiq";
```

```
//  
// Get processes by recipient  
ProcessArray processArray = stub.getProcessesByRecipient(recipient);  
Process [] process = processArray.getProcess();
```

getProcessesByInitiator

Used to get information about processes that have a specific initiator Id.

Method Signature

```
com.novell.soa.af.impl.soap.ProcessArray getProcessesByInitiator(java.lang.String  
initiator)
```

Example

```
String initiator = "cn=admin,ou=idmsample-komodo,o=netiq";  
  
//  
// Get processes by initiator  
ProcessArray processArray = stub.getProcessesByInitiator(initiator);  
Process [] process = processArray.getProcess();
```

setResult

Used to set the entitlement result (approval status) of a previously completed provisioning request.

Method Signature

```
void setResult(java.lang.String requestId,  
com.novell.soa.af.impl.soap.T_EntitlementState state,  
com.novell.soa.af.impl.soap.T_EntitlementStatus status, java.lang.String message)
```

Parameters

Parameter	Description
requestId	The Id of the provisioning request.
state	The state of the provisioning request. The possible values are: Unknown Granted Revoked
status	The status of the provisioning request. The possible values are: Unknown Success Warning Error Fatal Submitted
message	A message about the entitlement result.

Example

```
//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable Active Directory
Account (Mgr Approve-No Timeout)");
//
// Start request
// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap, null);
sleep(5);

//
// Get the process id for this running process
Process process = stub.getProcess(requestId);
String processId = null;
if (process != null)
    processId = process.getProcessId();
//
// Reset the state of the provisioning request
T_EntitlementState newEntitlementState =
T_EntitlementState.Revoked;
T_EntitlementStatus newEntitlementStatus = T_EntitlementStatus.Success;
String comment = "Revoked the provisioning request";
stub.setResult(processId, newEntitlementState, newEntitlementStatus, comment);
```


getProcessesByCreationInterval

Used to get information about processes started between two specified times.

Method Signature

```
com.novell.soa.af.impl.soap.ProcessArray getProcessesByCreationInterval(long start, long end)
```

Parameters

Parameter	Description
startTime	The start time (YYYY/MM/DD).
endTime	The end time (YYYY/MM/DD).

Example

```
    long startTime = System.currentTimeMillis();
    //
    // Initialize and start a provisioning request
    HashMap provMap = new HashMap();
    provMap.put(Helper.RECIPIENT, recipient);
    provMap.put("Provisioning_Request_To_Start_Key", "Enable Active Directory
Account (Mgr Approve-No Timeout)");
    //
    // Start request
    // Calls method startProvisioningRequest on the provUtils
    // utility object which refers to a utility class that does not
    // ship with the Identity Manager User Application.
    String requestId = provUtils.startProvisioningRequest(provMap, null);
    sleep(5);

    long endTime = System.currentTimeMillis();
    //
    // Get all the processes between the start and end time    ProcessArray
processArray = stub.getProcessesByCreationInterval(startTime, endTime);
    Process [] processes = processArray.getProcess();
```

11.3.2 Provisioning

This section provides reference information for each Provisioning method.

multiStart

Used to start a workflow request for each specified recipient.

Method Signature

```
com.novell.soa.af.impl.soap.StringArray multiStart(java.lang.String processId,
com.novell.soa.af.impl.soap.StringArray recipients,
com.novell.soa.af.impl.soap.DataItemArray items)
```

Parameters

Parameter	Description
processId	The Id of the provisioning request to start.
recipients	The DN of each recipient.
dataItem	The list of data items for the provisioning request.

Example

```
ProvisioningRequestArray requestArray =
stub.getAllProvisioningRequests(recipient);

//
// If there are some then,
if(requestArray != null)
{
    String Id = " ";
    StringArray requestIdStringArray = null;
    String [] listOfRecipients = {recipient, addressee};
    //
    // Select a provisioning resource
    String requestNameToStart = "Enable Active Directory Account (Mgr Approve-
No Timeout)";
    //
    // Loop thru and find the request that we want to start
    ProvisioningRequest [] requests = requestArray.getProvisioningrequest();
    for(int index = 0; index < requests.length; index++)
    {
        //
        // Is this the name of the request to start?
        if(requests[index].getName().compareTo(requestNameToStart) == 0)
        {
            //
            // Get the current associated data items. Replicate a new
            // dataitem array excluding the null values.
            Id = requests[index].getId();
            DataItem [] dataItem = requests[index].getItems().getDataitem();
            if(dataItem != null)
            // Call method replicateDataItemArray on the
            // provUtils utility object, which refers to a
            // utility class that does not ship with the
            // Identity Manager User Application.
            {
                DataItemArray newDataItemArray =
```

```

provUtils.replicateDataItemArray(dataItem);
    //
    // Create a string array initializing with multiple recipients
    StringArray listOfRecipientsStringArray = new
StringArray(listOfRecipients);
    //
    // Start the request for multiple recipients
    logStep("Calling stub.multiStart(" + Id +
",listOfRecipientsStringArray,newDataItemArray)");
    requestIdStringArray = stub.multiStart(Id,
listOfRecipientsStringArray, newDataItemArray);
    }
}
}

```

start

Used to start a provisioning request.

Method Signature

```

java.lang.String start(java.lang.String processId, java.lang.String recipient,
com.novell.soa.af.impl.soap.DataItemArray items)

```

Parameters

Parameter	Description
processId	The Id of the provisioning request to start.
recipient	The DN of each recipient.
dataItem	The list of data items for the provisioning request.

Example

```

//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable Active Directory
Account (Mgr Approve-No Timeout)");
//
// Start request
// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap, null);
sleep(5);

```

The example above calls the startProvisioningRequest method. This method is not part of the IDM User Application. We show it here to finish illustrating the example:

```

/**
 *Method to start a provisioning request using the supplied
 *Map and dataitem object. Handling of digital certificate
 *resources is also handled.
 * @param _map
 * @param _in_dataItem
 * @return String
 * @throws TestProgrammException
 */
public String startProvisioningRequest(Map _map, DataItem []
_in_dataItem) throws TestProgramException
{
    String requestId = null;
    try
    {
        String recipient =(String)_map.get(Helper.RECIPIENT);
        String requestToStart =
(String)_map.get(IProvisioningConstants.PROVISIONING_REQUEST_TO_START);
        String proxyUser =(String)_map.get(IWorkflowConstants.PROXY_USER);
        String digitalSignature =
(String)_map.get(IDigitalSignatureConstants.DIGITAL_SIGNATURE);
        RationalTestScript.logInfo("Step: Calling
startProvisioningRequest(_map)");
        //
        //Get the stub
        Provisioning stub = ServiceUtils.getInstance().getProvisioningStub();
        //
        //Get all the available resource requests for the recipient
        RationalTestScript.logInfo("Step: Calling stub.getAllProvisioningRequests("
+ recipient + ")");
        ProvisioningRequestArray requestArray =
stub.getAllProvisioningRequests(recipient);

        if(requestArray != null)
        {
            //
            //Get the provisioning request from the array
            ProvisioningRequest request =
getProvisioningRequestFromArray(requestArray, requestToStart);
            if(request != null)
            {
                DataItem [] dataItem = null;
                DataItemArray newDataItemArray = null;
                //
                // If the supplied data item is null then just replicate
                // what currently exists with the request.
                if(_in_dataItem == null)
                {
                    //
                    // Use the current data item associated with the request
                    dataItem = request.getItems().getDataitem();
                    if(dataItem != null)
                    {
                        newDataItemArray = replicateDataItemArray(dataItem);
                    }
                }
            }
            else
            {
                //

```

```

        // Set the incoming data item array
        newDataItemArray = new DataItemArray();
        newDataItemArray.setDataitem(_in_dataItem);
    }
    //
    // Start the Provisioning request for the recipient
    if(proxyUser == null && digitalSignature == null)
    {
        RationalTestScript.logInfo("Step: Calling stub.start(" +
request.getId() + ", " + recipient + "dataItemArray)");
        requestId = stub.start(
            request.getId(),
            recipient,
            newDataItemArray);
    }
    else if(proxyUser != null && digitalSignature == null)
    }
    .
    .
    .

```

getAllProvisioningRequests

Used to return an array of available provisioning requests.

Method Signature

```

com.novell.soa.af.impl.soap.ProvisioningRequestArray
getAllProvisioningRequests(java.lang.String recipient)

```

Example

```

//
// Get all the provisioning requests for this recipient

ProvisioningRequestArray provReqArray =
stub.getAllProvisioningRequests(recipient);
ProvisioningRequest [] provRequest = provReqArray.getProvisioningrequest();
if(provRequest != null)
{
    String description = provRequest[0].getDescription();
    String category = provRequest[0].getCategory();
    String digitalSignatureType = provRequest[0].getDigitalSignatureType();
    String requestId = provRequest[0].getId();
    DataItemArray itemArray = provRequest[0].getItems();
    String legalDisclaimer = provRequest[0].getLegalDisclaimer();
    String name = provRequest[0].getName();
    String operation = provRequest[0].getOperation();
}

```

getProvisioningRequests

Used to return an array of provisioning requests for a specified category and operation.

Method Signature

```
com.novell.soa.af.impl.soap.ProvisioningRequestArray  
getProvisioningRequests(java.lang.String recipient, java.lang.String category,  
java.lang.String operation)
```

Parameters

Parameter	Description
recipient	The recipient of the provisioning request.
category	The category of the provisioning request.
operation	The provisioning request operation (0=Grant,1=Revoke, 2=Both)

Example

```
String operation = IProvisioningRequest.GRANT;  
try  
{  
    //  
    // Get the stub  
    ProvisioningStub stub = ServiceUtils.getInstance().getProvisioningStub();  
    logStep("Calling stub.getProvisioningCategories()");  
    StringArray categoriesStringArray = stub.getProvisioningCategories();  
    String [] categories = categoriesStringArray.getString();  
    //  
    // Loop thru and get the provisioning requests for each category  
    for(int index = 0; index < categories.length; index++)  
    {  
        //  
        // Get the provisioning request based upon recipient  
        logStep("Calling stub.getProvisioningRequests(" + recipient + "," +  
categories[index] + "," + operation + ")");  
        ProvisioningRequestArray provRequestArray =  
stub.getProvisioningRequests(recipient, categories[index], operation);  
        ProvisioningRequest [] provRequests =  
provRequestArray.getProvisioningrequest();  
    }  
}
```

getProvisioningCategories

Used to get the list of available provisioning categories.

Method Signature

```
com.novell.soa.af.impl.soap.StringArray getProvisioningCategories()
```

Example

```
StringArray categoriesStringArray = stub.getProvisioningCategories();  
String [] categories = categoriesStringArray.getString();
```

startAsProxy

Used to start a workflow as a proxy.

Method Signature

```
java.lang.String startAsProxy(java.lang.String processId, java.lang.String recipient, com.novell.soa.af.impl.soap.DataItemArray items, java.lang.String proxyUser)
```

Parameters

Parameter	Description
processId	The Id of the provisioning request.
recipient	The recipient of the provisioning request.
Items	The data items for the provisioning request.
proxyUser	The DN of the proxy user.

Example

```
ProvisioningRequestArray requestArray =
stub.getAllProvisioningRequests(recipient);
//
// If there are some then,
if(requestArray != null)
{
    String Id = " ";
    String requestId = " ";
    String requestNameToStart = "Enable Active Directory Account (Mgr Approve-
No Timeout)";
    //
    // Loop thru and find the request that we want to start
    ProvisioningRequest [] requests = requestArray.getProvisioningrequest();
    for(int index = 0; index < requests.length; index++)
    {
        //
        // Is this the name of the request to start?
        if(requests[index].getName().compareTo(requestNameToStart) == 0)
        {
            //
            // Get the current associated data items. Replicate a new
            // dataitem array excluding the null values.
            Id = requests[index].getId();
            DataItem [] dataItem = requests[index].getItems().getDataitem();
            if(dataItem != null)
            {
                // Call method replicateDataItemArray on the
                // provUtils utility object, which refers to a
```


Parameter	Description
maxRecords	Used to specify maximum number of records to retrieve. A value of -1 returns unlimited records.

Example

```

//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable Active Directory
Account (Mgr Approve-No Timeout)");
//
// Start request
// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap, null);
sleep(5);
//
//
T_ProvisioningStatusQueryChoice [] choice = new
T_ProvisioningStatusQueryChoice[3];
choice[0] = new T_ProvisioningStatusQueryChoice();
choice[0].setRecipient(recipient);
choice[1] = new T_ProvisioningStatusQueryChoice();
choice[1].setRequestId(requestId);
choice[2] = new T_ProvisioningStatusQueryChoice();
choice[2].setStatus(new Integer(ProcessConstants.PROCESSING) );
//
// Initialize the query
T_ProvisioningStatusQuery query = new T_ProvisioningStatusQuery(T_Logic.AND,
T_ProvisioningStatusOrder.STATUS, choice);
//
// Make the query
StringBuffer sb = new StringBuffer();
int maxRecords = -1;

ProvisioningStatusArray provStatusArray = stub.getProvisioningStatuses(query,
maxRecords);

```

startWithDigitalSignature

Used to start a workflow and specify that a digital signature is required.

Method Signature

```

java.lang.String startWithDigitalSignature(java.lang.String processId,
java.lang.String recipient, com.novell.soa.af.impl.soap.DataItemArray items,
java.lang.String digitalSignature,
com.novell.soa.af.impl.soap.SignaturePropertyArray digitalSignaturePropertyArray)

```

Parameters

Parameter	Description
processId	The request identifier.
recipient	The request recipient.
items	The data items for the provisioning request.
digital signature	The digital signature.
digitalSignaturePropertyArray.	The digital signature property map.

Example

```
String recipient =
ServiceUtils.getInstance().getLoginData().getUsername(LoginData.RECIPIENT_TYPE);
//
// Get the digital signature string for admin
String digitalSignature =
DigitalSignatureUtils.getDigitalSignatureFromFile(IDigitalSignatureConstants.ADMIN
_DIGITAL_SIGNATURE_FILENAME);

ProvisioningRequestArray requestArray =
stub.getAllProvisioningRequests(recipient);
//
// If there are some then,

if(requestArray != null)
{
String Id = " ";
String requestId = " ";
String requestNameToStart = "Enable Active Directory Account (Mgr Approve-
No Timeout)";
//
// Loop thru and find the request that we want to start
ProvisioningRequest [] requests = requestArray.getProvisioningrequest();
for(int index = 0; index < requests.length; index++)
{
//
// Is this the name of the request to start?
if(requests[index].getName().compareTo(requestNameToStart) == 0)
{
//
// Get the current associated data items. Replicate a new
// dataitem array excluding the null values.
Id = requests[index].getId();
DataItem [] dataItem = requests[index].getItems().getDataitem();
if(dataItem != null)
{
// Call method replicateDataItemArray on the
// provUtils utility object, which refers to a
```



```

        String requestId = " ";
        String requestNameToStart = "Enable Active Directory Account (Mgr Approve-
No Timeout)";
        //
        // Loop thru and find the request that we want to start
        ProvisioningRequest [] requests = requestArray.getProvisioningrequest();
        for(int index = 0; index < requests.length; index++)
        {
            //
            // Is this the name of the request to start?
            if(requests[index].getName().compareTo(requestNameToStart) == 0)
            {
                //
                // Get the current associated data items. Replicate a new
                // dataitem array excluding the null values.
                Id = requests[index].getId();
                DataItem [] dataItem = requests[index].getItems().getDataitem();
                if(dataItem != null)
                {
                    // Call method replicateDataItemArray on the
                    // provUtils utility object, which refers to a
                    // utility class that does not ship with the
                    // Identity Manager User Application.
                    DataItemArray newDataItemArray =
provUtils.replicateDataItemArray(dataItem);
                    //
                    // Start a digitally signed provisioning resource as proxy for
                    the recipient

                                requestId =
stub.startAsProxyWithDigitalSignature(request.getId(), recipient,
newDataItemArray, digitalSignature, null, proxyUser);
                }
            }
        }
    }
}

```

startWithCorrelationId

Used to start a workflow with a correlation ID. The correlation ID provides a way to track a set of related workflow processes. When started with this method, workflow processes can be queried and sorted by correlation ID.

Method Signature

```

java.lang.String startWithCorrelationId(java.lang.String processId,
java.lang.String recipient, com.novell.soa.af.impl.soap.DataItemArray items,
java.lang.String signature, com.novell.soa.af.impl.soap.SignaturePropertyArray
props, java.lang.String proxyUser, java.lang.String correlationId)
throws com.novell.soa.af.impl.soap.AdminException, java.rmi.RemoteException;

```

Parameters

Parameter	Description
processId	The request identifier.
recipient	The request recipient.
items	The data items for the provisioning request.
digital signature	The digital signature.
digitalSignaturePropertyArray	The digital signature property map.
proxyUser	The DN of the proxy user.
correlationID	The string that identifies the correlation ID. The correlation ID cannot be longer than 32 characters.

11.3.3 Work Entries

This section provides reference information for each Work Entries method.

forward

Used to forward a task to the next activity in the workflow with the appropriate action (approve, deny, refuse).

Method Signature

```
void forward(java.lang.String wid, com.novell.soa.af.impl.soap.T_Action action, com.novell.soa.af.impl.soap.DataItemArray items, java.lang.String comment)
```

Parameters

Parameter	Description
wid	The work Id.
action	The action to take (approve, deny, refuse).
items	The data items required by the workflow.
comment	The comment.

Example

```
//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable Active Directory
Account (Mgr Approve-No Timeout)");
//
// Start request
// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap, null);
sleep(5);
//
// Get the process id for this running process
Process process = stub.getProcess(requestId);
String processId = null;
if(process != null)
    processId = process.getProcessId();

T_Action action = T_Action.APPROVE;

T_Logic logic = T_Logic.AND;

T_WorkEntryOrder workEntryOrder = T_WorkEntryOrder.REQUEST_ID;

T_WorkEntryQueryChoice [] workEntryqueryChoice = new
T_WorkEntryQueryChoice[3];
workEntryqueryChoice[0] = new T_WorkEntryQueryChoice();
workEntryqueryChoice[0].setRecipient(recipient);
workEntryqueryChoice[1] = new T_WorkEntryQueryChoice();
workEntryqueryChoice[1].setRequestId(requestId);
workEntryqueryChoice[2] = new T_WorkEntryQueryChoice();
workEntryqueryChoice[2].setProcessId(processId);
//
// Create work entry query
T_WorkEntryQuery query = new T_WorkEntryQuery(logic, _workEntryOrder,
workEntryqueryChoice);
//
// Get all work entries (max records)
WorkEntryArray workEntryArray = stub.getWorkEntries(query, -1);

WorkEntry [] workEntry = workEntryArray.getWorkentry();

if(workEntry != null
{
    for(int wIndex = 0; wIndex < workEntry.length; wIndex++)
    {
        String workId = workEntry[wIndex].getId();
        //
        //
        LoggerUtils.sendToLogAndConsole("Forwarding : " +
workEntry[wIndex].getActivityName() + " work id: " + workId);
        //
        // Get the dataitem for this item of work
        DataItemArray dataItemArray = stub.getWork(workId);
        DataItem [] dataItem = dataItemArray.getDataitem();
```

```

        DataItemArray newDataItemArray = null;
        if(dataItem != null)
            // Call method replicateDataItemArray on the
            // provUtils utility object, which refers to a
            // utility class that does not ship with the
            // Identity Manager User Application.
            newDataItemArray = provUtils.replicateDataItemArray(dataItem);
        else
            throw new TestProgramException("DataItem is null.");
        //
        // Claim request for recipient
        String comment = _action.toString() + " this request: " + requestId + "
for " + recipient;
        stub.forward(workId, _action, newDataItemArray, comment);
    }
}

```

reassignWorkTask

Used to reassign a task from one user to another.

Method Signature

```
void reassignWorkTask(java.lang.String wid, java.lang.String addressee,
java.lang.String comment)
```

Parameters

Parameter	Description
wid	The Id of the task.
addressee	The addressee of the task.
comment	A comment about the task.

Example

```

//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable Active Directory
Account (Mgr Approve-No Timeout)");
//
// Start request
// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap, null);
sleep(5);
//
// Get the process id for this running process
Process process = stub.getProcess(requestId);
if(process != null)
{
    String processId = process.getProcessId();
}

```

```

String initiator = process.getInitiator();
//
// Setup for the query
HashMap map = new HashMap();
map.put(Helper.REQUESTID, requestId);
map.put(Helper.RECIPIENT, recipient);
map.put(Helper.PROCESSID, processId);
map.put(Helper.INITIATOR, initiator);
WorkEntry [] workEntry = workEntryUtils.getWorkEntriesUsingQuery(map,
T_WorkEntryOrder.REQUEST_ID, T_Logic.AND);

if(workEntry == null)
    throw new TestProgramException("Work list is empty.");
//
// Reassign the work entry from recipient to the addressee
//
// Should only be one item
String reassignComment = null;
String workId = workEntry[0].getId();
if(workId != null)
{
    //
    // Reassign work entry(s) to addressee
    reassignComment = "Reassigning work entry " + workId + " from " +
recipient + " to " + addressee;
    stub.reassign(workId, addressee, reassignComment);
    LoggerUtils.sendToLogAndConsole("Reassign work entry " + workId + "
from " + recipient + " to " + addressee);
}
}

```

getWork

Used to retrieve data items for a work entry identified by the Id (UUID) of a task.

Method Signature

```
com.novell.soa.af.impl.soap.DataItemArray getWork(java.lang.String workId)
```

Example

```

//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable Active Directory
Account (Mgr Approve-No Timeout)");
//
// Start request
// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap, null);
sleep(5);
//
// Get the process id for this running process
Process process = stub.getProcess(requestId);
if(process != null)
{

```



```

String processId = process.getProcessId();
String initiator = process.getInitiator();
//
// Setup for the query
HashMap map = new HashMap();
map.put(Helper.REQUESTID, requestId);
map.put(Helper.RECIPIENT, recipient);
map.put(Helper.PROCESSID, processId);
map.put(Helper.INITIATOR, initiator);
WorkEntry [] workEntry = workEntryUtils.getWorkEntriesUsingQuery(map,
T_WorkEntryOrder.REQUEST_ID, T_Logig.AND);
//
// Do assertion here
Assert.assertNotNull("WorkEntry is null for recipient : " + recipient +
" with request id : " + requestId, workEntry);
DataItemArray dataItemArray = stub.getWork(workEntry[0].getId() );
DataItem [] dataItem = dataItemArray.getDataitem();
if(dataItem != null)
    LoggerUtils.sendToLogAndConsole(dataItem[0].getName());
}

```

forwardWithDigitalSignature

Used to forward a provisioning request with a digital signature and optional digital signature properties. For example, this can be used by an administrator to force a user-facing activity to be approved, denied or refused.

Method Signature

```

void forwardWithDigitalSignature(java.lang.String wid,
com.novell.soa.af.impl.soap.T_Action action,
com.novell.soa.af.impl.soap.DataItemArray items, java.lang.String comment,
java.lang.String digitalSignature,
com.novell.soa.af.impl.soap.SignaturePropertyArray digitalSignaturePropertyArray)

```

Parameters

Parameter	Description
wid	The workId.
action	The action to take (approve, deny, refuse).
items	The data items required by the workflow.
comment	A comment about the action.
digitalSignature	The digital signature.
digitalSignaturePropertyArray	The digital signature property map.

Example

```
//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable Active Directory
Account (Mgr Approve-No Timeout)");
//
// Start request
// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap, null);
sleep(5);
//
// Get the process id for this running process
Process process = stub.getProcess(requestId);
String processId = null;
if(process != null)
    processId = process.getProcessId();

T_Action action = T_Action.APPROVE;

T_Logic logic = T_Logic.AND;

T_WorkEntryOrder workEntryOrder = T_WorkEntryOrder.REQUEST_ID;

// Get the digital signature string for admin
String digitalSignature =
DigitalSignatureUtils.getDigitalSignatureFromFile(IDigitalSignatureConstants.ADMIN
_DIGITAL_SIGNATURE_FILENAME);

T_WorkEntryQueryChoice [] workEntryqueryChoice = new
T_WorkEntryQueryChoice[3];
workEntryqueryChoice[0] = new T_WorkEntryQueryChoice();
workEntryqueryChoice[0].setRecipient(recipient);
workEntryqueryChoice[1] = new T_WorkEntryQueryChoice();
workEntryqueryChoice[1].setRequestId(requestId);
workEntryqueryChoice[2] = new T_WorkEntryQueryChoice();
workEntryqueryChoice[2].setProcessId(processId);
//
// Create work entry query
T_WorkEntryQuery query = new T_WorkEntryQuery(logic, _workEntryOrder,
workEntryqueryChoice);
//
// Get all work entries (max records)
WorkEntryArray workEntryArray = stub.getWorkEntries(query, -1);

WorkEntry [] workEntry = workEntryArray.getWorkentry();

if(workEntry != null
{
    for(int wIndex = 0; wIndex < workEntry.length; wIndex++)
    {
        String workId = workEntry[wIndex].getId();
        //
        //
    }
}
```

```

        LoggerUtils.sendToLogAndConsole("Forwarding : " +
workEntry[wIndex].getActivityName() + " work id: " + workId);
        //
        // Get the dataitem for this item of work
        DataItemArray dataItemArray = stub.getWork(workId);
        DataItem [] dataItem = dataItemArray.getDataitem();
        DataItemArray newDataItemArray = null;
        if(dataItem != null)
            // Call method replicateDataItemArray on the
            // provUtils utility object, which refers to a
            // utility class that does not ship with the
            // Identity Manager User Application.
            newDataItemArray = provUtils.replicateDataItemArray(dataItem);
        else
            throw new TestProgramException("DataItem is null.");
        //
        // Claim request for recipient
        String comment = _action.toString() + " this request: " + requestId + "
for " + recipient;
        stub.forwardWithDigitalSignature(workId, _action, newDataItemArray,
comment, digitalSignature, null);
    }
}

```

forwardAsProxy

Used to forward a provisioning request. For example, this can be used by an administrator to force a user-facing activity to be approved, denied or refused.

Method Signature

```
void forwardAsProxy(java.lang.String wid, com.novell.soa.af.impl.soap.T_Action
action, com.novell.soa.af.impl.soap.DataItemArray items, java.lang.String comment,
java.lang.String proxyUser)
```

Parameters

Parameter	Description
wid	The workId (activity Id).
action	The action to take (approve, deny, refuse).
items	The data items required by the workflow.
comment	The comment to add to the activity.
proxyUser	The DN of the proxy user.

Example

```
//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable Active Directory
Account (Mgr Approve-No Timeout)");
//
// Start request
// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap, null);
sleep(5);
//
// Get the process id for this running process
Process process = stub.getProcess(requestId);
String processId = null;
if(process != null)
    processId = process.getProcessId();

T_Action action = T_Action.APPROVE;

T_Logic logic = T_Logic.AND;

T_WorkEntryOrder workEntryOrder = T_WorkEntryOrder.REQUEST_ID;

T_WorkEntryQueryChoice [] workEntryqueryChoice = new
T_WorkEntryQueryChoice[3];
workEntryqueryChoice[0] = new T_WorkEntryQueryChoice();
workEntryqueryChoice[0].setRecipient(recipient);
workEntryqueryChoice[1] = new T_WorkEntryQueryChoice();
workEntryqueryChoice[1].setRequestId(requestId);
workEntryqueryChoice[2] = new T_WorkEntryQueryChoice();
workEntryqueryChoice[2].setProcessId(processId);
//
// Create work entry query
T_WorkEntryQuery query = new T_WorkEntryQuery(logic, _workEntryOrder,
workEntryqueryChoice);
//
// Get all work entries (max records)
WorkEntryArray workEntryArray = stub.getWorkEntries(query, -1);

WorkEntry [] workEntry = workEntryArray.getWorkentry();

if(workEntry != null
{
    for(int wIndex = 0; wIndex < workEntry.length; wIndex++)
    {
        String workId = workEntry[wIndex].getId();
        //
        //
        LoggerUtils.sendToLogAndConsole("Forwarding : " +
workEntry[wIndex].getActivityName() + " work id: " + workId);
        //
        // Get the dataitem for this item of work
        DataItemArray dataItemArray = stub.getWork(workId);
```

```

        DataItem [] dataItem = dataItemArray.getDataitem();
        DataItemArray newDataItemArray = null;
        if(dataItem != null)
            // Call method replicateDataItemArray on the
            // provUtils utility object, which refers to a
            // utility class that does not ship with the
            // Identity Manager User Application.
            newDataItemArray = provUtils.replicateDataItemArray(dataItem);
        else
            throw new TestProgramException("DataItem is null.");
        //
        // Claim request for recipient
        String comment = _action.toString() + " this request: " + requestId + "
for " + recipient;
        String proxyUser =
ServiceUtils.getInstance().getLoginData().getUsername(LoginData.PROXY_TYPE);
        stub.forwardAsProxy(workId, _action, newDataItemArray, comment,
proxyUser);    }
    }
}

```

unclaim

Used to unclaim a provisioning request. This method only works if the request was claimed in the User Application. You cannot unclaim a request once it has been forwarded using the SOAP interface, because the `forward` API method (see [“forward” on page 269](#)) claims and forwards in one operation.

Method Signature

```
void unclaim(java.lang.String wid, java.lang.String comment)
```

Parameters

Parameter	Description
workId	The Id of the activity to unclaim.
comment	A comment about the action.

Example

```

// Action and Approval Types
final int SELECTED_ACTION = 0; final int CLAIMED_SELECTED_ACTION = 0;
T_Action [] action = {T_Action.APPROVE, T_Action.REFUSE, T_Action.DENY};
T_ApprovalStatus [] claimedAction = {T_ApprovalStatus.Approved,
T_ApprovalStatus.Retracted, T_ApprovalStatus.Denied};
//
// Get the process id for this running process
Process process = stub.getProcess(requestId);
String processId = null;
if(process != null)
    processId = process.getProcessId();

HashMap map = new HashMap();
map.put(Helper.REQUESTID, requestId);

```

```

map.put(Helper.RECIPIENT, recipient);
map.put(Helper.PROCESSID, processId);
//
// Claim the request
WorkEntry workEntry = workEntryUtils.claimWorkEntry(map,
action[SELECTED_ACTION]);
if(workEntry != null)
{
    //
    // Now unclaim the entry
    String workId = workEntry.getId();
    stub.unclaim(workId, "Unclaiming this work item : " + workId + " for request
id : " + requestId);
}

```

forwardAsProxyWithDigitalSignature

Used to forward a provisioning request with a digital signature and digital signature properties. For example, this can be used by an administrator to force a user-facing activity to be approved, denied or refused.

Method Signature

```

void forwardAsProxyWithDigitalSignature(java.lang.String wid,
com.novell.soa.af.impl.soap.T_Action action,
com.novell.soa.af.impl.soap.DataItemArray items, java.lang.String comment,
java.lang.String digitalSignature,
com.novell.soa.af.impl.soap.SignaturePropertyArray digitalSignaturePropertyArray,
java.lang.String proxyUser)

```

Parameters

Parameter	Description
wid	The workId (activity Id).
action	The action to take (approve, deny, refuse).
items	The data items required by the workflow.
comment	The comment to add to the activity.
digitalSignature	The digital signature.
digitalSignaturePropertyArray	The digital signature property map.
proxyUser	The DN of the proxy user.

Example

```
//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable Active Directory
Account (Mgr Approve-No Timeout)");
//
// Start request
// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap, null);
sleep(5);
//
// Get the process id for this running process
Process process = stub.getProcess(requestId);
String processId = null;
if(process != null)
    processId = process.getProcessId();

T_Action action = T_Action.APPROVE;

T_Logic logic = T_Logic.AND;

T_WorkEntryOrder workEntryOrder = T_WorkEntryOrder.REQUEST_ID;

T_WorkEntryQueryChoice [] workEntryqueryChoice = new
T_WorkEntryQueryChoice[3];
workEntryqueryChoice[0] = new T_WorkEntryQueryChoice();
workEntryqueryChoice[0].setRecipient(recipient);
workEntryqueryChoice[1] = new T_WorkEntryQueryChoice();
workEntryqueryChoice[1].setRequestId(requestId);
workEntryqueryChoice[2] = new T_WorkEntryQueryChoice();
workEntryqueryChoice[2].setProcessId(processId);
//
// Create work entry query
T_WorkEntryQuery query = new T_WorkEntryQuery(logic, _workEntryOrder,
workEntryqueryChoice);
//
// Get all work entries (max records)
WorkEntryArray workEntryArray = stub.getWorkEntries(query, -1);

WorkEntry [] workEntry = workEntryArray.getWorkentry();

if(workEntry != null
{
    for(int wIndex = 0; wIndex < workEntry.length; wIndex++)
    {
        String workId = workEntry[wIndex].getId();
        //
        //
        LoggerUtils.sendToLogAndConsole("Forwarding : " +
workEntry[wIndex].getActivityName() + " work id: " + workId);
        //
        // Get the dataitem for this item of work
        DataItemArray dataItemArray = stub.getWork(workId);
```

```

DataItem [] dataItem = dataItemArray.getDataitem();
DataItemArray newDataItemArray = null;
if(dataItem != null)
    // Call method replicateDataItemArray on the
    // provUtils utility object, which refers to a
    // utility class that does not ship with the
    // Identity Manager User Application.
    newDataItemArray = provUtils.replicateDataItemArray(dataItem);
else
    throw new TestProgramException("DataItem is null.");
//
// Claim request for recipient
String comment = _action.toString() + " this request: " + requestId + "
for " + recipient;
String digitalSignature =
DigitalSignatureUtils.getDigitalSignatureFromFile(IDigitalSignatureConstants.MMACK
ENZIE_DIGITAL_SIGNATURE_FILENAME);
String proxyUser =
ServiceUtils.getInstance().getLoginData().getUsername(LoginData.PROXY_TYPE);

    stub.forwardAsProxyWithDigitalSignature(workId, _action,
newDataItemArray, comment, digitalSignature, null, proxyUser);
}
}

```

reassign

Used to reassign a task from one user to another.

Method Signature

```
void reassign(java.lang.String wid, java.lang.String addressee, java.lang.String
comment)
```

Parameters

Parameter	Description
wid	The Id of the activity to be reassigned.
addressee	The addressee of the activity.
comment	A comment about the action.

Example

```
//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable Active Directory
Account (Mgr Approve-No Timeout)");
//
// Start request
// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap, null);
sleep(5);
//
// Get the process id for this running process
Process process = stub.getProcess(requestId);
if(process != null)
{
    String processId = process.getProcessId();
    String initiator = process.getInitiator();
    //
    // Setup for the query
    HashMap map = new HashMap();
    map.put(Helper.REQUESTID, requestId);
    map.put(Helper.RECIPIENT, recipient);
    map.put(Helper.PROCESSID, processId);
    map.put(Helper.INITIATOR, initiator);
    WorkEntry [] workEntry = workEntryUtils.getWorkEntriesUsingQuery(map,
T_WorkEntryOrder.REQUEST_ID, T_Logic.AND);

    if(workEntry == null)
        throw new TestProgramException("Work list is empty.");
    //
    // Reassign the work entry from recipient to the addressee
    //
    // Should only be one work item
    String reassignComment = null;
    String workId = workEntry[0].getId();
    if(workId != null)
    {
        //
        // Reassign work entry(s) to addressee
        reassignComment = "Reassigning work entry " + workId + " from " +
recipient + " to " + addressee;
        stub.reassign(workId, addressee, reassignComment);
        LoggerUtils.sendToLogAndConsole("Reassign work entry " + workId + "
from " + recipient + " to " + addressee);
    }
}
}
```

getWorkEntries

Used to query the work entries (activities) and returns a list of `WorkEntry` objects that satisfy the query.

Method Signature

```
com.novell.soa.af.impl.soap.WorkEntryArray  
getWorkEntries(com.novell.soa.af.impl.soap.T_WorkEntryQuery query, int maxRecords)
```

Parameters

Parameter	Description
query	<p>Used to specify the query used to retrieve the list of activities. The query has the following components:</p> <ul style="list-style-type: none">◆ <i>choice</i> - the parameters used to filter the results. You can specify multiple parameters. The possible parameters are: Addressee - Possible values for this parameter are a DN or <i>self</i>. Use <i>self</i> if you want to retrieve work entries for the caller of the query, as identified by the authentication header of the SOAP header. ProcessId RequestId ActivityId Status (an integer) Owner Priority CreationTime (YYYY/MM/DD) ExpTime (YYYY/MM/DD) CompletionTime (YYYY/MM/DD) Recipient Initiator ProxyFor◆ <i>logic</i> - AND or OR◆ <i>order</i> - the order in which to sort the results. Possible values for <i>order</i> are: ACTIVITY_ID RECIPIENT PROVISIONING_TIME RESULT_TIME STATE STATUS REQUEST_ID MESSAGE
maxRecords	<p>Used to specify maximum number of records to retrieve. A value of -1 returns unlimited records.</p>

Example

```
T_Action action = T_Action.APPROVE;

T_Logic logic = T_Logic.AND;

T_WorkEntryOrder workEntryOrder = T_WorkEntryOrder.REQUEST_ID;

T_WorkEntryQueryChoice [] workEntryqueryChoice = new
T_WorkEntryQueryChoice[3];
workEntryqueryChoice[0] = new T_WorkEntryQueryChoice();
workEntryqueryChoice[0].setRecipient(recipient);
workEntryqueryChoice[1] = new T_WorkEntryQueryChoice();
workEntryqueryChoice[1].setRequestId(requestId);
workEntryqueryChoice[2] = new T_WorkEntryQueryChoice();
workEntryqueryChoice[2].setProcessId(processId);
//
// Create work entry query
T_WorkEntryQuery query = new T_WorkEntryQuery(logic, _workEntryOrder,
workEntryqueryChoice);
//
// Get all work entries (max records)
WorkEntryArray workEntryArray = stub.getWorkEntries(query, -1);

WorkEntry [] workEntry = workEntryArray.getWorkentry();
```

getQuorumForWorkTask

Used to get information about the quorum for a workflow activity. A quorum must have actually been specified for the workflow activity by the workflow designer for this method to work.

Method Signature

```
com.novell.soa.af.impl.soap.Quorum getQuorumForWorkTask((java.lang.String workId)
```

Example

```
//

// Note: Provisioning resource must contain a quorum in the flow for this api
method to work

//
// Action and Approval Types
final int SELECTED_ACTION = 0; final int CLAIMED_SELECTED_ACTION = 0;
T_Action [] action = {T_Action.APPROVE, T_Action.REFUSE, T_Action.DENY};
T_ApprovalStatus [] claimedAction = {T_ApprovalStatus.Approved,
T_ApprovalStatus.Retracted, T_ApprovalStatus.Denied};
//
// Get the process id for this running process
Process process = stub.getProcess(requestId);
String processId = null;
if(process != null)
    processId = process.getProcessId();
//
// Setup for the query
HashMap map = new HashMap();
map.put(Helper.REQUESTID, requestId);
map.put(Helper.RECIPIENT, recipient);
```

```

        map.put(Helper.PROCESSID, processId);
        map.put(Helper.INITIATOR, process.getInitiator() );
        WorkEntry [] workEntry =
workEntryUtils.getWorkEntriesUsingQuery(map,
T_WorkEntryOrder.REQUEST_ID, T_Logic.AND);

        Assert.assertNotNull("WorkEntry is null for recipient : " +
recipient + " with request id : " + requestId, workEntry);
        //
        //
        String workId = workEntry[0].getId();

        Quorum quorum = stub.getQuorumForWorkTask(workId);

        Assert.assertNotNull("Quorum for work task is null for recipient :
" + recipient + " with request id : " + requestId, quorum);
        //

        // Extract some data
        int approvalCondition = quorum.getApprovalCondition();
        int status = quorum.getStatus();
        int approveCount = quorum.getApproveCount();
        int participantCount = quorum.getParticipantCount();
        int refuseCount = quorum.getRefuseCount();

```

resetPriorityForWorkTask

Used to reset the priority for a task. You should only use this method on provisioning requests that have a single approval branch.

Method Signature

```
void resetPriorityForWorkTask(java.lang.String workId, int priority,
java.lang.String comment)
```

Parameters

Parameter	Description
workId	The Id of the activity.
priority	The priority to set for the activity.
comment	A comment about the action.

Example

```
// Calls method getProvisioningResourceNameForRecipient
// on the provUtils utility object, which refers to a utility class
// that does not ship with the Identity Manager User Application.
String requestNameToStart =
provUtils.getProvisioningResourceNameForRecipient(recipient, "Enable
Active Directory Account");
    Map map = MapUtils.createAndSetMap(new Object[] {
        Helper.RECIPIENT, recipient,
        IProvisioningConstants.PROVISIONING_REQUEST_TO_START,
requestNameToStart});
    //
    // Try and start the provisioning request
    String requestId =
provWrapper.startProvisioningRequest(recipient, requestNameToStart);
    RationalTestScript.sleep(5);
    //
    // Get the process id for this running process
    Process process = stub.getProcess(requestId);
    if(process != null)
    {
        //
        // Setup for the query
        HashMap map = new HashMap();
        map.put(Helper.REQUESTID, requestId);
        map.put(Helper.RECIPIENT, recipient);
        map.put(Helper.PROCESSID, process.getProcessId());
        map.put(Helper.INITIATOR, process.getInitiator());
        WorkEntry [] workEntry =
workEntryUtils.getWorkEntriesUsingQuery(map,
T_WorkEntryOrder.REQUEST_ID, T_Logic.AND);
        //
        // Now reset the priority for this work item.
        String workId = workEntry[0].getId();
        String comment = "Resetting priority for this work item.";
        int priority = 0;
        stub.resetPriorityForWorkTask(workId, priority, comment);
    }
}
```

11.3.4 Comments

This section provides reference information for each Comments method.

getCommentsByType

Used to get workflow comments that are of a specific type (for example, user, system).

Method Signature

```
com.novell.soa.af.impl.soap.CommentArray getCommentsByType(java.lang.String
requestId, com.novell.soa.af.impl.soap.T_CommentType type)
```

Parameters

Parameter	Description
requestId	The process identifier.
type	The comment type (USER or SYSTEM)

Example

```
//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable
Active Directory Account (Mgr Approve-No Timeout)");
//
// Start request
// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap,
null);
sleep(5);
//
// Get the comments by type : either User or System
T_CommentType [] commentTypes = {T_CommentType.User,
T_CommentType.System};

for(int types = 0; types < commentTypes.length; types++)
{
    CommentArray commentArray = stub.getCommentsByType(requestId,
commentTypes[types]);
    Comment [] comments = commentArray.getComment();
    if(comments != null)
    {
        for(int index = 0; index < comments.length; index++)
        {
            LoggerUtils.sendToLogAndConsole(" \nComment Type = " +
commentTypes[types].getValue() + "\n" +
                "Activity Id: " +
comments[index].getActivityId() + "\n" +
                "Comment : " + comments[index].getComment()
+ "\n" +
                "User : " + comments[index].getUser() + "\n"
+
                "System comment : " +
comments[index].getSystemComment() + "\n" +
                "Time stamp : " +
comments[index].getTimestamp().getTime().toString() );
        }
    }
}
```

getCommentsByActivity

Used to get the comments for a specific activity.

Method Signature

```
com.novell.soa.af.impl.soap.CommentArray getCommentsByActivity(java.lang.String  
requestId, java.lang.String aid)
```

Parameters

Parameter	Description
requestId	The process identifier.
aid	The activity identifier.

Example

```
//  
// Initialize and start a provisioning request  
HashMap provMap = new HashMap();  
provMap.put(Helper.RECIPIENT, recipient);  
provMap.put("Provisioning_Request_To_Start_Key", "Enable  
Active Directory Account (Mgr Approve-No Timeout)");  
//  
// Start request  
// Calls method startProvisioningRequest on the provUtils  
// utility object which refers to a utility class that does not  
// ship with the Identity Manager User Application.  
String requestId = provUtils.startProvisioningRequest(provMap,  
null);  
sleep(5);  
//  
// Get the process id for this running process  
Process process = stub.getProcess(requestId);  
if(process != null)  
{  
    String processId = process.getProcessId();  
    String initiator = process.getInitiator();  
    //  
    // Setup for the query  
    HashMap map = new HashMap();  
    map.put(Helper.REQUESTID, requestId);  
    map.put(Helper.RECIPIENT, recipient);  
    map.put(Helper.PROCESSID, processId);  
    map.put(Helper.INITIATOR, initiator);  
    WorkEntry [] workEntry =  
workEntryUtils.getWorkEntriesUsingQuery(map,  
T_WorkEntryOrder.REQUEST_ID, T_Logic.AND);  
    //  
    // Get the activity id associated with the item of work  
    String activityId = workEntry[0].getActivityId();  
    //  
    // Get the comments based on activity  
    if(activityId != null)  
    {  
        CommentArray commentArray =  
stub.getCommentsByActivity(requestId, activityId);  
        Comment [] comments = commentArray.getComment();  
    }  
}  
}
```

getCommentsByUser

Used to get the comments made by a specific user.

Method Signature

```
com.novell.soa.af.impl.soap.CommentArray getCommentsByUser(java.lang.String  
requestId, java.lang.String user)
```

Parameters

Parameter	Description
requestId	The process identifier.
user	The the DN of the user (recipient) who created the comments

Example

```
//  
// Initialize and start a provisioning request  
HashMap provMap = new HashMap();  
provMap.put(Helper.RECIPIENT, recipient);  
provMap.put("Provisioning_Request_To_Start_Key", "Enable  
Active Directory Account (Mgr Approve-No Timeout)");  
//  
// Start request  
// Calls method startProvisioningRequest on the provUtils  
// utility object which refers to a utility class that does not  
// ship with the Identity Manager User Application.  
String requestId = provUtils.startProvisioningRequest(provMap,  
null);  
sleep(5);  
//  
// Get the comments by recipient (should be the same as user)  
CommentArray commentArray = stub.getCommentsByUser(requestId,  
recipient);  
Comment [] comments = commentArray.getComment();
```

getCommentsByCreationTime

Used to get comments made at a specific time.

Method Signature

```
com.novell.soa.af.impl.soap.CommentArray  
getCommentsByCreationTime(java.lang.String requestId, long time,  
com.novell.soa.af.impl.soap.T_Operator op)
```


Parameters

Parameter	Description
requestId	The process identifier.
time	The time stamp.
op	The query operator to use. Possible values for operator are: EQ - equals LT - less than LE - less than or equal to GT - greater than GE - greater than or equal to

Example

```
//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable
Active Directory Account (Mgr Approve-No Timeout)");
//
// Start request
// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap,
null);
sleep(5);
//
// Get comments by creation time for the provisioning request
started above.
long currentTime = System.currentTimeMillis();
LoggerUtils.sendToLogAndConsole("-->Current date = " + new
java.util.Date(currentTime).toString() );
//
//
T_Operator operator = T_Operator.GT;
CommentArray commentArray =
stub.getCommentsByCreationTime(requestId, currentTime, operator);
Comment [] comments = commentArray.getComment();
```

addComment

Used to add a comment to a workflow activity.

Method Signature

```
void addComment(java.lang.String workId, java.lang.String comment)
```

Parameters

Parameter	Description
workId	The activity identifier (UUID).
comment	A comment about the activity.

Example

```
//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable Active Directory
Account (Mgr Approve-No Timeout)");
//
// Start request

// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.

String requestId = provUtils.startProvisioningRequest(provMap, null);
sleep(5);
//
// Setup for the query
HashMap map = new HashMap();
map.put(Helper.REQUESTID, requestId);
map.put(Helper.RECIPIENT, recipient);
WorkEntry [] workEntry = workEntryUtils.getWorkEntriesUsingQuery(map,
T_WorkEntryOrder.REQUEST_ID, T_Logic.AND);
//
// Add comment to the work entry
String workId = workEntry[0].getId();
String processId = workEntry[0].getProcessId();
String addComment = "Test comment for work id " + workId;
stub.addComment(workId, addComment);
sleep(2);
```

getComments

Used to get comments from a workflow.

Method Signature

```
com.novell.soa.af.impl.soap.CommentArray getComments(java.lang.String workId, int
maxRecords)
```

Parameters

Parameter	Description
workId	The activity Id (UUID).
maxRecords	An integer specifying the maximum number of records to retrieve.

Example

```
//  
// Setup for the query  
HashMap map = new HashMap();  
map.put(Helper.RECIPIENT, addressee);  
WorkEntry [] workEntry =  
workEntryUtils.getWorkEntriesUsingQuery(map,  
T_WorkEntryOrder.ADDRESSEE, T_Logic.OR);  
//  
// Get all the comment records for this workId  
int maxRecords = -1;  
CommentArray commentArray = stub.getComments(workId, maxRecords);  
Comment [] comment = commentArray.getComment();
```

11.3.5 Configuration

This section provides reference information for each Configuration method.

setCompletedProcessTimeout

Used to set the timeout for completed processes. Processes that were completed more than timeout days ago are removed from the system. The default value is 120 days. The valid range is 0 days to 365 days.

Method Signature

```
void setCompletedProcessTimeout(int time)
```

Example

```
accessConfigurationSettings(SET_COMPLETED_PROCESS_TIMEOUT, new Integer(212) );
```

setEngineConfiguration

Used to set workflow engine configuration parameters.

Method Signature

```
void setEngineConfiguration(com.novell.soa.af.impl.soap.Configuration config)
```

Parameters

Parameter	Description
minPoolSize	The minumum thread pool size.
maxnPoolSize	The maximum thread pool size.
initialPoolSize	The initial thread pool size.
keepAliveTime	Thread pool keep live time.
pendingInterval	The cluster synchronization time.

Parameter	Description
cleanupInterval	The interval between purging processes from databases.
retryQueueInterval	The interval between retrying failed processes.
maxShutdownTime	The maximum time to let threads complete work before engine shutdown.
userActivityTimeout	The default user activity timeout.
completedProcessTimeout	The default completed process timeout.
webServiceActivityTimeout	The default Web service activity timeout.
emailNotification	Turns email notification on or off.
processCacheInitialCapacity	The process cache initial capacity.
processCacheMaxCapacity	The process cache maximum capacity.
processCacheLoadFactor	The process cache load factor.
heartbeatInterval	The heartbeat interval.
heartbeatFactor	The heartbeat factor.

Example

```
accessConfigurationSettings(SET_ENGINE_CONFIGURATION, new Integer(313) );
```

getCompletedProcessTimeout

Used to get the timeout for completed processes.

Method Signature

```
int getCompletedProcessTimeout()
```

Example

```
accessConfigurationSettings(GET_COMPLETED_PROCESS_TIMEOUT, new Integer(121) );
```

setEmailNotifications

Used to globally enable or disable email notifications.

Method Signature

```
void setEmailNotifications(boolean enable)
```

Parameters

Parameter	Description
enable	Email notifications are enabled if true; otherwise they are disabled.

Example

```
accessConfigurationSettings(SET_EMAIL_NOTIFICATIONS, new Boolean(false) );
```

clearNIMCaches

Clear the NetIQ Integration Manager (previously named exteNd Composer) caches.

Method Signature

```
void clearNIMCaches()
```

Example

```
accessConfigurationSettings(CLEAR_NIM_CACHES, new Object() );
```

setWebServiceActivityTimeout

Used to set the timeout for Web service activities. The default value is 50 minutes. The valid range is 1 minute to 7 days.

Method Signature

```
void setWebServiceActivityTimeout(int time)
```

Parameters

Parameter	Description
time	The timeout value in minutes.

Example

```
accessConfigurationSettings(SET_WEBSERVICE_ACTIVITY_TIMEOUT, new Integer(767) );
```

getUserActivityTimeout

Used to get the timeout for user-facing activities.

Method Signature

```
int getUserActivityTimeout()
```

Example

```
accessConfigurationSettings(GET_USER_ACTIVITY_TIMEOUT, new Integer(3767) );
```

getEmailNotifications

Used to determine if global email notifications are enabled or disabled.

Method Signature

```
boolean getEmailNotifications()
```

Example

```
accessConfigurationSettings(GET_EMAIL_NOTIFICATIONS, new Boolean(true) );
```

setUserActivityTimeout

Used to set the timeout for user-facing activities. The default value is no timeout (a value of zero). The valid range is 1 hour to 365 days.

Method Signature

```
void setUserActivityTimeout(int time)
```

Parameters

Parameter	Description
time	The timeout value in hours.

Example

```
accessConfigurationSettings(SET_USER_ACTIVITY_TIMEOUT, new Integer(1767) );
```

getEngineConfiguration

Used to get the workflow engine configuration parameters.

Method Signature

```
com.novell.soa.af.impl.soap.Configuration getEngineConfiguration()
```

Example

```
accessConfigurationSettings(GET_ENGINE_CONFIGURATION, new Integer(141) );
```

getWebServiceActivityTimeout

Used to get the timeout for Web service activities.

Method Signature

```
int getWebServiceActivityTimeout()
```

Example

```
accessConfigurationSettings(GET_WEBSERVICE_ACTIVITY_TIMEOUT, new Integer(808) );
```

11.3.6 Miscellaneous

This section provides reference information for each Miscellaneous method.

getGraph

Used to get a JPG image of the workflow. The Graphviz program must be installed on the computer where the application server and the IDM User Application is running. For more information about Graphviz, see [Graphviz \(http://www.graphviz.org\)](http://www.graphviz.org).

Method Signature

```
byte[] getGraph(java.lang.String processId)
```

Parameters

Parameters	Description
processId	The request Id.

Example

```
//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable Active
Directory Account (Mgr Approve-No Timeout)");
//
// Start request
// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap,
null);
sleep(5);
//

//
Process process = stub.getProcess(requestId);
if(process != null)
{
    byte [] graph = null;
    if( (recipient.compareTo(process.getRecipient()) == 0) &&
(requestId.compareTo(process.getRequestId()) == 0) )
    {
        graph = stub.getGraph(process.getProcessId() );
    }
    //
    // Do assert
    Assert.assertNotNull("Graph is null.", graph);
}
}
```

getFlowDefinition

Used to get the XML for a provisioning request.

Method Signature

```
java.lang.String getFlowDefinition(java.lang.String processId)
```

Parameters

Parameters	Description
processId	The request Id.

Example

```
//  
    // Initialize and start a provisioning request  
    HashMap provMap = new HashMap();  
    provMap.put(Helper.RECIPIENT, recipient);  
    provMap.put("Provisioning_Request_To_Start_Key", "Enable Active  
Directory Account (Mgr Approve-No Timeout)");  
    //  
    // Start request  
    // Calls method startProvisioningRequest on the provUtils  
    // utility object which refers to a utility class that does not  
    // ship with the Identity Manager User Application.  
    String requestId = provUtils.startProvisioningRequest(provMap, null);  
    sleep(5);  
    //  
    //  
    Process process = stub.getProcess(requestId);  
    if(process != null)  
    {  
        String XMLFlowDefinition = null;  
        if( (recipient.compareTo(process.getRecipient()) == 0) &&  
(requestId.compareTo(process.getRequestId()) == 0) )  
        {  
            XMLFlowDefinition = stub.getFlowDefinition(process.getProcessId() );  
        }  
        //  
        // Do assert  
        Assert.assertNotNull("Flow Definition is null.", XMLFlowDefinition);  
    }  
}
```

getFormDefinition

Used to get the XML for a form for a provisioning request.

Method Signature

```
java.lang.String getFormDefinition(java.lang.String processId)
```


Parameters

Parameters	Description
processId	The request Id.

Example

```
//
// Initialize and start a provisioning request
HashMap provMap = new HashMap();
provMap.put(Helper.RECIPIENT, recipient);
provMap.put("Provisioning_Request_To_Start_Key", "Enable Active
Directory Account (Mgr Approve-No Timeout)");
//
// Start request
// Calls method startProvisioningRequest on the provUtils
// utility object which refers to a utility class that does not
// ship with the Identity Manager User Application.
String requestId = provUtils.startProvisioningRequest(provMap, null);
sleep(5);
//

//
Process process = stub.getProcess(requestId);
if(process != null)
{

    String XMLFormDefinition = null;
    if( (recipient.compareTo(process.getRecipient()) == 0) &&
(requestId.compareTo(process.getRequestId()) == 0) )
    {
        XMLFormDefinition =
stub.getFormDefinition(process.getProcessId() );
    }
    //
    // Do assert
    Assert.assertNotNull("Form Definition is null.",
XMLFormDefinition);
}
```

getVersion

Used to get the version of the workflow system.

Method Signature

```
com.novell.soa.af.impl.soap.T_Version getVersion()
```

Example

```
StringBuffer result = new StringBuffer();

    T_Version version = stub.getVersion();
    if (version != null)
    {
        result.append(" Major = " + version.getMajor() );
        result.append(" Minor = " + version.getMinor() );
        result.append(" Revision = " + version.getRevision() );

        System.out.println("Version Information " + result.toString());
    }
```

11.3.7 Cluster

This section provides reference information for each Cluster method.

getEngineState

Used to get the `IEngineState` for a workflow engine, specified by engine Id.

Method Signature

```
com.novell.soa.af.impl.soap.EngineState getEngineState(java.lang.String engineId)
```

Parameters

Parameter	Description
engineId	The Id of the workflow engine.

Example

```
EngineStateArray engineStateArray = stub.getClusterState();
EngineState [] engineState = engineStateArray.getEngineStates();
if(engineState != null)
{
    LoggerUtils.sendToLogAndConsole("EngineCount in cluster:" +
engineState.length);
    for(int index = 0; index < engineState.length; index++)
    {
        EngineState engine =
stub.getEngineState(engineState[index].getEngineId() );
        LoggerUtils.sendToLogAndConsole(
            "Engine Id: " + engine.getEngineId() + "\n" +
            "Engine status: " + engine.getEngineStatus() + "\n" +
            "Value of engine status: " +
engine.getValueOfEngineStatus() + "\n" +
            "Heartbeat: " + ( (engine.getHeartbeat() != null) ?
engine.getHeartbeat().getTime().toString() : "null") + "\n" +
            "Shutdown time: " + ((engine.getShutdownTime() != null)
? engine.getShutdownTime().getTime().toString() : "null") + "\n" +
            "Start time: " + ((engine.getStartTime() != null) ?
engine.getStartTime().getTime().toString() : "null") );
        }
    }
}
```

reassignAllProcesses

Used to reassign all processes from the source engine to a list of target engines.

Method Signature

```
int reassignAllProcesses(java.lang.String sourceEngineId,
com.novell.soa.af.impl.soap.StringArray targetEngineIds)
```

Parameters

Parameter	Description
sourceEngineId	The Id of the source workflow engine.
targetEngineIds	The Ids of the target workflow engines.

getEngineState

Used to get a list that contains an IEngineState object for each engine in the cluster.

Method Signature

```
public com.novell.soa.af.impl.soap.EngineState getEngineState(java.lang.String
engineId)
```

Parameters

Parameter	Description
engineId	The Id of the workflow engine.

Example

```
EngineStateArray engineStateArray = stub.getClusterState();
EngineState [] engineState = engineStateArray.getEngineStates();
if(engineState != null)
{
    LoggerUtils.sendToLogAndConsole("EngineCount in cluster:" +
engineState.length);
    for(int index = 0; index < engineState.length; index++)
    {
        EngineState engine =
stub.getEngineState(engineState[index].getEngineId() );
        LoggerUtils.sendToLogAndConsole(
            "Engine Id: " + engine.getEngineId() + "\n" +
            "Engine status: " + engine.getEngineStatus() + "\n" +
            "Value of engine status: " +
engine.getValueOfEngineStatus() + "\n" +
            "Heartbeat: " + ( (engine.getHeartbeat() != null) ?
engine.getHeartbeat().getTime().toString() : "null") + "\n" +
            "Shutdown time: " + ((engine.getShutdownTime() != null)
? engine.getShutdownTime().getTime().toString() : "null") + "\n" +
            "Start time: " + ((engine.getStartTime() != null) ?
engine.getStartTime().getTime().toString() : "null") );
    }
}
```

reassignPercentageProcesses

Used to reassign a percentage of processes from the source engine to the target engine.

Method Signature

```
int reassignPercentageProcesses(int percent, java.lang.String sourceEngineId,
java.lang.String targetEngineId)
```

Parameters

Parameter	Description
percent	An integer representing the percentage of processes to be reassigned.
sourceEngineId	The Id of the source workflow engine.
targetEngineIds	The Id of the target workflow engine.

reassignProcesses

Used to reassign one or more processes from the source engine to the target engine.

Method Signature

```
int reassignProcesses(com.novell.soa.af.impl.soap.StringArray requestIds,  
java.lang.String sourceEngineId, java.lang.String targetEngineId)
```

Parameters

Parameter	Description
requestIds	A list of requestIds of the processes to be reassigned.
sourceEngineId	The Id of the source workflow engine.
targetEngineIds	The Id of the target workflow engine.

removeEngine

Used to remove an engine from the cluster state table. The engine must be in the SHUTDOWN or TIMEDOUT state.

Method Signature

```
void removeEngine(java.lang.String engineId)
```

Parameters

Parameter	Description
engineId	The Id of the workflow engine to be removed.

12 Metrics Web Service

This section describes the Metrics Web Service, which provides metrics for provisioning workflows.

12.1 About the Metrics Web Service

The workflow engine includes a Web Service for gathering workflow metrics. The addition of the Metrics Web Service to the workflow engine lets you monitor an approval flow process. In addition, it provides indicators the business manager can use to modify the process for optimal performance.

The metrics are based on traditional business process flow management principles, which emphasize the need for metrics to be actionable. This ensures that the metrics provided match what an operations manager usually looks for when analyzing and optimizing business flows. Therefore, the metrics identify bottlenecks and provide other capacity indicators. The Metrics Web Service allows you to narrow down the metrics to a common and established set of data, instead of trying to anticipate the myriad of metrics and reports that can be created for a business process flow.

When working with the Metrics Web Service, you should keep in mind that the service is not intended to be an all-purpose metrics system:

- ♦ The Metrics Web Service is not a reporting tool or reporting engine. Consequently it does not use a complex query language.
- ♦ The Metrics Web Service is not designed as an all-purpose performance management system. This helps to limit the impact of the needed queries against the live system being monitored.

Operations management stresses three key internal process performance measures that together capture the essence of process flow. These three measures can serve as leading indicators of customer satisfaction: flow time, flow rate, and inventory.

With these measures, an operations manager can answer the following questions:

- ♦ On average, how much time does a provisioning request spend within the process boundaries? (Flow time)
- ♦ On average, how many provisioning requests pass through the process per unit of time? (Flow rate)
- ♦ On average, how many provisioning requests are within the process boundaries at any point in time? (Inventory)

These three measures are related by Little's law:

`Inventory=Flow Rate*Flow Time`

12.1.1 Web Service Semantics

The following semantics apply to the use of the Metrics Web Service:

- ♦ Activities in the Metrics Web Service refer only to user-facing activities (Approval Activities). Negligible running time and the impossibility of controlling the other activities make collecting metrics for these inappropriate.
- ♦ The Metrics Web Service distinguishes between Working Days and Calendar Days. Calendar Days refer to all days between two dates. Working Days refer only to working days between two dates. Since working days may be specified differently in different environments, all Working Days methods return a raw data set that can be used to compute what is appropriate. If no such detail is required, the Calendar Days method will readily return the appropriate metric.

12.1.2 Accessing the Test Page

The Metrics Web Service endpoint can be accessed at the following URL:

```
http://server:port/warcontext/metrics/service
```

You can also access the SOAP endpoint by going to the **Administration** within the User Application. To do this, you need to select the **Application Configuration** tab, then select **Web Services** from the left-navigation menu. After selecting **Web Services**, pick the Web Service endpoint you want from the list.

WARNING: The test page is disabled by default. Since some of the methods allow data to be updated, the test page presents a potential security vulnerability and should not be allowed in a production environment. For details on enabling the test page, see the instructions provided for the Role Service in [“Enabling the Test Page” on page 359](#).

12.1.3 Web Service Methods Grouped by Security Permissions

The service is secured using Basic Authentication. Therefore, you should use SSL to connect to the service. The service uses the same security layer as the User Application and consequently not all service operations are allowed to all users. Only a Provisioning Administrator will have unconditional access to all the methods. On the other hand team managers will only have access to metrics that pertain to their team and team members.

Hence the Metrics Web Service operations are divided into 3 categories according to role and security permissions:

- ♦ Team manager operations
- ♦ Provisioning Application Administrator operations
- ♦ Utility operations

Team Metrics

Team managers can only retrieve metrics on a team for which they are managers. These are the methods available to team managers:

Table 12-1 Team Metrics Methods

Method	Description
getClaimedFlowTimeCalendarDays	Returns the average time in hours the provisioning request was claimed for within the specified time interval
getClaimedFlowTimeWorkingDays	Returns the result set required to compute the average time the provisioning request was claimed for the specified time interval
getToClaimedFlowTimeCalendarDays	Returns the average time in hours it took the provisioning request to be claimed from the moment it was available to addressees
getToClaimedFlowTimeWorkingDays	Returns the average time it took the provisioning request to be claimed from the moment it was available to addressees, within the specified time interval
getClaimedInventory	Returns the average number of provisioning requests claimed within the specified interval
getClaimedThroughputWorkingDays	Returns the average number of provisioning requests claimed within the specified interval
getTeamLongestRunning	Returns a result set of the longest running request in seconds for which members of the team acted as addressees
getTeamFlowHistory	Returns a result set of the activity outcomes, addressee and addressee messages for the specified list of provisioning requests
getTeamHistoryForInitiators	Returns a result set of the provisioning request and their status for which members of the team acted as initiators
getTeamHistoryForRecipients	Returns a result set of the provisioning request and their status for which members of the team acted as recipients
getTeamRunningTime	Returns the average time in seconds the specified provisioning requests have been running
getTeamDecisionCount	Returns the number of decisions the team made as addressees for the specified provisioning request
getTeamInitiatedCount	Returns the number of provisioning requests initiated by the team
getTeamRecipientCount	Returns the provisioning requests for which a member of the team acts as a recipient

Provisioning Administrator Metrics

This role is unrestricted and may perform any of the service's operations. These are the methods that are only available to Provisioning Administrators.

Table 12-2 Provisioning Administrator Metrics Methods

Method	Description
getActivityFlowTimeCalendarDays	Returns the average time in hours the user activity took to complete
getActivityFlowTimeWorkingDays	Returns the result set required to compute the average time the user activity took to complete
getActivityInventory	Returns the average number of provisioning requests at any one time for the specified user activity
getActivityThroughputCalendarDays	Returns the average number of provisioning requests per hours that exited the specified user activity within the specified time interval
getActivityThroughputWorkingDays	Returns the result set required to compute average time it takes a provisioning request to complete for the specified time interval
getFlowTimeCalendarDays	Returns average time in hours it takes a provisioning request to complete for the specified time interval
getFlowTimeWorkingDays	Returns the result set required to compute average time it takes a provisioning request to complete for the specified time interval
getInventory	Returns the average number of provisioning requests in the system at any one time for the specified time interval
getLongestClaimed	Returns a result set of the provisioning requests that have been claimed but not acted upon (time in seconds)
getLongestRunning	Returns a result set of the longest running provisioning requests (time in seconds)
getFlowCount	Returns the number of provisioning requests
getFlowHistory	Returns a result set of the activity outcomes, addressee and addressee messages for the specified list of provisioning requests
getFlowHistoryForInitiators	Returns the list of provisioning requests and their status for the specified initiators
getFlowHistoryForRecipients	Returns the list of provisioning requests and their status for the specified recipients
getRunningTime	Returns the average running time in seconds for the provisioning requests that are currently running
getThroughputCalendarDays	Returns the average number of provisioning requests per hour that completed within the specified interval
getThroughputWorkingDays	Returns the result set required to compute the average number per hour of provisioning requests that completed within the specified interval

Utility Operations

Both team managers and administrators may perform these operations:

Table 12-3 Utility Operations

Method	Description
getVersion	Returns the server version of the Web service. This should always be used to ensure version matching between client and server code.
getAllProvisioningFlows	Returns the list of provisioning flows that the logged in user can see
getUserActivityOnlyFlow	Returns a GraphViz DOT (http://www.graphviz.org/) representation of the provisioning workflow
getTeams	Returns the list of teams the logged in user manages
getTeamMembers	Returns the list of team members for the specified team

12.1.4 Specifying Filters

As mentioned above, the Metrics Webservice does not use a complex query language. Instead filters can be used to narrow results by criteria such as date ranges or approval statuses.

These are the filters you can specify (see type `FilterConstants` in service's WSDL):

Table 12-4 Filters for Narrowing Metric Results

Filter	Description
KEY_ACTIVITY_ID	A User Activity Id as defined in the provisioning request definition
KEY_APPROVAL_STATUS	The approval status for the provisioning request. Possible values are: <ul style="list-style-type: none">◆ <code>ApprovalStatusProcessing</code>◆ <code>ApprovalStatusDenied</code>◆ <code>ApprovalStatusRefused</code>◆ <code>ApprovalStatusApproved</code>◆ <code>ApprovalStatusRetract</code>◆ <code>ApprovalStatusError</code>
KEY_ENTITLEMENT_STATE	The state of the entitlement associated with the provisioning request. Possible values are: <ul style="list-style-type: none">◆ <code>EntitlementUnknown</code>◆ <code>EntitlementGranted</code>◆ <code>EntitlementRevoked</code>

Filter	Description
KEY_ENTITLEMENT_STATUS	The status of the entitlement associated with the provisioning request. Possible values are: <ul style="list-style-type: none"> ◆ EntitlementSuccess ◆ EntitlementWarning ◆ EntitlementError ◆ EntitlementFatal
KEY_INITIATOR	The user DN of the workflow initiator
KEY_L_COMPLETION_TIME	The date indicating the start of the interval for workflow completion
KEY_S_COMPLETION_TIME	The date indicating the end of the interval for workflow completion
KEY_L_ENTITLEMENT_TIME	The date indicating the start of the interval for entitlement time
KEY_S_ENTITLEMENT_TIME	The date indicating the end of the interval for entitlement time
KEY_S_START_TIME	The date indicating the start of the interval for workflow start
KEY_L_START_TIME	The date indicating the end of the interval for workflow start
KEY_PROCESS_ID	The DN of the provisioning request
KEY_PROCESS_STATUS	The status of the provisioning request. Possible values are: <ul style="list-style-type: none"> ◆ ProcessStatusRunning ◆ ProcessStatusStopped ◆ ProcessStatusTerminated ◆ ProcessStatusCompleted
KEY_PROCESS_VERSION	The process version associated with the workflow version
KEY_RECIPIENT	The user DN of the workflow recipient
KEY_REQUEST_ID	The unique id associated with the workflow instance

Here is a Java example. Note that your code will obviously differ depending on the platform you use for your Web Service client:

```

HashMap map=new HashMap();

map.put(MetricsFilter.KEY_PROCESS_STATUS,
MetricsFilter.ProcessStatusRunning);

double flowtime = metrics.getFlowTimeCalendarDays(processId,
processVer, activity, 5, calendar1.getTime(),
calendar2.getTime(), MetricsFilter.ACTIVITY_CLAIMED,

```

```
MetricsFilter.ACTIVITY_FORWARDED, map);
```

```
...
```

Please consult the WebService WSDL for more information:

```
http://server:port/warcontext/metrics/service?WSDL
```

12.1.5 Generating the Stub Classes

Before using the Web Service, you need to use the WSSDK tool or another SOAP tool kit to generate the stub classes. To allow your code to find the stub classes, you also need to add the JAR that contains the stub classes to your classpath.

If you want to use the NetIQ WSSDK tool, you can generate the client stubs by extracting the WSDL and running the `wsdl2java` utility. For example, you could run this command to generate the stubs in a package called `com.novell.soa.af.metrics.soap.impl`:

```
"C:\Program Files\Java\jdk1.6.0_31\bin\java" -cp "../lib/wssdk.jar;../lib/jaxrpc-api.jar";"../lib/mail.jar";"../lib/activation.jar";"c:\Program Files\Java\jdk1.6.0_31\lib\tools.jar"; com.novell.soa.ws.impl.tools.wsdl2java.Main -verbose -ds gensrc -d C:\ -noskel -notie -genclient -keep -package com.novell.soa.af.metrics.soap.impl -javadoc metrics.wsdl
```

You can change the `wsdl2java` parameters to suit your requirements.

12.1.6 Obtaining the Remote Interface

Before you can begin calling methods on the Metrics Web Service, you need to have a reference to the remote interface.

The code below shows how to obtain the remote interface.

```
import java.util.Locale;
import java.util.Properties;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.xml.rpc.Stub;
import com.novell.qa.soap.common.util.LoggerUtils;
import com.novell.qa.soap.common.util.LoginData;
import com.novell.qa.soap.common.util.ServiceUtils;
import com.novell.soa.af.ClusterException;
import com.novell.soa.af.impl.soap.Provisioning;
import com.novell.soa.af.impl.soap.ProvisioningService;
import com.novell.test.automator.framework.TestProgramException;
import com.rational.test.ft.script.RationalTestScript;
import com.novell.soa.af.metrics.soap.MetricsClientHelper;
import com.novell.soa.af.metrics.soap.MetricsStubWrapper;
import com.novell.soa.af.metrics.soap.impl.MetricsService;
import com.novell.soa.af.metrics.soap.impl.MetricsServiceException;
import com.novell.soa.af.metrics.soap.impl.IRemoteMetrics;

/**
 * Method to obtain the remote interface to the Metrics endpoint
 * @param _url
 * @param _username
 * @param _password
 * @return IRemoteMetrics interface
```

```

* @throws Exception
*/
private IRemoteMetrics getStub(String _url, String _username, String _password)
throws Exception

{
    Properties properties = new Properties();
    properties.put(Context.INITIAL_CONTEXT_FACTORY,
"org.jnp.interfaces.NamingContextFactory");

    String lookup =
"xmlrpc:soap:com.novell.soa.af.metrics.soap.impl.MetricsService";

    InitialContext ctx = new InitialContext();
    MetricsService svc = (MetricsService) ctx.lookup(lookup);

    Stub stub = (Stub)svc.getIRemoteMetricsPort();

    stub._setProperty(Stub.USERNAME_PROPERTY, _username);
    stub._setProperty(Stub.PASSWORD_PROPERTY, _password);
    stub._setProperty(Stub.SESSION_MAINTAIN_PROPERTY, Boolean.TRUE);
    stub._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY, _url);

    return (IRemoteMetrics) stub;
}

```

Here's the code to call the method defined above:

```

IRemoteMetrics stub = null;
    try
    {
        //
        // Get the stub
        String url = m_loginData.getURL();
        stub = getStub(url, _username, _password);
    }
    catch(Exception e)
    {
        String msg = e.getMessage();
        LoggerUtils.logError(msg);
        throw new TestProgramException(msg);
    }
    return stub;

```

In order for this code to work, the URL passed to the getStub() method would need to point to the SOAP endpoint, as shown below:

```
http://myserver:8080/IDMProv/metrics/service
```

The user name needs to be a fully qualified DN such as the following:

```
"cn=admin,ou=idmsample,o=netiq"
```

12.1.7 Metrics Configuration Settings

The Metrics Web Service impact on the live system is limited by 4 settings that may be modified in the `IDMfw.jar/WorkflowService-conf/config.xml` file:

Table 12-5 Metrics Configuration Settings

Key in config.xml	Description
<key>Metrics/TimeRequiredBetweenClientRequests</key>	Required time between client requests in ms (default is 250 ms)
<key>Metrics/MaxClients</key>	Maximum number of concurrent client sessions (default is 10)
<key>Metrics/MaxRows</key>	Maximum number of rows any query can return
<key>Metrics/MaxTeamMembers</key>	Maximum Number of Team Members
<key>Metrics/SecondsToAnythingDivider</key>	The divider used in all throughput computations (default 3600). Original values are in seconds so all throughputs are consequently per hour.

When the limit has been reached for any of these settings a Web Service fault is generated indicating the problem. In addition, for settings 1 and 2, the fault includes an error code.

- ♦ If the fault is caused by a TimeRequiredBetweenClientRequests error, the error code is 100.
- ♦ If the fault is caused by a MaxClients errors, the error code is 200.
- ♦ If the fault is caused by a closed client connection error, the error code is 300.

Client consumers of the Metrics Web Service will have to include in their code provisions for retrying a request. Here is a simple Java listing that shows how this can be achieved:

```
try {
    for (int i = 0; i < retries; i++) {
        try {
            return metrics.getFlowCount(strDN, strId, new
HashMap());
        } catch (MetricsServiceException e) {
            if (e.getErrorCode() == 100 //subsequent call
error
                || e.getErrorCode() == 200) { //too many
clients
                    try {
                        Thread.sleep(retryPause);
                    }
                }
            catch (Exception ex) {
                // to nothing
            }
        } else {
            throw e2;
        }
    }
}
```

```

        } else {
            throw new RuntimeException(e);
        }
    } catch (Exception e) {
        throw e;
    }
}
    throw new RuntimeException("Did not succeed making
webservice call");
} catch (Exception e) {
    throw e;
}
}
...

```

12.2 Metrics Web Service API

This section provides details about the methods available with the Metrics web service.

All of the methods throw `MetricsServiceException` and `RemoteException`. To improve readability, the throws clause has been omitted from the method signatures.

12.2.1 Team Manager Methods

This section provides reference information for each method available to team managers.

getClaimedFlowTimeCalendarDays

Syntax: Here's the method signature:

```
double getClaimedFlowTimeCalendarDays(String processId, String processVersion,
Date startCompletionTime, Date endCompletionTime, String teamDN, Map filters)
```

getClaimedFlowTimeWorkingDays

Syntax: Here is the method signature:

```
MetricsResultset getClaimedFlowTimeWorkingDays(String processId, String
processVersion, Date startCompletionTime, Date endCompletionTime, String teamDN,
Map filters)
```

getToClaimedFlowTimeCalendarDays

Syntax: Here is the method signature:

```
double getToClaimFlowTimeCalendarDays(String processId, String processVersion,
Date startCompletionTime, Date endCompletionTime, String teamDN, Map filters)
```

getToClaimedFlowTimeWorkingDays

Syntax: Here is the method signature:

```
MetricsResultset getToClaimFlowTimeWorkingDays(String processId, String
processVersion, Date startCompletionTime, Date endCompletionTime, String teamDN,
Map filters)
```


getClaimedInventory

Syntax: Here is the method signature:

```
double getClaimedInventory(String processId, String processVersion, Date startCompletionTime, Date endCompletionTime, String teamDN, Map filters)
```

getClaimedThroughputCalendarDays

Syntax: Here is the method signature:

```
double getClaimedThroughputCalendarDays(String processId, String processVersion, Date startCompletionTime, Date endCompletionTime, String teamDN, Map filters)
```

getClaimedThroughputWorkingDays

Syntax: Here is the method signature:

```
MetricsResultset getClaimedThroughputWorkingDays(String processId, String processVersion, Date startCompletionTime, Date endCompletionTime, String teamDN, Map filters)
```

getTeamLongestRunning

Syntax: Here is the method signature:

```
MetricsResultset getTeamLongestRunning(String processId, String processVersion, String teamDN, Map filters)
```

getTeamLongestClaimed

Syntax: Here is the method signature:

```
MetricsResultset getTeamLongestClaimed(String processId, String processVersion, String teamDN, Map filters)
```

getTeamFlowHistory

Syntax: Here is the method signature:

```
MetricsResultset getTeamFlowHistory(List requestIds)
```

getTeamHistoryForInitiators

Syntax: Here is the method signature:

```
MetricsResultset getTeamHistoryForInitiators(String teamDN, Map filters)
```

getTeamHistoryForRecipients

Syntax: Here is the method signature:

```
MetricsResultset getTeamHistoryForRecipients(String teamDN, Map filters)
```

getTeamRunningTime

Syntax: Here is the method signature:

```
double getTeamRunningTime(String processId, String processVersion, String teamDN,
Map filters)
```

getTeamDecisionCount

Syntax: Here is the method signature:

```
int getTeamDecisionCount(String processId, String processVersion, String teamDN,
Map filters)
```

getTeamInitiatedCount

Syntax: Here is the method signature:

```
int getTeamInitiatedCount(String processId, String processVersion, String teamDN,
Map filters)
```

getTeamRecipientCount

Syntax: Here is the method signature:

```
int getTeamRecipientCount(String processId, String processVersion, String teamDN,
Map filters)
```

12.2.2 Provisioning Application Administrator Methods

This section provides reference information for each method available to the Provisioning Application Administrator.

getActivityFlowTimeCalendarDays

Syntax: Here is the method signature:

```
double getActivityFlowTimeCalendarDays(String processId, String processVer, String
activityId, Date startTime, Date completeTime, Map filters)
```

getActivityFlowTimeWorkingDays

Syntax: Here is the method signature:

```
MetricsResultset getActivityFlowTimeWorkingDays(String processId, String
processVer, String activityId, Date startTime, Date completeTime, Map filters)
```

getActivityInventory

Syntax: Here is the method signature:

```
double getActivityInventory(String processId, String processVersion, String
activityId, Date startTime, Date completeTime, Map filters)
```

getActivityThroughputCalendarDays

Syntax: Here is the method signature:

```
double getActivityThroughputCalendarDays(String processId, String processVersion,
String activityId, Date startTime, Date completiontime, Map filters)
```

getActivityThroughputWorkingDays

Syntax: Here is the method signature:

```
MetricsResultset getActivityThroughputWorkingDays(String processId, String
processVersion, String activityId, Date startTime, Date completiontime, Map
filters)
```

getInventory

Syntax: Here is the method signature:

```
double getInventory(String processId, String processVersion, Date startTime, Date
completeTime, Map filters)
```

getLongestClaimed

Syntax: Here is the method signature:

```
MetricsResultset getLongestClaimed(String processId, String processVersion, Map
filters)
```

getLongestRunning

Syntax: Here is the method signature:

```
MetricsResultset getLongestRunning(String processId, String processVersion, Map
filters)
```

getFlowCount

Syntax: Here is the method signature:

```
int getFlowCount(String processId, String processVersion, Map filters)
```

getFlowHistory

Syntax: Here is the method signature:

```
MetricsResultset getFlowHistory(List requestIds)
```

getFlowHistoryForInitiators

Syntax: Here is the method signature:

```
MetricsResultset getFlowHistoryForInitiators(List initiators, Map filters)
```

getFlowHistoryForRecipients

Syntax: Here is the method signature:

```
MetricsResultset getFlowHistoryForRecipients(List recipients, Map filters)
```

getRunningTime

Syntax: Here is the method signature:

```
double getRunningTime(String processId, String processVersion, Map filters)
```

getThroughputCalendarDays

Syntax: Here is the method signature:

```
double getThroughputCalendarDays(String processId, String processVersion, Date  
startTime, Date completiontime, Map filters)
```

getThroughputWorkingDays

Syntax: Here is the method signature:

```
MetricsResultset getActivityThroughputWorkingDays(String processId, String  
processVersion, String activityId, Date startTime, Date completiontime, Map  
filters)
```

12.2.3 Utility Methods

This section provides reference information for each utility method. Both team managers and administrators can call these methods.

getVersion

Syntax: Here is the method signature:

```
VersionVO getVersion()
```

getAllProvisioningFlows

Syntax: Here is the method signature:

```
MetricsResultset getAllProvisioningFlows()
```

getUserActivityOnlyFlow

Syntax: Here is the method signature:

```
BasicModelVO getUserActivityOnlyFlow(String processId, String processVer)
```

getTeams

Syntax: Here is the method signature:

```
MetricsResultset getTeams()
```

getTeamMembers

Syntax: Here is the method signature:

```
MetricsResultset getTeamMembers(String teamDN)
```

12.3 Metrics Web Service Examples

This section provides examples that show how to use the Metrics Web Service to gather workflow metrics. The examples assume that you have obtained a stub, as shown in [Section 12.1.6, “Obtaining the Remote Interface,” on page 309](#), and potentially wrapped it in an object that handles the potential error conditions, as described in [Section 12.1.7, “Metrics Configuration Settings,” on page 310](#).

12.3.1 General Examples

This example uses the KEY_APPROVAL_STATUS filter to compare the decision outcomes for a provisioning request type. This could be used to generate a pie chart for example.

```
FilterConstants constants=new FilterConstants();
Map<MetricsFilter, Object> map = new HashMap<MetricsFilter, Object>();
map.put(MetricsFilter.KEY_APPROVAL_STATUS, constants.getApprovalStatusApproved());
double accepted=stubWrapper.getFlowCount(processId,processVersion,map);
map.put(MetricsFilter.KEY_APPROVAL_STATUS, constants.getApprovalStatusDenied());
double denied=stubWrapper.getFlowCount(processId,processVersion,map);
map.put(MetricsFilter.KEY_APPROVAL_STATUS, constants.getApprovalStatusError());
double error=stubWrapper.getFlowCount(processId,processVersion,map);
map.put(MetricsFilter.KEY_APPROVAL_STATUS, constants.getApprovalStatusRetract());
double retracted=stubWrapper.getFlowCount(processId,processVersion,map);
map.put(MetricsFilter.KEY_APPROVAL_STATUS,
constants.getApprovalStatusRefused());
double refused = stubWrapper.getFlowCount(processId,
processVersion, map);
```

Additional filters may be specified by adding appropriate entries to the filter map. The following examples illustrate how you might add various types of filters.

Adding a start date filter

To add a start date filter (01/01/2006 < date < 02/01/2006):

```
Calendar startDate=Calendar.getInstance();
startDate.set(2006,0,1);
Calendar endDate=Calendar.getInstance();
endDate.set(2006,1,1);
map.put(MetricsFilter.KEY_L_START_TIME,startDate);
map.put(MetricsFilter.KEY_S_START_TIME,endDate)
```

Adding a completion date filter

To add a completion date filter (02/01/2005 < date <03/01/2005)

```

Calendar startDate=Calendar.getInstance();
startDate.set(2006,0,1);
Calendar endDate=Calendar.getInstance();
endDate.set(2006,1,1);
map.put(MetricsFilter.KEY_L_COMPLETION_TIME,startDate);
map.put(MetricsFilter.KEY_S_COMPLETION_TIME,endDate)

```

Narrowing requests to a specific initiator

To narrow down counted requests to a specific initiator

```
map.put(MetricsFilter.KEY_INITIATOR, "cn=admin,ou=idmsample,o=netiq");
```

Narrowing requests to a specific recipient

To narrow down counted requests to a specific recipient

```
map.put(MetricsFilter.KEY_RECIPIENT, "cn=admin,ou=idmsample,o=netiq");
```

12.3.2 Other Examples

The following examples illustrate the use of various methods for retrieving workflow counts.

Retrieving decision counts for a team

This example describes how to retrieve the various decision outcomes of a team. The team's DN is required and can be obtained by using the `getTeams()` method:

```

FilterConstants constants=new FilterConstants();
Map<MetricsFilter, Object> map = new HashMap<MetricsFilter, Object>();
map.put(MetricsFilter.KEY_ACTIVITY_END,
constants.getActivityApproved());
double accepted = stubWrapper.getTeamDecisionCount(processId,
processVersion, teamDN, map);
map.put(MetricsFilter.KEY_ACTIVITY_END, constants.getActivityDenied());
double denied = stubWrapper.getTeamDecisionCount(processId, processVersion,
teamDN, map);
map.put(MetricsFilter.KEY_ACTIVITY_END,
constants.getActivityReassigned());
double reassigned = stubWrapper.getTeamDecisionCount(processId,
processVersion, teamDN, map);
map.put(MetricsFilter.KEY_ACTIVITY_END,
constants.getActivityRefused());
double refused = stubWrapper.getTeamDecisionCount(processId,
processVersion, teamDN, map);

```

Retrieving decision counts for requests where team members are recipients

This example describes how to retrieve the various decisions outcomes for requests for which members of the team act as recipients

```

FilterConstants constants = new FilterConstants();
Map<MetricsFilter, Object> map = new HashMap<MetricsFilter, Object>();
map.put(MetricsFilter.KEY_APPROVAL_STATUS,
constants.getActivityApproved());
double accepted = stubWrapper.getTeamRecipientCount(processId, processVersion,
teamDN, map);
map.put(MetricsFilter.KEY_APPROVAL_STATUS,
constants.getApprovalStatusDenied());
double denied = stubWrapper.getTeamRecipientCount(processId, processVersion,
teamDN, map);
map.put(MetricsFilter.KEY_APPROVAL_STATUS, constants.getApprovalStatusError());
double error = stubWrapper.getTeamRecipientCount(processId, processVersion,
teamDN, map);
map.put(MetricsFilter.KEY_APPROVAL_STATUS, constants.getApprovalStatusError());
double retracted = stubWrapper.getTeamRecipientCount(processId, processVersion,
teamDN, map);
map.put(MetricsFilter.KEY_APPROVAL_STATUS, constants.getApprovalStatusRefused());
double refused = stubWrapper.getTeamRecipientCount(processId, processVersion,
teamDN, map);

```

Retrieving requests that have been claimed but not acted on

This example describes how to retrieve the requests started after 03/01/2006 that have been claimed but not acted upon.

```

Map<MetricsFilter, Object> map = new HashMap<MetricsFilter, Object>();
Calendar startDate=Calendar.getInstance();
startDate.set(2006,2,1);
map.put(MetricsFilter.KEY_L_START_TIME,startDate);
MetricsResultset rset = stubWrapper.getLongestClaimed(processId, processVersion,
map);

```

Retrieving the longest running requests started by a particular user

This example describes how to retrieve the longest running requests that have been started by initiator "cn=admin,ou=idmsample,o=netiq";

```

Map<MetricsFilter, Object> map = new HashMap<MetricsFilter, Object>();
map.put(MetricsFilter.KEY_INITIATOR, "cn=admin,ou=idmsample,o=netiq");
MetricsResultset rset = stubWrapper.getLongestRunning(processId, processVersion,
map);

```

Retrieving activity inventory

This example describes the average inventory for users handling decision with activity id "managerApproval" between 01/01/2006 and 02/01/2006

```

Map<MetricsFilter, Object> map = new HashMap<MetricsFilter, Object>();
Calendar startDate=Calendar.getInstance();
startDate.set(2006,0,1);
Calendar endDate=Calendar.getInstance();
endDate.set(2006,1,1);
MetricsResultset rset = stubWrapper.getActivityInventory(processId,
processVersion,"managerApproval", startDate, endDate, map );

```

Retrieving the Claimed Throughput and Inventory for a Team

This example describes the team's throughput and inventory over the time interval between 01/01/2006 and 02/01/2006

```
Map<MetricsFilter, Object> map = new HashMap<MetricsFilter, Object>();
Calendar startDate=Calendar.getInstance();
startDate.set(2006,0,1);
Calendar endDate=Calendar.getInstance();
endDate.set(2006,1,1);
double throughput = stubWrapper.getClaimedThroughputCalendarDays(processId,
processVersion, startDate, endDate,teamDN, map);
double inventory = stubWrapper.getClaimedInventory(processId, processVersion,
startDate, endDate, teamDN, map)
```

13 Notification Web Service

This section describes the Notification Web Service, which allows SOAP clients to use the email notification facility.

13.1 About the Notification Web Service

The Identity Manager User Application includes an email notification facility that lets you send email messages to notify users of changes in the state of the provisioning system, as well as tasks that they need to perform. To support access by third-party software applications, the notification facility includes a Web service endpoint. The endpoint lets you send an email message to one or more users. When you send an email, you include parameters that specify the target email address, the email template to use, and the replacement values for tokens in the email template.

This Appendix describes the programming interface for the Notification Web Service.

13.1.1 Accessing the Test Page

You can access the Notification Web Service endpoint using a URL similar to the following:

```
http://server:port/warcontext/notification/service?test
```

For example, if your server is named “myserver”, your User Application is listening on port 8080, and your User Application war file is named “IDMPROV”, the URL would be:

```
http://myserver:8080/IDMPROV/notification/service?test
```

You can also access the SOAP endpoint by going to the **Administration** within the User Application. To do this, you need to select the **Application Configuration** tab, then select **Web Services** from the left-navigation menu. After selecting **Web Services**, pick the Web Service endpoint you want from the list.

WARNING: The test page is disabled by default. Since some of the methods allow data to be updated, the test page presents a potential security vulnerability and should not be allowed in a production environment. For details on enabling the test page, see the instructions provided for the Role Service in [“Enabling the Test Page” on page 359](#).

13.1.2 Accessing the WSDL

You can access the WSDL for the Notification Web Service using a URL similar to the following:

```
http://server:port/warcontext/notification/service?wsdl
```

For example, if your server is named “myserver”, your User Application is listening on port 8080, and your User Application war file is named “IDMPROV”, the URL would be:

```
http://myserver:8080/IDMPROV/notification/service?wsdl
```

13.1.3 Generating the Stub Classes

Before using the Web Service, you need to use the WSSDK tool or another SOAP tool kit to generate the stub classes. To allow your code to find the stub classes, you also need to add the JAR that contains the stub classes to your classpath.

If you want to use the NetIQ WSSDK tool, you can generate the client stubs by extracting the WSDL and running the `wsdl2java` utility. For example, you could run this command to generate the stubs in a package called `com.novell.ws.client.notification`:

```
"C:\Program Files\Java\jdk1.6.0_31\bin\java" -cp "../lib/wssdk.jar;../lib/jaxrpc-api.jar;../lib/mail.jar;../lib/activation.jar";c:\Program Files\Java\jdk1.6.0_31\lib\tools.jar"; com.novell.soa.ws.impl.tools.wsdl2java.Main -verbose -ds gensrc -d C:\ -noskel -notie -genclient -keep -package com.novell.ws.client.notification -javadoc notification.wsdl
```

You can change the `wsdl2java` parameters to suit your requirements.

13.2 Notification Web Service API

This section provides details about the methods available with the Notification Web service. This API presumes you're using Java code generated by the WSSDK toolkit. The API will be different if you're using another Web Service toolkit.

All of the methods throw `RemoteException`. To improve readability, the `throws` clause has been omitted from the method signatures.

13.2.1 iRemoteNotification

This section provides reference information for each method associated with the `iRemoteNotification` interface.

getVersion

Returns the version number of the notification facility you're running.

Syntax: Here is the method signature:

```
VersionVO getVersion()
```

sendNotification

Sends an email notification.

Syntax: Here is the method signature:

```
void sendNotification(NotificationMap arg0)
```

13.2.2 BuiltInTokens

This section provides reference information for each method associated with the `BuiltInTokens` class.

BuiltInTokens constructor

The BuiltInTokens class has a single constructor.

Syntax: Here is the constructor for the BuiltInTokens class:

```
BuiltInTokens()
```

getTO

Returns the fixed string TO, which can be used as a key to identify the value for the TO system token.

Syntax: Here is the method signature:

```
public java.lang.String getTO()
```

getCC

Returns the fixed string CC, which can be used as a key to identify the value for the CC system token.

Syntax: Here is the method signature:

```
public java.lang.String getCC()
```

getBCC

Returns the fixed string BCC, which can be used as a key to identify the value for the BCC system token.

Syntax: Here is the method signature:

```
public java.lang.String getBCC()
```

getTO_DN

Returns the fixed string TO_DN, which can be used as a key to identify the value for the TO_DN system token.

Syntax: Here is the method signature:

```
public java.lang.String getTO_DN()
```

getCC_DN

Returns the fixed string CC_DN, which can be used as a key to identify the value for the CC_DN system token.

Syntax: Here is the method signature:

```
public java.lang.String getCC_DN()
```

getBCC_DN

Returns the fixed string BCC_DN, which can be used as a key to identify the value for the BCC_DN system token.

Syntax: Here is the method signature:

```
public java.lang.String getBCC_DN()
```

getREPLYTO

Returns the fixed string REPLYTO, which can be used as a key to identify the value for the REPLYTO system token.

Syntax: Here is the method signature:

```
public java.lang.String getREPLYTO()
```

getREPLYTO_DN

Returns the fixed string REPLYTO_DN, which can be used as a key to identify the value for the REPLYTO_DN system token.

Syntax: Here is the method signature:

```
public java.lang.String getREPLYTO_DN()
```

getLOCALE

Returns the fixed string LOCALE, which can be used as a key to identify the value for the LOCALE system token.

Syntax: Here is the method signature:

```
public java.lang.String getLOCALE()
```

getNOTIFICATION_TEMPLATE_DN

Returns the fixed string NOTIFICATION_TEMPLATE, which can be used as a key to identify the value for the NOTIFICATION_TEMPLATE system token.

Syntax: Here is the method signature:

```
public java.lang.String getNOTIFICATION_TEMPLATE_DN()
```

13.2.3 Entry

The Entry class represents an entry in an EntryArray object. It is used to specify a token in an email template.

This section provides reference information for each method associated with the Entry class.

Entry constructors

The Entry class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
Entry()
```

Syntax 2: Here is the syntax for a constructor that takes two parameters, the key value and an array of values:

```
Entry(java.lang.String KeyVal, StringArray ValuesVal)
```

getKey

Returns the key defined for the Entry object. The key identifies the token.

Syntax: Here is the method signature:

```
java.lang.String getKey()
```

setKey

Sets the key for the Entry object. The key identifies the token. If the object represents a built-in token, you can use the BuiltInTokens class to set the key. Otherwise, you can pass a string to the setKey method that specifies the key.

Syntax: Here is the method signature:

```
void setKey(java.lang.String KeyVal)
```

getValues

Returns a StringArray object representing the values for the Entry object.

Syntax: Here is the method signature:

```
StringArray getValues()
```

setValues

Sets the values for the Entry object.

Syntax: Here is the method signature:

```
void setValues(StringArray ValuesVal)
```

13.2.4 EntryArray

The EntryArray class is a container for an array of Entry objects. It is contained by the NotificationMap object.

This section provides reference information on the methods associated with the EntryArray class.

EntryArray constructors

The EntryArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
EntryArray()
```

Syntax 2: Here is the syntax for a constructor that takes an array of Entry objects as a parameter:

```
EntryArray(Entry[] EntryVal)
```

getEntry

Returns the Entry object contained within this EntryArray object.

Syntax: Here is the method signature:

```
Entry[] getEntry()
```

setEntry

Sets the Entry object for this EntryArray object.

Syntax: Here is the method signature:

```
void setEntry(Entry[] EntryVal)
```

13.2.5 NotificationMap

The NotificationMap object is a map that contains an EntryArray object. It is passed to the sendNotification method on the stub.

This section provides reference information for the methods associated with the NotificationMap class.

NotificationMap constructors

The NotificationMap class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
NotificationMap()
```

Syntax 2: Here is the syntax for a constructor that takes an EntryArray object as a parameter:

```
NotificationMap(EntryArray EntriesVal)
```

getEntries

Returns the EntryArray object contained by this NotificationMap object.

Syntax: Here is the method signature:

```
EntryArray getEntries()
```

setEntries

Sets the EntryArray object for this NotificationMap object.

Syntax: Here is the method signature:

```
void setEntries(EntryArray EntriesVal)
```

13.2.6 NotificationService

This section provides reference information for the NotificationService interface.

getIRemoteNotificationPort

Gets the stub for the remote service. The stub is a port of type IRemoteNotification.

Syntax: Here is the method signature:

```
IRemoteNotification getIRemoteNotificationPort() throws  
javax.xml.rpc.ServiceException
```

13.2.7 **StringArray**

This section provides reference information for the `StringArray` class.

StringArray constructors

The `StringArray` class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
StringArray()
```

Syntax 2: Here is the syntax for a constructor that takes a `String` array as a parameter:

```
StringArray(java.lang.String[] StringVal)
```

getString

Returns the array of strings defined for this `StringArray` object.

Syntax: Here is the method signature:

```
java.lang.String[] getString()
```

setString

Sets the array of strings for this `StringArray` object. This method is called by the second constructor, which takes a `String` array as a parameter.

Syntax: Here is the method signature:

```
void setString(java.lang.String[] StringVal)
```

13.2.8 **VersionVO**

This section provides reference information on the `VersionVO` class.

getValue

Returns the version number of the service.

Syntax: Here is the method signature:

```
java.lang.String getValue()
```

13.3 Notification Example

The following code example shows how one might use the Notification service to send an email message using a pre-defined system template. To get a reference to the SOAP endpoint for the Notification service, a call is made to the `getNotificationStub()` method. After acquiring the stub interface, the code sets the email notification template as well as values for the built-in tokens in the template. In addition, the code specifies values for the `requestTitle` and `initiatorFullName` tokens. For each token, the code creates an `Entry` object. Once all of the entries have been created, it packages the entry array into a map of type `NotificationMap`, which is then passed to the `sendNotification` method on the stub.

```
import java.util.Properties;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.xml.rpc.Stub;
import java.rmi.RemoteException;
//
// Notification imports
import com.novell.ws.client.notification.IRemoteNotification;
import com.novell.ws.client.notification.BuiltInTokens;
import com.novell.ws.client.notification.Entry;
import com.novell.ws.client.notification.EntryArray;
import com.novell.ws.client.notification.StringArray;
import com.novell.ws.client.notification.NotificationMap;
import com.novell.ws.client.notification.IRemoteNotification;
import com.novell.ws.client.notification.NotificationService;

public class NotificationTest
{
    private static final int LOCALHOST    = 0; // localhost
    private static final int TESTSERVER   = 1; // testserver
    private static final int SELECTED_URL = TESTSERVER;

    private String [] SERVER_URLS = {
        "http://localhost:8080/IDMProv/notification/service",
        "http://testserver:8080/IDMProv/notification/service"
    };
    private String url = SERVER_URLS[SELECTED_URL];
    private String username = "cn=admin,ou=idmsample,o=netiq";
    private String password = "test";

    public void emailNotificationTestCase()
    throws Exception
    {
        System.out.println("\nCalling emailNotificationTestCase() test
case");

        try
        {
            String targetEmailAddress = "jsmith@somewhere.com";
            //
            // Get the notification stub
            IRemoteNotification notificationStub =
getNotificationStub(url, username, password);

            BuiltInTokens builtInTokens = new BuiltInTokens();
            //
```



```

        // Set the To: entry
        Entry to = new Entry();
        to.setKey(builtInTokens.getTO());
        StringArray arr = new StringArray(new
String[] {targetEmailAddress} );
        to.setValues(arr);
        //
        // Set which email template to use : list in iManager
        (Workflow Admin->Email Templates)
        Entry notificationTemplate = new Entry();

notificationTemplate.setKey(builtInTokens.getNOTIFICATION_TEMPLATE_DN());
        //
        // Use one of the email templates specifying DN
        String EMAIL_TEMPLATE_NAME = "Provisioning Notification";
        String templatedDN = "cn=" + EMAIL_TEMPLATE_NAME +
",cn=Default Notification Collection,cn=Security";
        arr = new StringArray(new String[] {templatedDN} );
        notificationTemplate.setValues(arr);
        //
        // Substitute key values defined in email templates
        Entry token1 = new Entry();
        token1.setKey("requestTitle"); // key is %requestTitle%
        arr = new StringArray(new String[] {"Sample Email using
Notification Web Service" } );
        token1.setValues(arr);
        Entry token2 = new Entry();
        token2.setKey("initiatorFullName");
        arr = new StringArray(new String[] {username} );
        token2.setValues(arr);
        //
        // Setup the notification map
        NotificationMap map = new NotificationMap();
        Entry[] entries = new
Entry[] {to,notificationTemplate,token1,token2};
        EntryArray entryArray = new EntryArray();
        entryArray.setEntry(entries);
        map.setEntries(entryArray);
        //
        // Make the notification endpoint call
        notificationStub.sendNotification(map);
    }
    catch(RemoteException error)
    {
        System.out.println(error.getMessage() );
        throw new Exception(error.getMessage() );
    }
}

/**
 * Method to obtain the remote interface to the Notification
endpoint
 * @param _url
 * @param _username
 * @param _password
 * @return IRemoteNotification interface
 * @throws Exception
 */
private IRemoteNotification getNotificationStub(String _url,

```

```

String _username, String _password)
    throws Exception
    {
        Properties properties = new Properties();
        properties.put(Context.INITIAL_CONTEXT_FACTORY,
"org.jnp.interfaces.NamingContextFactory");

        String lookup =
"xmlrpc:soap:com.novell.ws.client.notification.NotificationService";

        InitialContext ctx = new InitialContext();
        NotificationService svc = (NotificationService)
ctx.lookup(lookup);

        Stub stub = (Stub)svc.getIRemoteNotificationPort();

        stub._setProperty(Stub.USERNAME_PROPERTY, _username);
        stub._setProperty(Stub.PASSWORD_PROPERTY, _password);
        stub._setProperty(Stub.SESSION_MAINTAIN_PROPERTY,
Boolean.TRUE);
        stub._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY, _url);

        return (IRemoteNotification) stub;
    }
}

```

14 Directory Abstraction Layer (VDX) Web Service

This section describes the VDX Web Service, which allows SOAP clients to access the directory abstraction layer.

14.1 About the Directory Abstraction Layer (VDX) Web Service

The directory abstraction layer provides a logical view of the Identity Vault data. To support access by third-party software applications, the directory abstraction layer includes a Web service endpoint called the VDX Web Service. This endpoint lets you access the attributes associated with entities defined in the directory abstraction layer. It also lets you perform ad hoc searches for entities and execute predefined searches called global queries. You can think of global queries as stored procedures for LDAP.

This Appendix describes the programming interface for the VDX Web Service.

14.1.1 Accessing the Test Page

You can access the VDX Web Service endpoint using a URL similar to the following:

```
http://server:port/warcontext/vdx/service?test
```

For example, if your server is named “myserver”, your User Application is listening on port 8080, and your User Application war file is named “IDMPROV”, the URL would be:

```
http://myserver:8080/IDMPROV/vdx/service?test
```

You can also access the SOAP endpoint by going to the **Administration** within the User Application. To do this, you need to select the **Application Configuration** tab, then select **Web Services** from the left-navigation menu. After selecting **Web Services**, pick the Web Service endpoint you want from the list.

WARNING: The test page is disabled by default. Since some of the methods allow data to be updated, the test page presents a potential security vulnerability and should not be allowed in a production environment. For details on enabling the test page, see the instructions provided for the Role Service in [“Enabling the Test Page” on page 359](#).

14.1.2 Accessing the WSDL

You can access the WSDL for the VDX Web Service using a URL similar to the following:

```
http://server:port/warcontext/vdx/service?wsdl
```

For example, if your server is named “myserver”, your User Application is listening on port 8080, and your User Application war file is named “IDMPROV”, the URL would be:

http://myserver:8080/IDMPROV/vdx/service?wsdl

14.1.3 Generating the Stub Classes

Before using the Web Service, you need to use the WSSDK tool or another SOAP tool kit to generate the stub classes. To allow your code to find the stub classes, you also need to add the JAR that contains the stub classes to your classpath.

If you want to use the NetIQ WSSDK tool, you can generate the client stubs by extracting the WSDL and running the `wsdl2java` utility. For example, you could run this command to generate the stubs in a package called `com.novell.ws.client.vdx`:

```
"C:\Program Files\Java\jdk1.6.0_31\bin\java" -cp "../lib/wssdk.jar;../lib/jaxrpc-api.jar;../lib/mail.jar;../lib/activation.jar";"c:\Program Files\Java\jdk1.6.0_31\lib\tools.jar"; com.novell.soa.ws.impl.tools.wsdl2java.Main -verbose -ds gensrc -d C:\ -noskel -notie -genclient -keep -package com.novell.ws.client.vdx -javadoc vdx.wsdl
```

You can change the `wsdl2java` parameters to suit your requirements.

14.1.4 Removing Administrator Credential Restrictions

The VDX Web Service supports two levels of security, one that restricts access to Provisioning Administrators and another that restricts access to the authenticated user. The default setting restricts access to all operations to the Provisioning Administrator.

You can modify the settings for security configuration in the `ism-configuration.properties` file, located by default in the `/netiq/idm/apps/tomcat/conf` directory. . Each property can be set to true or false. A value of true locks down the operation, whereas a value of false opens up the operation. After making your changes, you need to import the file back into the WAR file and redeploy.

You can open up the VDX Web Service to authenticated users by setting the `VirtualDataService/soap` property to false. To open up a particular operation to authenticated users, you need to set the property for that operation (`VirtualDataService/soap/operation`) to false as well. If you set all of the properties to false, you can open up all operations to authenticated users. The *operation* names are the same as the names of the methods supported by the service.

Example To ensure that the security configuration opens up all operations within the VDX Web Service, the `ism-configuration.properties` file must have the following setting:

```
VirtualDataService/soap = false
```

To restrict `globalQuery`, change the `VirtualDataService/soap/globalQuery` to true in the `ism-configuration.properties` file.

Even though the service does not require the Administrator credentials since you set the `VirtualDataService/soap` property to false, the `globalQuery` operation will still require the Administrator credentials since you set a property for the operation to true.

14.2 VDX Web Service API

This section provides details about the methods available with the VDX Web service. This API presumes you're using Java code generated by the WSSDK toolkit. The API will be different if you're using another Web Service toolkit.

All of the methods throw `VdxServiceException`. To improve readability, the throws clause has been omitted from the method signatures.

14.2.1 IRemoteVdx

This section provides reference information for each method associated with the `IRemoteVdx` interface.

getVersion

Returns the version number of the VDX service you're running.

Syntax: Here is the method signature:

```
VersionVO getVersion() throws java.rmi.RemoteException;
```

globalQuery

Allows you to execute predefined searches called global queries. Global queries are saved searches for LDAP. They provide some of the capabilities of stored procedures.

To define a global query, you need to use the directory abstraction layer editor. For details, see the chapter on the directory abstraction layer editor in the *Identity Manager User Application: Design Guide*.

Syntax: Here is the method signature:

```
java.lang.String[] globalQuery(java.lang.String queryDN, StringMap  
queryParameterValues) throws VdxServiceException, java.rmi.RemoteException;
```

query

Allows you to perform ad hoc queries by specifying an entity, a set of attributes, and a query expression that filters the data returned.

Syntax: Here is the method signature:

```
EntityAttributeMap query(java.lang.String entityDefinition, java.lang.String[]  
attributeKeys, java.lang.String queryFilter) throws VdxServiceException,  
java.rmi.RemoteException;
```

Query Grammar

The `queryFilter` parameter of the `query()` method lets you pass in search criteria expressions that filter the data returned. This section describes the grammar for these expressions.

Query syntax 1: The simplest form of a query is the following:

RelationalExpression1

Query syntax 2: A query can also combine relational expressions with a logical operator:

RelationalExpression1 logicalOperator RelationalExpression2

Query syntax 3: Alternatively, a query can use parentheses to set off the expressions:

(RelationalExpression1) logicalOperator (RelationalExpression2)

Query syntax 4: A query can also use parentheses to set off sub queries:

*RelationalExpression1 logicalOperator (RelationalExpression2 logicalOperator1
RelationalExpression3)*

Relational expressions must be separated by a logical operator which must remain the same. In other words, the following query is valid:

expression1 AND expression2 AND expression3

However, this query is not valid:

expression1 AND expression2 OR expression3

You can use parentheses to create a condition group, as in the following example:

expression1 AND (expression2 OR expression3)

Grammar for Relational Expressions

Relational expression syntax: A relational expression must conform to this syntax:

attribute relationalOperator value

Grammar for Operators and Values

Relational operators: The relational operator must be one of the following:

*> , < , >= , <= , = , != , !< , !> , !<= , !>= , STARTWITH, !STARTWITH, IN , !IN ,
PRESENT, !PRESENT*

Logical operators: The logical operator must be one of the following:

AND, OR

Value: The value side of an expression must be one of the following:

'foo', "foo", 1-9, true, false

The PRESENT and !PRESENT relational operators require no value.

getAttribute

Returns a single Attribute object that can be used to retrieve and examine data for an attribute in the directory abstraction layer.

Syntax: Here is the method signature:

```
Attribute getAttribute(java.lang.String objectDN, java.lang.String  
entityDefinition, java.lang.String attributeKey) throws VdxServiceException,  
java.rmi.RemoteException;
```

getAttributes

Returns an array of Attribute objects that can be used to retrieve and examine data in the directory abstraction layer.

Syntax: Here is the method signature:

```
Attribute[] getAttributes(java.lang.String objectDN, java.lang.String
entityDefinition, java.lang.String[] attributeKeys) throws VdxServiceException,
java.rmi.RemoteException;
```

14.2.2 Attribute

The Attribute class represents an attribute in the directory abstraction layer.

This section provides reference information for the Attribute class.

Attribute constructors

The Attribute class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no arguments:

```
Attribute()
```

Syntax 2: Here is the syntax for a constructor that takes arrays of all the supported data types as arguments:

```
Attribute(ByteArrayArray BinariesVal, BooleanArray BooleansVal, DateArray
DatesVal, IntegerArray IntegersVal, StringArray StringsVal, AttributeType TypeVal)
```

getBinaries

Returns the ByteArrayArray object for the attribute.

Syntax: Here is the method signature:

```
ByteArrayArray getBinaries()
```

setBinaries

Sets the ByteArrayArray object for the attribute.

Syntax: Here is the method signature:

```
void setBinaries(ByteArrayArray BinariesVal)
```

getBooleans

Returns the BooleanArray object for the attribute.

Syntax: Here is the method signature:

```
BooleanArray getBooleans()
```

setBooleans

Sets the BooleanArray object for the attribute.

Syntax: Here is the method signature:

```
void setBooleans(BooleanArray BooleansVal)
```

getDates

Returns the DateArray object for the attribute.

Syntax: Here is the method signature:

```
DateArray getDates()
```

setDates

Sets the DateArray object for the attribute.

Syntax: Here is the method signature:

```
void setDates(DateArray DatesVal)
```

getIntegers

Returns the IntegerArray object for the attribute.

Syntax: Here is the method signature:

```
IntegerArray getIntegers()
```

setIntegers

Sets the IntegerArray object for the attribute.

Syntax: Here is the method signature:

```
void setIntegers(IntegerArray IntegersVal)
```

getStrings

Returns the StringArray object for the attribute.

Syntax: Here is the method signature:

```
StringArray getStrings()
```

setStrings

Set the StringArray object for the attribute.

Syntax: Here is the method signature:

```
void setStrings(StringArray StringsVal)
```


getType

Returns the AttributeType object for the attribute.

Syntax: Here is the method signature:

```
AttributeType getType()
```

setType

Sets the AttributeType object for the attribute.

Syntax: Here is the method signature:

```
void setType(AttributeType TypeVal)
```

14.2.3 AttributeArray

This section provides reference information on the AttributeArray class.

AttributeArray constructors

The AttributeArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
AttributeArray()
```

Syntax 2: Here is the syntax for a constructor that takes an array of Attribute objects as a parameter:

```
AttributeArray(Attribute[] AttributeVal)
```

getAttribute

Returns an array of Attribute objects.

Syntax: Here is the method signature:

```
Attribute[] getAttribute()
```

setAttribute

Sets the array of Attribute objects associated with the AttributeArray class.

Syntax: Here is the method signature:

```
void setAttribute(Attribute[] AttributeVal)
```

14.2.4 AttributeType

This section provides reference information on the AttributeType class.

AttributeType constructors

The AttributeType class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
protected AttributeType(java.lang.String value)
```

getValue

Returns a String that indicates the attribute type.

Syntax: Here is the method signature:

```
java.lang.String getValue()
```

14.2.5 BooleanArray

This section provides reference information for the BooleanArray class.

BooleanArray constructors

The BooleanArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
BooleanArray()
```

Syntax 2: Here is the syntax for a constructor that takes a boolean value as a parameter:

```
BooleanArray(boolean[] BooleanVal)
```

getBoolean

Returns an array of boolean values for an attribute.

Syntax: Here is the method signature:

```
boolean[] getBoolean()
```

setBoolean

Sets an array of boolean values for an attribute.

Syntax: Here is the method signature:

```
void setBoolean(boolean[] BooleanVal)
```

14.2.6 ByteArrayArray

This section provides reference information on the ByteArrayArray class.

ByteArrayArray constructors

The ByteArrayArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
ByteArrayArray()
```

Syntax 2: Here is the syntax for a constructor that takes a Base 64 binary value as a parameter:

```
ByteArrayArray(byte[][] Base64BinaryVal)
```

getBase64Binary

Returns a two-dimensional array of bytes for an attribute.

Syntax: Here is the method signature:

```
byte[][] getBase64Binary()
```

setBase64Binary

Sets a two-dimensional array of bytes for an attribute.

Syntax: Here is the method signature:

```
void setBase64Binary(byte[][] Base64BinaryVal)
```

14.2.7 DateArray

This section provides reference information for the DateArray class.

DateArray constructors

The DateArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
DateArray()
```

Syntax 2: Here is the syntax for a constructor that takes a Calendar array as a parameter:

```
DateArray(java.util.Calendar[] DatetimeVal)
```

getDatetime

Returns an array of Calendar objects for an attribute.

Syntax: Here is the method signature:

```
java.util.Calendar[] getDatetime()
```

setDatetime

Sets an array of Calendar objects for an attribute.

Syntax: Here is the method signature:

```
void setDatetime(java.util.Calendar[] DatetimeVal)
```

14.2.8 EntryAttributeMap

The EntryAttributeMap class is a container for an EntryArray object. It is returned by the query method on the stub.

This section provides reference information on the methods associated with the EntryAttributeMap class.

EntryAttributeMap constructors

The EntryAttributeMap class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
EntryAttributeMap()
```

Syntax 2: Here is the syntax for a constructor that takes an EntryArray object as a parameter:

```
EntityAttributeMap(EntryArray EntriesVal)
```

getEntries

Returns the EntryArray object contained within this EntryAttributeMap object.

Syntax: Here is the method signature:

```
EntryArray getEntries()
```

setEntries

Sets the EntryArray object for this EntryAttributeMap object.

Syntax: Here is the method signature:

```
void setEntry(EntryArray EntriesVal)
```

14.2.9 Entry

The Entry class represents an entry in an EntryArray object.

This section provides reference information for each method associated with the Entry class.

Entry constructors

The Entry class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
Entry()
```

Syntax 2: Here is the syntax for a constructor that takes two parameters, the key value and an array of attribute values:

```
Entry(java.lang.String KeyVal, AttributeArray ValuesVal)
```

getKey

Returns the key defined for the Entry object. The key identifies the attribute.

Syntax: Here is the method signature:

```
java.lang.String getKey()
```

setKey

Sets the key for the Entry object. The key identifies the attribute.

Syntax: Here is the method signature:

```
void setKey(java.lang.String KeyVal)
```

getValues

Returns a AttributeArray object representing the values for the Entry object.

Syntax: Here is the method signature:

```
AttributeArray getValues()
```

setValues

Sets the values for the Entry object.

Syntax: Here is the method signature:

```
void setValues(AttributeArray ValuesVal)
```

14.2.10 EntryArray

The EntryArray class is a container for an array of Entry objects. It is contained by the EntryAttributeMap object.

This section provides reference information on the methods associated with the EntryArray class.

EntryArray constructors

The EntryArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
EntryArray()
```

Syntax 2: Here is the syntax for a constructor that takes an array of Entry objects as a parameter:

```
EntryArray(Entry[] EntryVal)
```

getEntry

Returns the Entry object contained within this EntryArray object.

Syntax: Here is the method signature:

```
Entry[] getEntry()
```

setEntry

Sets the Entry object for this EntryArray object.

Syntax: Here is the method signature:

```
void setEntry(Entry[] EntryVal)
```

14.2.11 IntegerArray

This section provides reference information for the IntegerArray class.

IntegerArray constructors

The IntegerArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
IntegerArray()
```

Syntax 2: Here is the syntax for a constructor that takes an int array as a parameter:

```
IntegerArray(int[] IntVal)
```

getInt

Returns an array of integers for an attribute.

Syntax: Here is the method signature:

```
int[] getInt()
```

setInt

Sets an array of integers for an attribute.

Syntax: Here is the method signature:

```
void setInt(int[] IntVal)
```

14.2.12 StringArray

The StringArray class is a container for an array of String objects. When you call the query() and getAttributes() methods, you pass in a StringArray object to specify which attributes you want to retrieve values for.

This section provides reference information for the StringArray class.

StringArray constructors

The StringArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
StringArray()
```

Syntax 2: Here is the syntax for a constructor that takes an String array as a parameter:

```
StringArray(java.lang.String[] StringVal)
```

getString

Returns the array of String objects associated with the StringArray object.

Syntax: Here is the method signature:

```
java.lang.String[] getString()
```

setString

Sets the array of String objects associated with the StringArray object.

Syntax: Here is the method signature:

```
void setString(java.lang.String[] StringVal)
```

14.2.13 StringEntry

The StringEntry class is contained by the the StringEntryArray class.

This section provides reference information for the StringEntry class.

StringEntry constructors

The StringEntry class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
StringEntry()
```

Syntax 2: Here is the syntax for a constructor that takes a key and a String value as parameters:

```
StringEntry(java.lang.String KeyVal, java.lang.String ValuesVal)
```

getKey

Returns the key defined for the StringEntry object.

Syntax: Here is the method signature:

```
java.lang.String getKey()
```

setKey

Sets the key for the StringEntry object.

Syntax: Here is the method signature:

```
void setKey(java.lang.String KeyVal)
```

14.2.14 StringEntryArray

The `StringEntryArray` class is a container for an array of `StringEntry` objects. It is contained by the `StringMap` object.

This section provides reference information for the `StringEntryArray` class.

StringEntryArray constructors

The `StringEntryArray` class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
StringEntryArray()
```

Syntax 2: Here is the syntax for a constructor that takes a `StringEntry` array as a parameter:

```
StringEntryArray(StringEntry[] StringentryVal)
```

getStringentry

Returns the key for the `StringEntryArray` object.

Syntax: Here is the method signature:

```
StringEntry[] getStringentry()
```

setStringentry

Sets the key for the `StringEntryArray` object.

Syntax: Here is the method signature:

```
void setStringentry(StringEntry[] StringentryVal)
```

14.2.15 StringMap

The `StringMap` is a container for a `StringEntryArray` object.

This section provides reference information on the `StringMap` class.

StringMap constructors

The `StringMap` class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
StringMap()
```

Syntax 2: Here is the syntax for a constructor that takes a `StringEntryArray` as a parameter:

```
StringMap(StringEntryArray EntriesVal)
```

getEntries

Returns the `StringEntryArray` object contained by this `StringMap` object.

Syntax: Here is the method signature:

```
StringEntryArray getEntries()
```

setEntries

Sets the `StringEntryArray` object for this `StringMap` object.

Syntax: Here is the method signature:

```
void setEntries(StringEntryArray EntriesVal)
```

14.2.16 VdxService

This section provides reference information for the `VdxService` interface.

getIRemoteVdxPort

Gets the stub for the remote service. The stub is a port of type `IRemoteVdx`.

Syntax: Here is the method signature:

```
IRemoteVdx getIRemoteVdxPort() throws javax.xml.rpc.ServiceException;
```

14.2.17 VersionVO

This section provides reference information on the `VersionVO` class.

getValue

Returns the version number of the service.

Syntax: Here is the method signature:

```
java.lang.String getValue()
```

14.3 VDX Example

The following code example shows how one might use the VDX service to access the attributes associated with entities defined in the directory abstraction layer. It demonstrates the use of ad hoc searches, as well as predefined searches called global queries. This code listing includes examples that use the `getAttribute()`, `getAttributes()`, `query()`, and `globalQuery()` methods on the service.

To get a reference to the SOAP endpoint for the VDX service, it calls a method called `getVdxStub()`. The implementation for this method is shown at the end of the listing:

NOTE: This example presumes that you have generated client stubs from the WSDL file `IRemoteVdx.wsdl`. Use the SOAP stack provider of your choice (such as AXIS or CFX) to generate client stubs.

With Apache CXF, for example, you should be able to generate the stubs to match the package names in the `import` statement by using the following command:

```
wsdl2java -p com.netiq.ws.client.vdx IRemoteVdx.wsdl
```

```

import java.util.Properties;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.xml.rpc.Stub;
import java.rmi.RemoteException;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.rmi.RemoteException;
import java.util.Calendar;
import java.util.Date;
import java.util.Hashtable;
import java.util.Map;
//
// Vdx imports
import com.netiq.ws.client.vdx.IRemoteVdx;
import com.netiq.ws.client.vdx.VdxService;
import com.netiq.ws.client.vdx.VdxServiceException;
import com.netiq.ws.client.vdx.VersionVO;
import com.netiq.ws.client.vdx.Attribute;
import com.netiq.ws.client.vdx.AttributeArray;
import com.netiq.ws.client.vdx.AttributeType;
import com.netiq.ws.client.vdx.ByteArrayArray;
import com.netiq.ws.client.vdx.BooleanArray;
import com.netiq.ws.client.vdx.DateArray;
import com.netiq.ws.client.vdx.StringArray;
import com.netiq.ws.client.vdx.IntegerArray;
import com.netiq.ws.client.vdx.EntryArray;
import com.netiq.ws.client.vdx.Entry;
import com.netiq.ws.client.vdx.EntityAttributeMap;

public class ServiceTest
{
    public static final int VDX                = 0;
    public static final int NOTIFICATION      = 1;
    public static final int RESOURCE          = 2;
    public static final int ENDPOINT_SERVICE  = VDX;

    private static final int LOCALHOST       = 0; // localhost
    private static final int TESTSERVER      = 1; // testserver
    private static final int SELECTED_URL    = TESTSERVER;

    private String [] SERVER_URLS = {
        "http://localhost:8080/IDMProv/vdx/service",
        "http://testserver:8080/IDMProv/vdx/service"
    };

    private String url = SERVER_URLS[SELECTED_URL];
    private String username = "cn=admin,ou=idmsample,o=netiq";
    private String password = "test";

    private String [] userAttributes = {
        // "passwordAllowChange", // boolean
        "UserPhoto",           // binary
        // "loginTime",           // time
        "Department",         // string
        "Title",
    };

```

```

        "Email",
        "manager", // dn = string
        "TelephoneNumber",
        "directReports",
        "FirstName",
        //"surname",
        "group",
        "srvprvHideAttributes",
        "NotificationPrefs",
        "srvprvQueryList",
        "Location",
    };

public ServiceTest() { };

public static void main(String [] args)
{
    ServiceTest serviceTest = new ServiceTest();
    //
    // Set default if no params are given
    int wService = ENDPOINT_SERVICE;
    if(args.length == 1)
        wService = Integer.parseInt(args[0]);

    try
    {
        serviceTest.run(wService);
    }
    catch(Exception e)
    {
        System.exit(-1);
    }
}

private void waitHere(long _time) { try { Thread.sleep(_time *
1000); } catch(InterruptedException ie) {} }

public void run(int _service)
throws Exception
{
    if(_service == VDX)
    {
        System.out.println("Calling VDX endpoint");
        //
        // Get the version number
        getVersionTestCase();
        waitHere(2);
        //
        // Get attribute data for entity user
        getAttributeTestCase();
        waitHere(2);
        //
        // Get attributes
        getAttributesTestCase();
        waitHere(2);
        //
        // Query attributes
        queryAttributesTestCase();
    }
}

```

```

        waitHere(2);
        //
        // Global query
        // Global query MUST be associated with a defined and
deployed query.
        // This can be done via the Designer.

        globalQueryTestCase();
    }
    else if(_service == NOTIFICATION)
    {
        System.out.println("Calling Notification endpoint");
        NotificationTest notificationTest = new
NotificationTest();
        //
        // Email Notification
        notificationTest.emailNotificationTestCase();
    }
    else if(_service == RESOURCE)
    {
        System.out.println("Calling Resource endpoint");
    }
    else
    {
        System.out.println("Unrecognized service selection");
    }
}

public void globalQueryTestCase()
    throws Exception
    {

System.out.println("\n<=====queryAttributesTestCase=====>");
        try
        {
            //
            // Get the vdx stub
            IRemoteVdx vdxStub = getVdxStub(url, username, password);
            //
            // Create entry items corresponding to param key in DAL
            StringEntry [] entry = {
                new StringEntry("titleattribute", "Chief Operating
Officer"),
                new StringEntry("managerattribute",
"cn=jmiller,ou=users,ou=idmsample-pproto,o=netiq")
            };
            //
            // Create and set the array of entries (key,value pairs)
            StringEntryArray entryArr = new StringEntryArray();
            entryArr.setStringentry(entry);
            //
            // Create and set the map using the entries
            StringMap map = new StringMap();
            map.setEntries(entryArr);

```

```

        //
        // Define and execute the global query
        int QUERY_KEY_INDEX = 0;
        String [] queryKeyName = {"TestVdxGlobalQuery2",
"TestVdxGlobalQuery"};
        //
        // Results from global query TestVdxGlobalQuery2 ----->
cn=apalani,ou=users,OU=idmsample-pproto,O=netiq
        //
        // Make the vdx endpoint call
        StringArray array =
vdxStub.globalQuery(queryKeyName[QUERY_KEY_INDEX], map);
        String [] str = array.getString();
        if(str == null)

            throw new Exception("Global query returns null for key
name " + queryKeyName);
        else
        {
            System.out.println("Results for global query : " +
queryKeyName[QUERY_KEY_INDEX]);

System.out.println("=====  

====");

            for(int index = 0; index < str.length; index++)
            {
                System.out.println(str[index]);
            }
        }
        catch(VdxServiceException error)
        {
            System.out.println(error.getReason() );
            throw new Exception(error.getReason() );
        }
        catch(RemoteException error)
        {
            System.out.println(error.getMessage() );
            throw new Exception(error.getMessage() );
        }
    }

    public void queryAttributesTestCase()
    throws Exception
    {
        System.out.println("\nCalling queryAttributesTestCase() test
case");
        try
        {
            IRemoteVdx vdxStub = getVdxStub(url, username, password);

            StringArray attributes = new StringArray();
            attributes.setString(new String[]{"FirstName", "Title",
"UserPhoto", "Department"});
            String expression1 = "FirstName STARTWITH 'J'";
            String expression2 = "Title = 'Controller'";
            String expression3 = "vdxInteger > 0";
            String expression4 = "TelephoneNumber != '(555) 555-1201'";
            //

```

```

        // Test Cases
        // expression1 --> Should yield all users whose firstname
starts with J
        // expression1 AND expression2 --> Should yield jkelley who
is the Controller
        // expression1 AND expression3 --> Should yield only jmiller
        // expression1 AND expression4 --> Should yield all users
starting with J EXCEPT jmiller
        String finalExpression = expression1 + " AND " +
expression2;
        //
        // Make the vdx endpoint call
        EntityAttributeMap map = vdxStub.query("user", attributes,
finalExpression);
        EntryArray entryArray = map.getEntries();
        Entry [] entries = entryArray.getEntry();
        if(entries != null)
        {
            for(int index = 0; index < entries.length; index++)
            {
                String dnKey = entries[index].getKey();
                System.out.println("DN Key = " + dnKey);
                AttributeArray attributeArray =
entries[index].getValues();
                Attribute [] attributeData =
attributeArray.getAttribute();
                for(int attrIndex = 0; attrIndex <
attributeData.length; attrIndex++)
                {
                    //
                    // Determine how to handle the return data
                    examineAttributeData(attributeData[attrIndex],
" ");
                }
            }
        }
        catch(VdxServiceException error)
        {
            System.out.println(error.getReason() );
            throw new Exception(error.getReason() );
        }
        catch(RemoteException error)
        {
            System.out.println(error.getMessage() );
            throw new Exception(error.getMessage() );
        }
    }

    public void getVersionTestCase()
    throws Exception
    {
        System.out.println("\nCalling getVersionTestCase() test
case");
    }

    try
    {
        IRemoteVdx vdxStub = getVdxStub(url, username, password);
    }

```

```

        VersionVO version = vdxStub.getVersion();
        System.out.println("Version : " + version.getValue() );
    }

    catch(RemoteException error)
    {
        System.out.println(error.getMessage() );
        throw new Exception(error.getMessage() );
    }
}

public void getAttributeTestCase()
throws Exception
{
    System.out.println("\nCalling getAttributeTestCase() test
case");

    try
    {
        IRemoteVdx vdxStub = getVdxStub(url, username, password);

        String recipient =
"cn=jmiller,ou=users,ou=idmsample,o=netiq";
        String entity = "user";
        for(int attributeIndex = 0; attributeIndex <
userAttributes.length; attributeIndex++)
        {
            //
            // Now, get the values for each attribute from the VDX
layer
            Attribute attributeData =
vdxStub.getAttribute(recipient,
                entity, userAttributes[attributeIndex]);
            //
            // Determine how to handle the return data
            examineAttributeData(attributeData,
userAttributes[attributeIndex]);
        }
    }
    catch(VdxServiceException error)
    {
        System.out.println(error.getReason() );
        throw new Exception(error.getReason() );
    }
    catch(RemoteException error)
    {
        System.out.println(error.getMessage() );
        throw new Exception(error.getMessage() );
    }
}

public void getAttributesTestCase()
throws Exception
{
    System.out.println("\nCalling getAttributesTestCase() test
case");

    try
    {
        IRemoteVdx vdxStub = getVdxStub(url, username, password);

```

```

        String recipient =
"cn=jmiller,ou=users,ou=idmsample,o=netiq";
        String entity = "user";
        StringArray userAttributesArray = new
StringArray(userAttributes);
        AttributeArray attributeArray =
vdxStub.getAttributes(recipient,
entity, userAttributesArray);
        Attribute [] attributeData = attributeArray.getAttribute();
        for(int index = 0; index < attributeData.length; index++)
        {
            //
            // Determine how to handle the return data
            examineAttributeData(attributeData[index],
userAttributes[index]);
        }
    }
    catch(VdxServiceException error)
    {
        System.out.println(error.getReason() );
        throw new Exception(error.getReason() );
    }
    catch(RemoteException error)
    {
        System.out.println(error.getMessage() );
        throw new Exception(error.getMessage() );
    }
}

private void examineAttributeData(Attribute _attribute, String _attributeName)
throws Exception
{
    AttributeType type = _attribute.getType();
    System.out.println("Attribute type : " + type);
    //
    // What type are we dealing with?
    if(type.getValue().compareTo(AttributeType._Integer) == 0)
    {
        IntegerArray intArray = _attribute.getIntegers();
        int [] intData = intArray.getInt();
        if(intData == null)
            System.out.println(_attributeName + " attribute : " +
"null because no attribute value exists.");
        else
        {
            for(int intIndex = 0; intIndex < intData.length;
intIndex++)
            {
                System.out.println(_attributeName + " attribute : "
+ intData[intIndex]);
            }
        }
    }
    else if(type.getValue().compareTo(AttributeType._Boolean) == 0)
    {
        BooleanArray boolArray = _attribute.getBooleans();
        boolean [] booleanData = boolArray.getBoolean();
        if(booleanData == null)

```



```

        }
        catch(IOException ioe)
        {
            throw new Exception(ioe.getMessage());
        }
    }
}
else if(type.getValue().compareTo(AttributeType._Time) == 0)
{
    DateArray dateArray = _attribute.getDates();
    Calendar [] calendar = dateArray.getDatetime();
    if(calendar == null)
        System.out.println(_attributeName + " attribute : " +
"null because no attribute value exists.");
    else
    {
        for(int calIndex = 0; calIndex < calendar.length;
calIndex++)
        {
            System.out.println(_attributeName + " attribute : "
+ calendar[calIndex].getTime().toString());
        }
    }
}

/**
 * Method to obtain the remote interface to the Vdx endpoint
 * @param _url
 * @param _username
 * @param _password
 * @return IRemoteMetrics interface
 * @throws Exception
 */
private IRemoteVdx getVdxStub(String _url, String _username, String
_password)
throws Exception
{
    Properties properties = new Properties();
    properties.put(Context.INITIAL_CONTEXT_FACTORY,
"org.jnp.interfaces.NamingContextFactory");

    String lookup =
"xmlrpc:soap:com.netiq.ws.client.vdx.VdxService";

    InitialContext ctx = new InitialContext();

```

```
VdxService svc = (VdxService) ctx.lookup(lookup);

Stub stub = (Stub)svc.getIRemoteVdxPort();

stub._setProperty(Stub.USERNAME_PROPERTY, _username);
stub._setProperty(Stub.PASSWORD_PROPERTY, _password);
stub._setProperty(Stub.SESSION_MAINTAIN_PROPERTY,
Boolean.TRUE);
stub._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY, _url);

return (IRemoteVdx) stub;
}
}
```

15 Role Web Service

This section describes the Role Web Service, which allows SOAP clients to access the role management and SoD management functions.

15.1 About the Role Web Service

To support access by third-party software applications, the Role subsystem includes a Web service endpoint called the Role Web Service. It supports a wide range of role management and SoD management functions.

This Appendix describes the programming interface for the Role Web Service.

15.1.1 Accessing the Test Page

You can access the Role Web Service endpoint using a URL similar to the following:

```
http://server:port/warcontext/role/service?test
```

For example, if your server is named “myserver”, your User Application is listening on port 8080, and your User Application war file is named “IDMPROV”, the URL would be:

```
http://myserver:8080/IDMPROV/role/service?test
```

You can also access the SOAP endpoint by going to the **Administration** within the User Application. To do this, you need to select the **Application Configuration** tab, then select **Web Services** from the left-navigation menu. After selecting **Web Services**, pick the Web Service endpoint you want from the list.

WARNING: The test page is disabled by default. Since some of the methods allow data to be updated, the test page presents a potential security vulnerability and should not be allowed in a production environment.

Servlet Declaration for the Test Page

A SOAP service using WSSDK is deployed by adding the following declarations in the deployment descriptor (i.e. WEB-INF/web.xml):

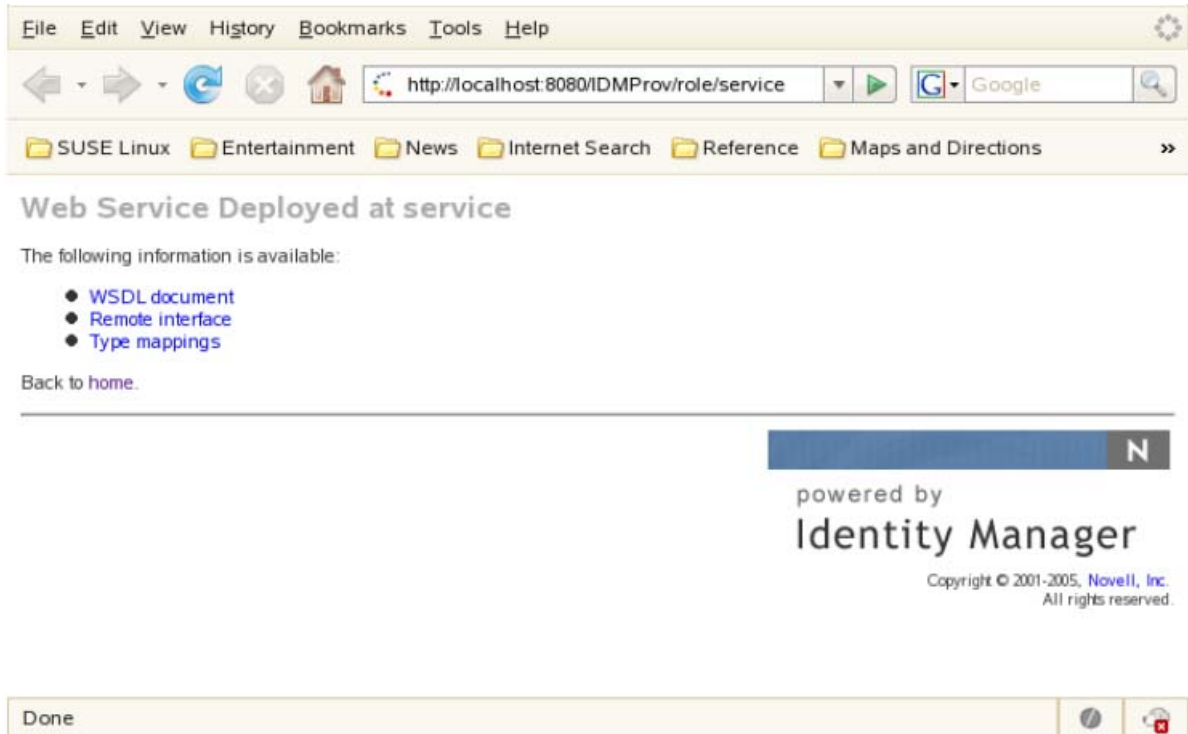
```
<servlet>
  <servlet-name>Role</servlet-name>
  <servlet-class>com.novell.idm.nrf.soap.ws.role.impl.RoleServiceSkeletonImpl</
servlet-class>

<servlet-mapping>
  <servlet-name>Role</servlet-name>
  <url-pattern>/role/service</url-pattern>
</servlet-mapping>
</servlet>
```

This follows the normal servlet declaration pattern. It indicates that the servlet `com.novell.idm.nrf.soap.ws.role.impl.RoleServiceSkeletonImpl` is deployed at `/role/service`.

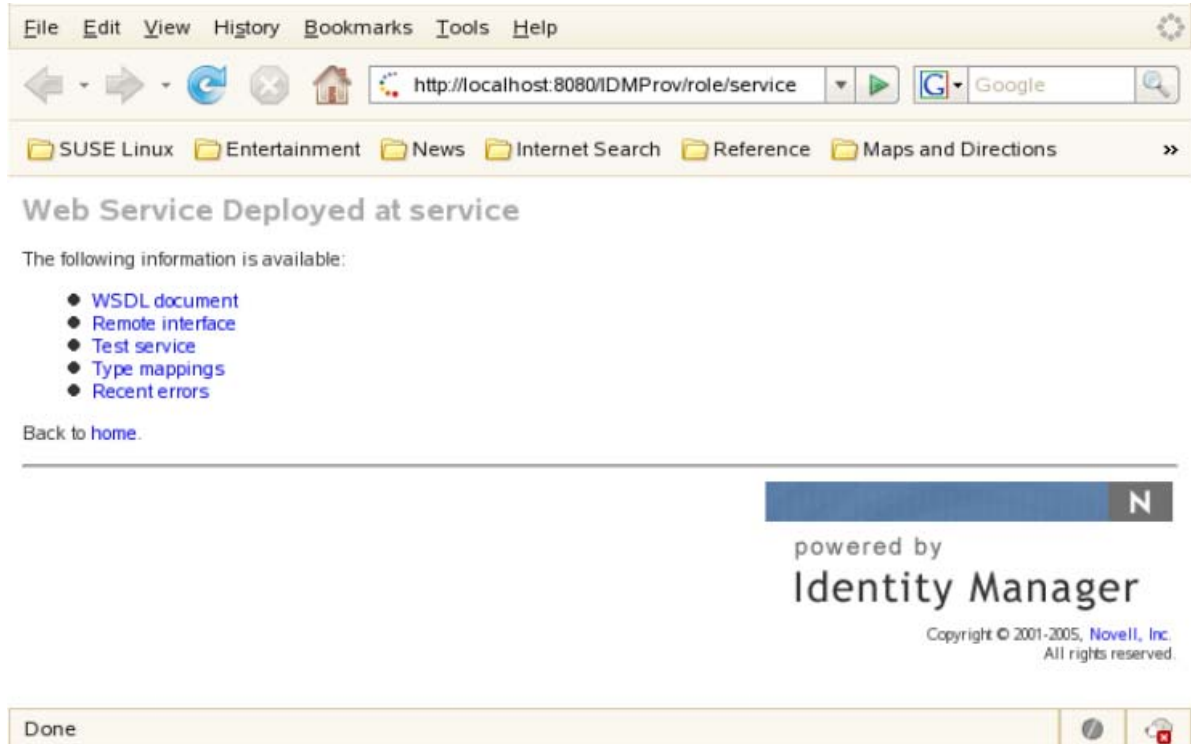
When a user reaches this servlet using a HTTP GET by entering `http://server-name/context/role/service` (for example, `http://localhost:8080/IDMProv/role/service`) in their browser, the WSSDK provides a page that exposes some information about the deployed service. By default the page looks like this:

Figure 15-1 SOAP Service with Test Page Disabled



After you enable the test page, the **Test Service** link is available:

Figure 15-2 SOAP Servlet with Test Page Enabled



On the test page, the user can retrieve the WSDL document that describes the Web Service, see the Java Remote Interface that represents the service, and also see the type mappings from XML to Java. In addition, the user can test the service by invoking individual methods.

Enabling the Test Page

WARNING: The test page is disabled by default. Since some of the methods allow data to be updated, the test page presents a potential security vulnerability and should not be allowed in a production environment.

To enable the test page, you need to update the WEB-INF/web.xml file in the IDMProv.war file. Before you make your changes, the web.xml should look like this:

```
<servlet>
  <servlet-name>Role</servlet-name>
  <servlet-class>com.novell.idm.nrf.soap.ws.role.impl.RoleServiceSkeletonImpl</
servlet-class>
  <init-param>
    <param-name>com.novell.soa.ws.test.disable</param-name>
    <param-value>>true</param-value>
  </init-param>
</servlet>
```

Change the servlet declaration, as follows:

```

<servlet>
  <servlet-name>Role</servlet-name>
  <servlet-class>com.novell.idm.nrf.soap.ws.role.impl.RoleServiceSkeletonImpl</
servlet-class>
</servlet>

```

15.1.2 Accessing the WSDL

You can access the WSDL for the Role Web Service using a URL similar to the following:

```
http://server:port/warcontext/role/service?wsdl
```

For example, if your server is named “myserver”, your User Application is listening on port 8080, and your User Application war file is named “IDMPROV”, the URL would be:

```
http://myserver:8080/IDMPROV/role/service?wsdl
```

15.1.3 Generating the Stub Classes

Before using the Web Service, you need to use the WSSDK tool or another SOAP tool kit to generate the stub classes. To allow your code to find the stub classes, you also need to add the JAR that contains the stub classes to your classpath.

If you want to use the NetIQ WSSDK tool, you can generate the client stubs by extracting the WSDL and running the `wsdl2java` utility. For example, you could run this command to generate the stubs in a package called `com.novell.soa.af.role.soap.impl`:

```

"C:\Program Files\Java\jdk1.6.0_31\bin\java" -cp "../lib/wssdk.jar;../lib/jaxrpc-
api.jar";"../lib/mail.jar";"../lib/activation.jar";"c:\Program
Files\Java\jdk1.6.0_31\lib\tools.jar"; com.novell.soa.ws.impl.tools.wsdl2java.Main
-verbose -ds gensrc -d C:\ -noskel -notie -genclient -keep -package
com.novell.soa.af.role.soap.impl -javadoc role.wsdl

```

You can change the `wsdl2java` parameters to suit your requirements.

15.1.4 Removing Administrator Credential Restrictions

The Role Web Service supports two levels of security, one that restricts access to Role Administrators, and another that restricts access to the authenticated user. The default setting restricts access to all operations to the Role Administrator.

You can modify the settings for security configuration in the `ism-configuration.properties` file, located by default in the `/netiq/idm/apps/tomcat/conf` directory. Each property can be set to true or false. A value of true locks down the operation, whereas a value of false opens up the operation. After making your changes, you need to import the file back into the WAR file and redeploy.

You can open up the Role Web Service to authenticated users by setting the `RoleService/Role/soap` property to false. To open up a particular operation to authenticated users, you need to set the property for that operation (`RoleService/Role/soap/operation`) to false as well. If you set all of the properties to false, you can open up all operations to authenticated users. The *operation* names are the same as the names of the methods supported by the service.

Example To ensure that the security configuration opens up all operations within the Role Web Service, the `ism-configuration.properties` file must have the following setting:

```
RoleService/Role/soap = false
```


15.2 Role API

This section provides details about the methods available with the Role Web service. This API presumes you're using Java code generated by the WSSDK toolkit. The API will be different if you're using another Web Service toolkit.

15.2.1 IRemoteRole

This section provides reference information for each method associated with the IRemoteRole interface.

createResourceAssociation

Create a resource association and return the resource association object with the newly created resource association DN.

Syntax: Here is the method signature:

```
ResourceAssociation
createResourceAssociation(com.novell.idm.nrf.soap.ws.ResourceAssociation
resourceAssociation)
    throws com.novell.idm.nrf.soap.ws.NrfServiceException,
java.rmi.RemoteException;
```

deleteResourceAssociation

Deletes a resource association object.

Syntax: Here is the method signature:

```
void deleteResourceAssociation(com.novell.idm.nrf.soap.ws.DNString
resourceAssociationDn)
    throws com.novell.idm.nrf.soap.ws.NrfServiceException,
java.rmi.RemoteException;
```

getResourceAssociations

Retrieves resource association objects for a given role DN or resource DN. If the *roleDn* and *resourceDn* parameters are null, the entire list is returned.

Syntax: Here is the method signature:

```
ResourceAssociation[] getResourceAssociations(com.novell.idm.nrf.soap.ws.DNString
roleDn, com.novell.idm.nrf.soap.ws.DNString resourceDn)
    throws com.novell.idm.nrf.soap.ws.NrfServiceException,
java.rmi.RemoteException;
```

createRole

Creates a new role according to the specified parameters and returns the DN of the created role.

A correlation ID is generated automatically for this method that uses this format:

```
UserApp#RemoteRoleRequest#xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx
```

The correlation ID is used for auditing.

Syntax: Here is the method signature:

```
public DNString createRole(RoleRequest role)
    throws NrfServiceException, RemoteException;
```

createRoleAid

Creates a new role with a correlation ID that you provide. The correlation ID is used for auditing to link a set of related roles. This method returns the DN of the created role.

Syntax: Here is the method signature:

```
public DNString createRoleAid (RoleRequest role, String correlationId)
    throws NrfServiceException, RemoteException;
```

findRoleByExampleWithOperator

Finds an array of Role objects based on the search criteria specified in the given Role object. This method also lets you specify whether to use AND as the operator for multi-value searches.

Syntax: Here is the method signature:

```
RoleArray findRoleByExampleWithOperator(Role searchCriteria, boolean
useAndForMultiValueSearch) throws NrfServiceException, java.rmi.RemoteException
```

This method follows a query by example approach. It allows you to populate a Role object to specify the desired search criteria. An AND operation is always used across multiple attributes within the Role search object. For example, you might provide a value for the `name` and `description` attributes, which indicates that the criteria for both attributes must be satisfied for a successful search.

The second parameter (`useAndForMultiValueSearch`) allows you to specify which operator should be used for multi-valued attributes (such as when multiple child roles are provided). A value of `true` indicates that AND should be used for these operations, whereas a value of `false` indicates that OR should be used.

Not all attributes in the Role object can be used for the search expression. Values found in the non-supported search attributes are ignored.

Table 15-1 Guidelines for Defining Search Criteria in the Role Object

Attribute	Supported?	Description
approvers	Yes	<p>Uses a standard LDAP equal operator for the search. You can enter multiple approvers and use the operator parameter to determine whether an AND or an OR is used for the multi-valued search. You need to provide valid Dns for the approvers. Note that an approver is made up of multiple parts. It is of type TypedNameSyntax. You need to specify the sequence number of the approver to execute a successful search. This is a limitation in LDAP.</p> <p>Sample SOAP Request:</p> <pre data-bbox="690 556 1333 892"> <ser:findRoleByExampleWithOperatorRequest> <ser:role> <ser:approvers> <!--Zero or more repetitions:--> <ser:approver> <ser:approverDN>cn=ablake,ou=users,ou=medical- idmsample,o=netiq</ser:approverDN> <ser:sequence>1</ser:sequence> </ser:approver> </ser:approvers> </ser:role> <ser:operator>false</ser:operator> </ser:findRoleByExampleWithOperatorRequest> </pre> <p>The example above shows how to find roles that have the specified approver associated with them. An OR search is used since the operator parameter is set to false.</p>
childRoles	Yes	<p>Uses a standard LDAP equal operator for the search. You can enter multiple child roles and use the operator parameter to determine whether an AND or an OR is used for the multi-valued search. You need to provide valid Dns for the child roles.</p> <p>Sample SOAP Request:</p> <pre data-bbox="690 1220 1438 1690"> <ser:findRoleByExampleWithOperatorRequest> <ser:role> <ser:childRoles> <!--Zero or more repetitions:--> <ser:dnstring> <ser:dn>cn=Doctor,cn=Level20,cn=RoleDefs,cn=RoleConfig ,cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=netiq</ ser:dn> </ser:dnstring> <ser:dnstring> <ser:dn>cn=Nurse,cn=Level20,cn=RoleDefs,cn=RoleConfig, cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=netiq</ ser:dn> </ser:dnstring> </ser:childRoles> </ser:role> <ser:operator>false</ser:operator> </ser:findRoleByExampleWithOperatorRequest> </pre> <p>The example above shows how to find roles with a child role of "Doctor" or "Nurse. An OR search is used since the operator parameter is set to false.</p>

Attribute	Supported?	Description
description	Yes	<p>Uses an LDAP contains search. All entries are prefixed and suffixed with the * (wild card character). Therefore, a search for "Doctor" translates to "**Doctor*". This is to accommodate searches across any localized language.</p> <p>Sample SOAP Request:</p> <pre><ser:findRoleByExampleWithOperatorRequest> <ser:role> <ser:description>Doctor</ser:description> </ser:role> <ser:operator>>false</ser:operator> </ser:findRoleByExampleWithOperatorRequest></pre> <p>The example above shows how to find roles with a description of "Doctor". This description string results in a search string of "**Doctor*".</p>
entityKey	Yes	<p>If entered, this attribute causes a getRole operation to be performed. All other search criteria are ignored in this case.</p> <p>Sample SOAP Request:</p> <pre><ser:findRoleByExampleWithOperatorRequest> <ser:role> <ser:entityKey>cn=Doctor,cn=Level20,cn=RoleDefs,cn=RoleC onfig,cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=netiq </ser:entityKey> </ser:role> <ser:operator>>false</ser:operator> </ser:findRoleByExampleWithOperatorRequest></pre> <p>The example above shows how to retrieve a role with a specific entity key.</p>
implicitContainers	Yes	<p>Uses a standard LDAP equal operator for the search. You can enter multiple implicit containers and use the operator parameter to determine whether an AND or an OR will be used for the multi-valued search. You need to provide valid Dns for the implicit containers.</p> <p>Sample SOAP Request:</p> <pre><ser:findRoleByExampleWithOperatorRequest> <ser:role> <ser:implicitContainers> <!--Zero or more repetitions:--> <ser:dnstring> <ser:dn>ou=medical-idmsample,o=netiq</ser:dn> </ser:dnstring> </ser:implicitContainers> </ser:role> <ser:operator>>false</ser:operator> </ser:findRoleByExampleWithOperatorRequest></pre> <p>The example above shows how to find roles that have the specified implicit container associated with them. An OR search is used since the operator parameter is set to false.</p>

Attribute	Supported?	Description
implicitGroups	Yes	<p>Uses a standard LDAP equal operator for the search. You can enter multiple implicit groups and use the operator parameter to determine whether an AND or an OR will be used for the multi-valued search. You need to provide valid Dns for the implicit groups.</p> <p>Sample SOAP Request:</p> <pre><ser:findRoleByExampleWithOperatorRequest> <ser:role> <ser:implicitGroups> <!--Zero or more repetitions:--> <ser:dnstring> <ser:dn>cn=HR,ou=groups,ou=medical-idmsample,o=netiq</ ser:dn> </ser:dnstring> </ser:implicitGroups> </ser:role> <ser:operator>>false</ser:operator> </ser:findRoleByExampleWithOperatorRequest></pre> <p>The example above shows how to find roles that have the specified implicit group associated with them. An OR search is used since the operator parameter is set to false.</p>
name	Yes	<p>Uses an LDAP contains search. All entries will be prefixed and suffixed with the * (wild card character). Therefore, a search for "Doctor" translates to "**Doctor*". This is to accommodate searches across any localized language.</p> <p>Sample SOAP Request:</p> <pre><ser:findRoleByExampleWithOperatorRequest> <ser:role> <ser:name>Doctor</ser:name> </ser:role> <ser:operator>>false</ser:operator> </ser:findRoleByExampleWithOperatorRequest></pre> <p>The above example shows how to find roles with a name of "Doctor". The name string results in a search string of "**Doctor*"</p>

Attribute	Supported?	Description
owners	Yes	<p>Uses a standard LDAP equal operator for the search. You can enter multiple owners and use the operator parameter to determine whether an AND or an OR is used for the multi-valued search. You must provide valid Dns for the owners.</p> <p>SoapUI Example Request:</p> <pre><ser:findRoleByExampleWithOperatorRequest> <ser:role> <ser:owners> <!--Zero or more repetitions:--> <ser:dnstring> <ser:dn>cn=ablake,ou=users,ou=medical- idmsample,o=netiq</ser:dn> </ser:dnstring> <ser:dnstring> <ser:dn>cn=mmackenzie,ou=users,ou=medical- idmsample,o=netiq</ser:dn> </ser:dnstring> </ser:owners> </ser:role> <ser:operator>true</ser:operator> </ser:findRoleByExampleWithOperatorRequest></pre> <p>The example above shows how to find roles that have the specified owners. An AND search is used since the operator parameter is set to true.</p>
parentRoles	Yes	<p>Uses a standard LDAP equal operator for the search. You can enter multiple parent roles and use the operator parameter to determine whether an AND or an OR is used for the multi-valued search. You must provide valid Dns for the parent roles.</p> <p>Sample SOAP Request:</p> <pre><ser:findRoleByExampleWithOperatorRequest> <ser:role> <ser:parentRoles> <!--Zero or more repetitions:--> <ser:dnstring> <ser:dn>cn=Doctor- East,cn=Level30,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn= PicassoDriver,cn=TestDrivers,o=netiq</ser:dn> </ser:dnstring> <ser:dnstring> <ser:dn>cn=Doctor- West,cn=Level30,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn= PicassoDriver,cn=TestDrivers,o=netiq</ser:dn> </ser:dnstring> </ser:parentRoles> </ser:role> <ser:operator>true</ser:operator> </ser:findRoleByExampleWithOperatorRequest></pre> <p>The example above shows how to find roles that have the specified parent roles. An AND search is used since the operator parameter is set to true.</p>

Attribute	Supported?	Description
quorum	Yes	<p>Uses a standard LDAP equal operator for the search.</p> <p>Sample SOAP Request:</p> <pre><ser:findRoleByExampleWithOperatorRequest> <ser:role> <ser:quorum>50%</ser:quorum> </ser:role> <ser:operator>>false</ser:operator> </ser:findRoleByExampleWithOperatorRequest></pre> <p>The example above shows how to find roles with the specified quorum search string. The search string can include the wild card character ("*").</p>
requestDef	Yes	<p>Uses a standard LDAP equal operator for the search. You must provide a valid DN for the request definition.</p> <p>Sample SOAP Request:</p> <pre><ser:findRoleByExampleWithOperatorRequest> <ser:role> <ser:requestDef>cn=Role Approval,cn=RequestDefs,cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=netiq</ser:requestDef> </ser:role> <ser:operator>>false</ser:operator> </ser:findRoleByExampleWithOperatorRequest></pre> <p>The example above shows how to find roles with the specified request definition DN.</p>
roleCategoryKeys	Yes	<p>Uses a standard LDAP equal operator for the search. You can enter multiple category keys and use the operator parameter to determine whether an AND or an OR is used for the multi-valued search.</p> <p>Sample SOAP Request:</p> <pre><ser:findRoleByExampleWithOperatorRequest> <ser:role> <ser:roleCategoryKeys> <!--Zero or more repetitions:--> <ser:categorykey> <ser:categoryKey>doctor</ser:categoryKey> </ser:categorykey> <ser:categorykey> <ser:categoryKey>nurse</ser:categoryKey> </ser:categorykey> </ser:roleCategoryKeys> </ser:role> <ser:operator>>false</ser:operator> </ser:findRoleByExampleWithOperatorRequest></pre> <p>The example above shows how to find roles with a category of "doctor" or "nurse". An OR search is used since the operator parameter is set to false.</p>

Attribute	Supported?	Description
roleLevel	Yes	<p>Uses a standard LDAP equal operator for the search. You can only enter one level at a time.</p> <p>Sample SOAP Request:</p> <pre><ser:findRoleByExampleWithOperatorRequest> <ser:role> <ser:roleLevel> <ser:level>10</ser:level> </ser:roleLevel> </ser:role> <ser:operator>false</ser:operator> </ser:findRoleByExampleWithOperatorRequest></pre> <p>The example above shows how to find all level 10 roles.</p>
associatedRoles	No	Not supported.
entitlementRef	No	Not supported.
roleAssignments	No	Not supported.
systemRole	No	Not supported.

findSodByExample

Finds all SoD objects based on the search criteria in the given SOD object.

Syntax: Here is the method signature:

```
SodArray findSodByExample(Sod sod) throws NrfServiceException,
java.rmi.RemoteException
```

findSodByExampleWithOperator

Finds all SoD objects based on the search criteria found in the given SOD object. This method also lets you specify whether to use And as the operator for multi-value searches.

Syntax: Here is the method signature:

```
SodArray findSodByExampleWithOperator(Sod searchCriteria, boolean
useAndForMultiValueSearch) throws NrfServiceException, java.rmi.RemoteException
```

findSodById

Find by key.

Syntax: Here is the method signature:

```
Sod findSodById(java.lang.String entityKey) throws NrfServiceException,
java.rmi.RemoteException
```

getAssignedIdentities

Returns returns the list of identities having a particular role DN.

Syntax: Here is the method signature:


```
RoleAssignment[] getAssignedIdentities(java.lang.String roleDN, IdentityType
identityType, boolean directAssignOnly)
```

getConfigProperty

Retrieves configuration properties stored in the User Application configuration XML files by passing in a configuration property key or macro name.

Syntax: Here is the method signature:

```
public ConfigProperty getConfigProperty(String configPropertyKey) throws
NrfServiceException, RemoteException;
```

The configPropertyKey parameter can accept a fully qualified configuration key name from any of the configuration XML files, such as the following:

```
DirectoryService/realms/jndi/params/USER_ROOT_CONTAINER
```

Alternatively, the configPropertyKey parameter can accept a macro name that references a fully qualified configuration key name. The following macro names are allowed:

Table 15-2 Macro Names Allowed

Configuration Macro Name	Configuration Key Value
USER_CONTAINER	DirectoryService/realms/jndi/params/ USER_ROOT_CONTAINER
GROUP_CONTAINER	DirectoryService/realms/jndi/params/ GROUP_ROOT_CONTAINER
ROOT_CONTAINER	DirectoryService/realms/jndi/params/ROOT_NAME
PROVISIONING_DRIVER	DirectoryService/realms/jndi/params/ PROVISIONING_ROOT

getConfiguration

Returns the role system configuration defined in the Role Catalog root (nrfConfiguration).

Syntax: Here is the method signature:

```
Configuration getConfiguration() throws NrfServiceException,
java.rmi.RemoteException
```

getContainer

Gets container and role information for a given container DN.

Syntax: Here is the method signature:

```
Container getContainer(java.lang.String containerDn) throws
NrfServiceException, java.rmi.RemoteException
```

getExceptionList

Returns a list of Sod instances for all SOD violations found for a specific identity and type.

Syntax: Here is the method signature:

```
SodArray getExceptionsList(java.lang.String identity, IdentityType identityType)
throws NrfServiceException, java.rmi.RemoteException
```

getGroup

Gets group and role information for a given group DN.

Syntax: Here is the method signature:

```
Group getGroup(java.lang.String groupDn) throws NrfServiceException,
java.rmi.RemoteException
```

getIdentitiesInViolation

Returns a map of identities which are in violation of a given SoD.

Syntax: Here is the method signature:

```
IdentityTypeDnMapArray getIdentitiesInViolation(java.lang.String sodDn) throws
NrfServiceException, java.rmi.RemoteException
```

getIdentityRoleConflicts

Returns a list of Sod instances for all SOD conflicts found for a given list of roles for a given identity.

Syntax: Here is the method signature:

```
SodArray getIdentityRoleConflicts(java.lang.String identity, IdentityType
identityType, DNStringArray requestedRoles) throws NrfServiceException,
java.rmi.RemoteException
```

getRole

Retrieves a role object defined by a role DN. Returns several role attributes, such as name, dn, description, role level. Returns child roles, assigned containers, and assigned groups. However, this API does not return assigned users. If you want assigned users, use the `getAssignedIdentities` API with `USER` for `identityType` and `true` for `directAssignOnly`.

Syntax: Here is the method signature:

```
Role getRole(java.lang.String roleDn) throws NrfServiceException,
java.rmi.RemoteException
```

getRoleAssignmentRequestStatus

Returns a list of role assignment request status instances given a correlation ID.

Syntax: Here is the method signature:

```
RoleAssignmentRequestStatusArray getRoleAssignmentRequestStatus(java.lang.String
correlationId) throws NrfServiceException, java.rmi.RemoteException
```

getRoleAssignmentRequestStatusByIdentityType

Returns a list of role assignment request status instances given an identity and an identity type.

Syntax: Here is the method signature:

```
RoleAssignmentRequestStatusArray  
getRoleAssignmentRequestStatusByIdentityType(java.lang.String identityDn,  
IdentityType identityType) throws NrfServiceException, java.rmi.RemoteException
```

getRoleAssignmentTypeInfo

Retrieves details about a RoleAssignmentType.

Syntax: Here is the method signature:

```
RoleAssignmentTypeInfo getRoleAssignmentTypeInfo(RoleAssignmentType type) throws  
NrfServiceException, java.rmi.RemoteException
```

getRoleCategories

Gets role categories.

Syntax: Here is the method signature:

```
CategoryArray getRoleCategories() throws NrfServiceException,  
java.rmi.RemoteException
```

getRoleConflicts

Returns a list of Sod instances found for all given roles. This method always returns a list.

Syntax: Here is the method signature:

```
SodArray getRoleConflicts(DNStringArray roles) throws NrfServiceException,  
java.rmi.RemoteException
```

getRoleLevels

Gets the role levels.

Syntax: Here is the method signature:

```
RoleLevelArray getRoleLevels() throws NrfServiceException,  
java.rmi.RemoteException
```

getRoleLocalizedStrings

Gets role localized strings, such as names and descriptions. The method takes an integer parameter that allows you to specify the type of the string. The number 1 indicates names; the number 2 indicates descriptions.

Syntax: Here is the method signature:

```
public LocalizedValue[] getRoleLocalizedStrings(DNString roleDn, int type)  
    throws NrfServiceException, RemoteException;
```

getRolesInfo

Returns a list of RoleInfo instances given a list of role DNs.

Syntax: Here is the method signature:

```
RoleInfoArray getRolesInfo(DNStringArray roleDns) throws NrfServiceException,  
java.rmi.RemoteException
```

getRolesInfoByCategory

Returns a list of RoleInfo instances given a list of role category keys.

Syntax: Here is the method signature:

```
RoleInfoArray getRolesInfoByCategory(CategoryKeyArray roleCategoryKeys) throws  
NrfServiceException, java.rmi.RemoteException
```

getRolesInfoByLevel

Returns a list of RoleInfo instances given a list of role levels.

Syntax: Here is the method signature:

```
RoleInfoArray getRolesInfoByLevel(LongArray roleLevels) throws  
NrfServiceException, java.rmi.RemoteException
```

getTargetSourceConflicts

Returns a list of Sod instances for all SOD conflicts defined between the target role DN and the source role DN.

Syntax: Here is the method signature:

```
SodArray getTargetSourceConflicts(java.lang.String targetName, java.lang.String  
sourceName) throws NrfServiceException, java.rmi.RemoteException
```

getUser

Gets user info including all role assignments for a given user DN stored in a UserIdentity object.

Syntax: Here is the method signature:

```
User getUser(java.lang.String userDn) throws NrfServiceException,  
java.rmi.RemoteException
```

getVersion

Returns the version of this Web Service.

Syntax: Here is the method signature:

```
VersionVO getVersion() throws java.rmi.RemoteException
```

isUserInRole

Returns boolean flag; true if role has been assigned to a User identity.

Syntax: Here is the method signature:

```
boolean isUserInRole(java.lang.String userDn, java.lang.String roleDn)
```

modifyRole

Modifies a role definition. This method does not update localized strings. Use the `getRoleLocalizedStrings(DNString roleDn, LocalizedString[] locStrings, int strType)` method to update localized names or descriptions for a role.

A correlation ID is generated automatically for this method that uses this format:

```
UserApp#RemoteRoleRequest#xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

The correlation ID is used for auditing.

Syntax: Here is the method signature:

```
public Role modifyRole(Role role)
    throws NrfServiceException, RemoteException;
```

modifyRoleAid

Modifies a role definition with a correlation ID that you provide. The correlation ID is used for auditing to link a set of related roles. This method does not update localized strings. Use the `getRoleLocalizedStrings(DNString roleDn, LocalizedString[] locStrings, int strType)` method to update localized names or descriptions for a role.

Syntax: Here is the method signature:

```
public Role modifyRoleAid(Role role, String correlationId)
    throws NrfServiceException, RemoteException;
```

removeRoles

Deletes specified roles from the Role Catalog and returns an array of DNs for the deleted roles as a confirmation.

A correlation ID is generated automatically for this method that uses this format:

```
UserApp#RemoteRoleRequest#xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

The correlation ID is used for auditing.

Syntax: Here is the method signature:

```
public DNString[] removeRoles(DNString[] roleDns)
    throws NrfServiceException, RemoteException;
```

removeRolesAid

Deletes specified roles from the Role Catalog with a correlation ID that you provide. The correlation ID is used for auditing to link a set of related roles. This method returns an array of DNs for the deleted roles as a confirmation.

Syntax: Here is the method signature:

```
public DNString[] removeRolesAid(DNString[] roleDns, String correlationId)
    throws NrfServiceException, RemoteException;
```

requestRolesAssignment

Returns a list of request DNs created by the role assignment.

If you do not want to supply date (effective or expiration) for role assignments with the requestRolesAssignment endpoint, then you must remove these two elements from the SOAP call. They must not be included with empty tags:

```
<ser:effectiveDate/>
<ser:expirationDate/>
```

If you want to omit the effective date or the expiration date, a request similar to the following will work:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://www.netiq.com/role/service">
  <soapenv:Header/>
  <soapenv:Body>
    <ser:requestRolesAssignmentRequest>
      <!--Optional:-->
      <ser:assignRequest>
        <ser:actionType>grant</ser:actionType>
        <ser:assignmentType>USER_TO_ROLE</ser:assignmentType>
        <ser:correlationID>testpolina</ser:correlationID>
        <ser:identity>cn=uaadmin,ou=sa,o=data</ser:identity>
        <ser:originator/>
        <ser:reason>test without expiration date</ser:reason>
        <ser:roles>
          <!--Zero or more repetitions:-->
          <ser:dnstring>
            <ser:dn>cn=test2
id,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=User Application
Driver,cn=driverset1,o=system</ser:dn>
          </ser:dnstring>
        </ser:roles>
        <ser:sodOverridesRequested/>
      </ser:assignRequest>
    </ser:requestRolesAssignmentRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

With that said, without the these two elements in the soap request, the request will not validate. It will work, but will not validate.

Syntax: Here is the method signature:

```
DNStringArray requestRolesAssignment(RoleAssignmentRequest roleAssignmentRequest)
throws NrfServiceException, java.rmi.RemoteException
```

setRoleLocalizedStrings

Sets role localized strings, such as names and descriptions.

A correlation ID is generated automatically for this method that uses this format:

```
UserApp#RemoteRoleRequest#xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxxxx
```

The correlation ID is used for auditing.

Syntax: Here is the method signature:

```
public LocalizedValue[] setRoleLocalizedStrings(DNString roleDn, LocalizedValue[]
locStrings, int type)
    throws NrfServiceException, RemoteException;
```

setRoleLocalizedStringsAid

Sets role localized strings, such as name and description, with a correlation ID that you provide. The correlation ID is used for auditing to link a set of related roles.

Syntax: Here is the method signature:

```
public LocalizedValue[] setRoleLocalizedStringsAid(DNString roleDn, String
correlationId, LocalizedValue[] locStrings, int type)
    throws NrfServiceException, RemoteException;
```

15.2.2 Approver

Class to hold the approver information for SOD or normal request approvals.

Approver constructors

The Approver class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
Approver()
```

getApproverDN

Gets the approver DN.

Syntax: Here is the method signature:

```
public java.lang.String getApproverDN()
```

getSequence

Gets the approver sequence.

Syntax: Here is the method signature:

```
public long getSequence()
```

setApproverDN

Sets the approver DN.

Syntax: Here is the method signature:

```
public void setApproverDN(java.lang.String approverDN)
```

setSequence

Sets the approver sequence.

Syntax: Here is the method signature:

```
public void setSequence(long sequence)
```

15.2.3 ApproverArray

This section provides reference information on the ApproverArray class.

ApproverArray constructors

The ApproverArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
ApproverArray()
```

Syntax 2: Here is the syntax for a constructor that takes an array of Attribute objects as a parameter:

```
ApproverArray(Approver[] ApproverVal)
```

getApprover

Returns an array of Approver objects.

Syntax: Here is the method signature:

```
Approver[] getApprover()
```

setApprover

Sets the array of Approver objects associated with the ApproverArray class.

Syntax: Here is the method signature:

```
void setApprover (Approver[] ApproverVal)
```

15.2.4 Category

Class to represent a role category.

Category constructors

The Category class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
Category()
```

getCategoryKey

Gets the category key.

Syntax: Here is the method signature:

```
public java.lang.String getCategoryKey()
```


getCategoryLabel

Gets the category label.

Syntax: Here is the method signature:

```
public java.lang.String getCategoryLabel()
```

setCategoryKey

Sets the category key.

Syntax: Here is the method signature:

```
public void setCategoryKey(java.lang.String categoryKey)
```

setCategoryLabel

Sets the category label.

Syntax: Here is the method signature:

```
public void setCategoryLabel(java.lang.String categoryLabel)
```

15.2.5 CategoryArray

This section provides reference information on the CategoryArray class.

CategoryArray constructors

The CategoryArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
CategoryArray()
```

Syntax 2: Here is the syntax for a constructor that takes an array of Category objects as a parameter:

```
CategoryArray(Category[] CategoryVal)
```

getCategory

Returns an array of Category objects.

Syntax: Here is the method signature:

```
Category[] getCategory()
```

setCategory

Sets the array of Category objects associated with the CategoryArray class.

Syntax: Here is the method signature:

```
void setCategory(Category[] CategoryVal)
```

15.2.6 CategoryKey

Class to hold a Category Key.

CategoryKey constructors

The CategoryKey class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
CategoryKey()
```

Syntax 2: Here is the syntax for a constructor that takes a String as a parameter:

```
CategoryKey(java.lang.String categoryKey)
```

getCategoryKey()

Gets the categoryKey.

Syntax: Here is the method signature:

```
public java.lang.String getCategoryKey()
```

setCategoryKey

Sets the category key.

Syntax: Here is the method signature:

```
public void setCategoryKey(java.lang.String categoryKey)
```

15.2.7 CategoryKeyArray

This section provides reference information on the CategoryKeyArray class.

CategoryKeyArray constructors

The CategoryKeyArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
CategoryKeyArray()
```

Syntax 2: Here is the syntax for a constructor that takes an array of CategoryKey objects as a parameter:

```
CategoryKeyArray(CategoryKey[] CategoryVal)
```

getCategorykey

Returns an array of Category objects.

Syntax: Here is the method signature:

```
CategoryKey[] getCategorykey()
```

setCategorykey

Sets the array of CategoryKey objects associated with the CategoryKeyArray class.

Syntax: Here is the method signature:

```
void setCategorykey(CategoryKey[] CategoryKeyVal)
```

15.2.8 Configuration

Class to represent the configuration object.

Configuration constructors

The Configuration class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
Configuration()
```

getDefaultRequestDef

Gets the default request definition.

Syntax: Here is the method signature:

```
public java.lang.String getDefaultRequestDef()
```

getDefaultSODRequestDef

Gets the default SOD request definition.

Syntax: Here is the method signature:

```
public java.lang.String getDefaultSODRequestDef()
```

getRemovalGracePeriod

Gets the removal grace period.

Syntax: Here is the method signature:

```
public int getRemovalGracePeriod()
```

getReportContainer

Gets the report container.

Syntax: Here is the method signature:

```
public java.lang.String getReportContainer()
```

getRoleLevels

Gets the role levels.

Syntax: Here is the method signature:

```
public RoleLevelArray getRoleLevels()
```

getRoleRequestContainer

Gets the role request container.

Syntax: Here is the method signature:

```
public java.lang.String getRoleRequestContainer()
```

getRolesContainer

Gets the role container.

Syntax: Here is the method signature:

```
public java.lang.String getRolesContainer()
```

getSODApprovers

Gets SOD approvers.

Syntax: Here is the method signature:

```
public ApproverArray getSODApprovers()
```

getSODContainer

Gets the SOD container.

Syntax: Here is the method signature:

```
public java.lang.String getSODContainer()
```

getSODQuorum

Gets the SOD quorum amount.

Syntax: Here is the method signature:

```
public java.lang.String getSODContainer()
```

getSODRequestDef

Gets the SOD request definition.

Syntax: Here is the method signature:

```
public java.lang.String getSODRequestDef()
```

setDefaultRequestDef

Sets the default request definition.

Syntax: Here is the method signature:

```
public void setDefaultRequestDef(java.lang.String defaultRequestDef)
```

setDefaultSODRequestDef

Sets the default SOD request definition.

Syntax: Here is the method signature:

```
public void setDefaultSODRequestDef(java.lang.String defaultSODRequestDef)
```

setRemovalGracePeriod

Sets the removal grace period.

Syntax: Here is the method signature:

```
public void setRemovalGracePeriod(int removalGracePeriod)
```

setReportContainer

Sets the report container.

Syntax: Here is the method signature:

```
public void setReportContainer(java.lang.String reportContainer)
```

setRoleLevels

Sets the role levels.

Syntax: Here is the method signature:

```
public void setRoleLevels(RoleLevelArray roleLevels)
```

setRoleRequestContainer

Sets the role request container.

Syntax: Here is the method signature:

```
public void setRoleRequestContainer(java.lang.String roleRequestContainer)
```

setRolesContainer

Sets the role container.

Syntax: Here is the method signature:

```
public void setRolesContainer(java.lang.String rolesContainer)
```

setSODApprovers

Sets the SoD approvers.

Syntax: Here is the method signature:

```
public void setSODApprovers(ApproverArray sODApprovers)
```

setSODContainer

Sets the SoD container.

Syntax: Here is the method signature:

```
public void setSODContainer(java.lang.String sODContainer)
```

15.2.9 Container

Class to represent a Container object.

Container constructors

The Container class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
Container()
```

getAssociatedRoles

Gets associated roles for this identity.

Syntax: Here is the method signature:

```
public DNStringArray getAssociatedRoles()
```

getEntityKey

Gets identity entity key.

Syntax: Here is the method signature:

```
public java.lang.String getEntityKey()
```

getIdentityType

Gets identity type.

Syntax: Here is the method signature:

```
public IdentityType getIdentityType()
```

getRoleAssignments

Gets role assignments for this identity.

Syntax: Here is the method signature:

```
public RoleAssignmentArray getRoleAssignments()
```

setAssociatedRoles

Sets the associated roles for this identity.

Syntax: Here is the method signature:

```
public void setAssociatedRoles(DNStringArray associatedRoles)
```

setEntityKey

Sets the identity entity key.

Syntax: Here is the method signature:

```
public void setEntityKey(java.lang.String entityKey)
```

setIdentityType

Sets the identity type.

Syntax: Here is the method signature:

```
public void setIdentityType(IdentityType identityType)
```

setRoleAssignments

Sets the role assignments for this identity.

Syntax: Here is the method signature:

```
public void setRoleAssignments(RoleAssignmentArray roleAssignments)
```

15.2.10 DNString

Class to hold a DN.

DNString constructors

The DNString class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
DNString()
```

Syntax 2: Here is the syntax for a constructor that takes a String as a parameter:

```
DNString(java.lang.String dn)
```

getDn

Gets the DN.

Syntax: Here is the method signature:

```
public java.lang.String getDn()
```

setDn

Sets the DN.

Syntax: Here is the method signature:

```
public void setDn(java.lang.String dn)
```

15.2.11 DNStringArray

This section provides reference information on the DNStringArray class.

DNStringArray constructors

The DNStringArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
DNStringArray()
```

Syntax 2: Here is the syntax for a constructor that takes an array of DNString objects as a parameter:

```
DNStringArray(DNString[] DNStringVal)
```

getDnstring

Returns an array of DNString objects.

Syntax: Here is the method signature:

```
DNString[] getDnstring()
```

setDnstring

Sets the array of DNString objects associated with the DNStringArray class.

Syntax: Here is the method signature:

```
void setDnstring(DNString[] DnstringVal)
```

15.2.12 Entitlement

Class to hold Entitlement information.

Entitlement constructors

The Entitlement class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
Entitlement()
```

getEntitlementDn

Gets the entitlement DN.

Syntax: Here is the method signature:

```
public java.lang.String getEntitlementDn()
```


getEntitlementParameters

Gets the entitlement parameters.

Syntax: Here is the method signature:

```
public java.lang.String getEntitlementParameters()
```

setEntitlementDn

Sets the entitlement DN.

Syntax: Here is the method signature:

```
public void setEntitlementDn(java.lang.String entitlementDn)
```

setEntitlementParameters

Sets the entitlement parameters.

Syntax: Here is the method signature:

```
public void setEntitlementParameters(java.lang.String entitlementParameters)
```

15.2.13 EntitlementArray

This section provides reference information on the EntitlementArray class.

EntitlementArray constructors

The EntitlementArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
EntitlementArray()
```

Syntax 2: Here is the syntax for a constructor that takes an array of Entitlement objects as a parameter:

```
EntitlementArray(Entitlement[] EntitlementVal)
```

getEntitlement

Returns an array of Entitlement objects.

Syntax: Here is the method signature:

```
Entitlement[] getEntitlement()
```

setEntitlement

Sets the array of Entitlement objects associated with the EntitlementArray class.

Syntax: Here is the method signature:

```
void setEntitlement(EntitlementArray EntitlementVal)
```

15.2.14 Group

Class to represent a Group object.

Group constructors

The Group class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
Group()
```

getAssociatedRoles

Gets associated roles for this identity.

Syntax: Here is the method signature:

```
public DNStringArray getAssociatedRoles()
```

getDescription

Gets group description.

Syntax: Here is the method signature:

```
public java.lang.String getDescription()
```

getEntityKey

Gets identity entity key.

Syntax: Here is the method signature:

```
public java.lang.String getEntityKey()
```

getIdentityType

Gets identity type.

Syntax: Here is the method signature:

```
public IdentityType getIdentityType()
```

getRoleAssignments

Gets role assignments for this identity.

Syntax: Here is the method signature:

```
public RoleAssignmentArray getRoleAssignments()
```

setAssociatedRoles

Sets the associated roles for this identity.

Syntax: Here is the method signature:

```
public void setAssociatedRoles(DNStringArray associatedRoles)
```

setDescription

Sets the group description.

Syntax: Here is the method signature:

```
public void setDescription(java.lang.String description)
```

setEntityKey

Sets the identity entity key.

Syntax: Here is the method signature:

```
public void setEntityKey(java.lang.String entityKey)
```

setIdentityType

Sets the identity type.

Syntax: Here is the method signature:

```
public void setIdentityType(IdentityType identityType)
```

setRoleAssignments

Sets the role assignments for this identity.

Syntax: Here is the method signature:

```
public void setRoleAssignments(RoleAssignmentArray roleAssignments)
```

15.2.15 IdentityType

An JAX-RPC friendly representation of `com.novell.idm.nrf.api.IdentityType`.

Table 15-3 Field summary

Type	Name
static IdentityType	CONTAINER
static IdentityType	GROUP
static IdentityType	ROLE
static IdentityType	USER

IdentityType constructors

The `IdentityType` class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
IdentityType()
```

Syntax 2: Here is the syntax for a constructor that takes a String as a parameter:

```
IdentityType(java.lang.String value)
```

convertToAPI

Reconstructs an API representation object from an RPC representation.

Syntax: Here is the method signature:

```
public com.novell.idm.nrf.api.IdentityType convertToAPI()
```

convertToRPC

Constructs an RPC friendly representation from an API object.

Syntax: Here is the method signature:

```
public static IdentityType convertToRPC(com.novell.idm.nrf.api.IdentityType type)
```

equals

This is an implementation of equals(). This implementation overrides the equals() method in java.lang.Object.

Syntax: Here is the method signature:

```
public boolean equals(java.lang.Object obj)
```

fromValue

This method is for WSSDK serialization.

Syntax: Here is the method signature:

```
public static IdentityType fromValue(java.lang.String value)
```

getValue

Gets the type.

Syntax: Here is the method signature:

```
public java.lang.String getValue()
```

hashCode

This is an implementation of hashCode(). This implementation overrides the hashCode() method in java.lang.Object.

Syntax: Here is the method signature:

```
public int hashCode()
```

setValue

Sets the type.

Syntax: Here is the method signature:

```
public void setValue(java.lang.String type)
```

toString

Implementation of toString() that returns a string representation of the class.

Syntax: Here is the method signature:

```
public java.lang.String toString()
```

15.2.16 IdentityTypeDnMap

Class to represent DNs grouped by identity type. Used for SOD violations.

IdentityTypeDnMap

The IdentityTypeDnMap class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
IdentityTypeDnMap()
```

Syntax 2: Here is the syntax for a constructor that takes a String as a parameter:

```
IdentityTypeDnMap(IdentityType identityType, DNStringArray dns)
```

getDns

Gets the DNs associated with the identity type.

Syntax: Here is the method signature:

```
public DNStringArray getDns()
```

getIdentityType

Gets identity type (USER, ROLE, GROUP, CONTAINER).

Syntax: Here is the method signature:

```
public IdentityType getIdentityType()
```

setDns

Sets the DNs to associate with the identity type.

Syntax: Here is the method signature:

```
public void setDns(DNStringArray dns)
```

setIdentityType

Sets the identity type (USER, ROLE, GROUP, or CONTAINER).

Syntax: Here is the method signature:

```
public void setIdentityType(IdentityType identityType)
```

15.2.17 IdentityTypeDnMapArray

This section provides reference information on the IdentityTypeDnMapArray class.

IdentityTypeDnMapArray constructors

The IdentityTypeDnMapArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
IdentityTypeDnMapArray()
```

Syntax 2: Here is the syntax for a constructor that takes an array of IdentityTypeDnMap objects as a parameter:

```
IdentityTypeDnMapArray(IdentityTypeDnMap[] IdentityTypeDnMapVal)
```

getIdentitytypednmap

Returns an array of IdentityTypeDnMap objects.

Syntax: Here is the method signature:

```
IdentityTypeDnMap[] getIdentitytypednmap()
```

setidentitytypednmap

Sets the array of IdentityTypeDnMap objects associated with the IdentityTypeDnMapArray class.

Syntax: Here is the method signature:

```
void setidentitytypednmap(IdentityTypeDnMap[] IdentityTypeDnMapVal)
```

15.2.18 LocalizedValue

The LocalizedValue class has been added to support management of localized strings for role definitions.

getValue

Returns a localized string value.

Syntax: Here is the method signature:

```
public String getValue()
```

setValue

Sets a localized string value.

Syntax: Here is the method signature:

```
public void setValue(final String value)
```

getLocale

Returns a string representaton of the Locale object.

Syntax: Here is the method signature:

```
public String getLocale()
```

setLocale

Sets a string representation of the Locale object.

Syntax: Here is the method signature:

```
public void setLocale()
```

15.2.19 LongArray

This section provides reference information on the LongArray class.

LongArray constructors

The LongArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
LongArray()
```

Syntax 2: Here is the syntax for a constructor that takes an array of Long objects as a parameter:

```
LongArray(long[] LongVal)
```

getLong

Returns an array of Long objects.

Syntax: Here is the method signature:

```
long[] getLong()
```

setLong

Sets the array of long objects associated with the LongArray class.

Syntax: Here is the method signature:

```
void setLong(LongArray LongVal)
```

15.2.20 NrfServiceException

This is the exception thrown by the remote Roles Web Service.

NrfServiceException constructors

The NrfServiceException class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
NrfServiceException()
```

Syntax 2: Here is the syntax for a constructor that takes a String as a parameter:

```
NrfServiceException(java.lang.String reason)
```

getReason

Returns the reason for the exception.

Syntax: Here is the method signature:

```
public java.lang.String getReason()
```

setReason

Sets the reason for the exception.

Syntax: Here is the method signature:

```
public void setReason(java.lang.String reason)
```

15.2.21 RequestCategoryType

An JAX-RPC friendly representation of com.novell.idm.nrf.persist.RequestCategoryType.

Table 15-4 Field Summary

Type	Name
static RequestCategoryType	ROLE_TO_CONTAINER_ADD
static RequestCategoryType	ROLE_TO_CONTAINER_ADD_SUBTREE
static RequestCategoryType	ROLE_TO_CONTAINER_REMOVE
static RequestCategoryType	ROLE_TO_GROUP_ADD
static RequestCategoryType	ROLE_TO_GROUP_REMOVE
static RequestCategoryType	ROLE_TO_ROLE_ADD
static RequestCategoryType	ROLE_TO_ROLE_REMOVE
static RequestCategoryType	ROLE_TO_USER_ADD
static RequestCategoryType	ROLE_TO_USER_REMOVE

RequestCategoryType constructors

The RequestCategoryType class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
RequestCategoryType()
```

Syntax 2: Here is the syntax for a constructor that takes a String as a parameter:

```
RequestCategoryType(java.lang.String value)
```

equals

Implementation of equals(). This implementation overrides the equals() method in java.lang.Object.

Syntax: Here is the method signature:

```
public boolean equals(java.lang.Object obj)
```

fromRPC

Reconstructs an API representation object from an RPC representation.

Syntax: Here is the method signature:

```
public com.novell.idm.nrf.persist.RequestCategoryType fromRPC() throws  
com.novell.idm.nrf.exception.NrfException
```

fromValue

This method is for WSSDK serialization.

Syntax: Here is the method signature:

```
public static RequestCategoryType fromValue(java.lang.String value)
```

getValue

Gets the type.

Syntax: Here is the method signature:

```
public java.lang.String getValue()
```

hashCode

This implementation overrides the hashCode() method in java.lang.Object.

Syntax: Here is the method signature:

```
public int hashCode()
```

setValue

Sets the type.

Syntax: Here is the method signature:

```
public void setValue(java.lang.String type)
```

toRPC

Constructs an RPC friendly representation off of an API object.

Syntax: Here is the method signature:

```
public static RequestCategoryType  
toRPC(com.novell.idm.nrf.persist.RequestCategoryType type)
```

toString

Implementation of toString() that returns a string representation of the class.

Syntax: Here is the method signature:

```
public java.lang.String toString()
```

15.2.22 RequestStatus

An JAX-RPC friendly representation of com.novell.idm.nrf.persist.RequestStatus.

Table 15-5 Field Summary

Type	Name
static RequestStatus	ACTIVATION_TIME_PENDING
static RequestStatus	APPROVAL_PENDING
static RequestStatus	APPROVAL_START_PENDING
static RequestStatus	APPROVAL_START_SUSPENDED
static RequestStatus	APPROVED
static RequestStatus	CLEANUP
static RequestStatus	DENIED
static RequestStatus	NEW_REQUEST
static RequestStatus	PROVISION
static RequestStatus	PROVISIONED
static RequestStatus	PROVISIONING_ERROR
static RequestStatus	SOD_APPROVAL_START_PENDING
static RequestStatus	SOD_APPROVAL_START_SUSPENDED
static RequestStatus	SOD_EXCEPTION_APPROVAL_PENDING
static RequestStatus	SOD_EXCEPTION_APPROVED
static RequestStatus	SOD_EXCEPTION_DENIED

RequestStatus constructors

The RequestStatus class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
RequestStatus()
```

Syntax 2: Here is the syntax for a constructor that takes a String as a parameter:

```
RequestStatus(java.lang.String value)
```

equals

Implementation of equals().

Syntax: Here is the method signature:

```
public boolean equals(java.lang.Object obj)
```

fromRPC

Reconstructs an API representation object from an RPC representation.

Syntax: Here is the method signature:

```
public com.novell.idm.nrf.persist.RequestStatus fromRPC() throws  
com.novell.idm.nrf.exception.NrfException
```

fromValue

This method is for WSSDK serialization.

Syntax: Here is the method signature:

```
public static RequestStatus fromValue(java.lang.String value)
```

getValue

Gets the type.

Syntax: Here is the method signature:

```
public java.lang.String getValue()
```

hashCode

This implementation overrides the hashCode() method in java.lang.Object.

Syntax: Here is the method signature:

```
public int hashCode()
```

setValue

Sets the type.

Syntax: Here is the method signature:

```
public void setValue(java.lang.String type)
```

toRPC

Constructs an RPC friendly representation off of an API object.

Syntax: Here is the method signature:

```
public static RequestStatus toRPC(com.novell.idm.nrf.persist.RequestStatus type)
```

toString

Implementation of toString() that returns a string representation of the class.

Syntax: Here is the method signature:

```
public java.lang.String toString()
```

15.2.23 ResourceAssociation

Supporting class that holds information about resource associations for a role.

getRole

Returns the DN for the role involved in the association.

```
public String getRole()
```

setRole

Sets the DN for the role involved in the association.

```
public void setRole(String role)
```

getEntityKey

Returns the entity key for the association.

```
public String getEntityKey()
```

setEntityKey

Sets the entity key for the association.

```
public void setEntityKey(String entityKey)
```

getResource

Returns the DN for the resource involved in the association.

```
public String getResource()
```

setResource

Sets the DN for the resource involved in the association.

```
public void setResource(String resource)
```

getDynamicParameters

Returns the list of dynamic parameters for the resource.

```
public DynamicParameter[] getDynamicParameters()
```

setDynamicParameters

Sets the list of dynamic parameters for the resource.

```
public void setDynamicParameters(DynamicParameter[] parameterValues)
```

getLocalizedDescriptions

Returns the list of localized descriptions.

```
public LocalizedValue[] getLocalizedDescriptions()
```

setLocalizedDescriptions

Sets the list of localized descriptions.

```
public void setLocalizedDescriptions(LocalizedValue[] descriptions)
```

getApprovalOverride

Returns the boolean flag indicating whether the role approval process overrides the resource approval process.

```
public boolean getApprovalOverride()
```

setApprovalOverride

Sets the boolean flag indicating whether the role approval process overrides the resource approval process.

```
public void setApprovalOverride(boolean override)
```

getStatus

Returns the status of the association.

```
public int getStatus()
```

setStatus

Sets the status of the association.

```
public void setStatus(int status)
```

toString

Converts the resource association to a string.

```
public String toString()
```

15.2.24 Role

Value class to hold the role information.

Role constructors

The Role class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
Role()
```

getApprovers

Gets the approvers of the role approval.

Syntax: Here is the method signature:

```
public ApproverArray getApprovers()
```

getAssociatedRoles

Gets the associated roles.

Syntax: Here is the method signature:

```
public DNStringArray getAssociatedRoles()
```

getChildRoles

Gets the children roles.

Syntax: Here is the method signature:

```
public DNStringArray getChildRoles()
```

getDescription

Gets the role description.

Syntax: Here is the method signature:

```
public java.lang.String getDescription()
```

getEntitlementRef

Gets the entitlement references.

Syntax: Here is the method signature:

```
public EntitlementArray getEntitlementRef()
```

getEntityKey

Gets the role entity key.

Syntax: Here is the method signature:

```
public java.lang.String getEntityKey()
```

getImplicitContainers

Gets the implicit container DNs.

Syntax: Here is the method signature:

```
public DNStringArray getImplicitContainers()
```

getImplicitGroups

Gets implicit group DNs.

Syntax: Here is the method signature:

```
public DNStringArray getImplicitGroups()
```

getName

Gets the role name.

Syntax: Here is the method signature:

```
public java.lang.String getName()
```

getOwners

Gets the owner DNs.

Syntax: Here is the method signature:

```
public DNStringArray getOwners()
```

getParentRoles

Gets the parent roles.

Syntax: Here is the method signature:

```
public DNStringArray getParentRoles()
```

getQuorum

Gets the quorum amount.

Syntax: Here is the method signature:

```
public java.lang.String getQuorum()
```

getRequestDef

Gets the request definition for approval processing.

Syntax: Here is the method signature:

```
public java.lang.String getRequestDef()
```

getRoleAssignments

Gets the role assignments.

Syntax: Here is the method signature:

```
public RoleAssignmentArray getRoleAssignments()
```

getRoleCategoryKeys

Gets the role category keys.

Syntax: Here is the method signature:

```
public CategoryKeyArray getRoleCategoryKeys()
```

getRoleLevel

Gets the role level object.

Syntax: Here is the method signature:

```
public RoleLevel getRoleLevel()
```

getSystemRole

Gets the system role flag.

Syntax: Here is the method signature:

```
public boolean getSystemRole()
```

setApprovers

Sets the approvers for role approval processing.

Syntax: Here is the method signature:

```
public void setApprovers(ApproverArray approvers)
```

setAssociatedRoles

Sets the associated roles.

Syntax: Here is the method signature:

```
public void setAssociatedRoles(DNStringArray associatedRoles)
```


setChildRoles

Sets the children roles.

Syntax: Here is the method signature:

```
public void setChildRoles(DNStringArray childRoles)
```

setDescription

Sets the role description.

Syntax: Here is the method signature:

```
public void setDescription(java.lang.String description)
```

setEntitlementRef

Sets the entitlement references.

Syntax: Here is the method signature:

```
public void setEntitlementRef(EntitlementArray entitlementRef)
```

setEntityKey

Sets the role entity key.

Syntax: Here is the method signature:

```
public void setEntityKey(java.lang.String entityKey)
```

setImplicitContainers

Sets the implicit container DNs.

Syntax: Here is the method signature:

```
public void setImplicitContainers(DNStringArray implicitContainers)
```

setImplicitGroups

Sets the implicit group DNs.

Syntax: Here is the method signature:

```
public void setImplicitGroups(DNStringArray implicitGroups)
```

setName

Sets the role name.

Syntax: Here is the method signature:

```
public void setName(java.lang.String name)
```

setOwners

Sets the owner DNs.

Syntax: Here is the method signature:

```
public void setOwners(DNStringArray owners)
```

setParentRoles

Sets the parent roles.

Syntax: Here is the method signature:

```
public void setParentRoles(DNStringArray parentRoles)
```

setQuorum

Sets the quorum amount.

Syntax: Here is the method signature:

```
public void setQuorum(java.lang.String quorum)
```

setRequestDef

Sets the request definition for approval processing.

Syntax: Here is the method signature:

```
public void setRequestDef(java.lang.String requestDef)
```

setRoleAssignments

Sets the role assignments.

Syntax: Here is the method signature:

```
public void setRoleAssignments(RoleAssignmentArray roleAssignments)
```

setRoleCategoryKeys

Sets the role category keys.

Syntax: Here is the method signature:

```
public void setRoleCategoryKeys(CategoryKeyArray roleCategoryKeys)
```

setRoleLevel

Sets the role level object.

Syntax: Here is the method signature:

```
public void setRoleLevel(RoleLevel roleLevel)
```

setSystemRole

Sets the system role flag.

Syntax: Here is the method signature:

```
public void setSystemRole(boolean systemRole)
```

15.2.25 RoleAssignment

Value class to hold role assignment information.

RoleAssignment

The RoleAssignment class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
RoleAssignment()
```

getAssignmentType

Gets the role assignment type.

Syntax: Here is the method signature:

```
public RoleAssignmentType getAssignmentType()
```

getCauseIdentities

Gets the cause identities DNs.

Syntax: Here is the method signature:

```
public IdentityTypeDnMapArray getCauseIdentities()
```

getEffectiveDate

Gets the effective date.

Syntax: Here is the method signature:

```
public java.util.Date getEffectiveDate()
```

getExpirationDate

Gets the expiration date.

Syntax: Here is the method signature:

```
public java.util.Date getExpirationDate()
```

getExplicitIdentities

Gets the explicit identities DNs.

Syntax: Here is the method signature:

```
public DNStringArray getExplicitIdentities()
```

getRole

Gets the role associated with the assignment.

Syntax: Here is the method signature:

```
public java.lang.String getRole()
```

setAssignmentType

Sets the role assignment type.

Syntax: Here is the method signature:

```
public void setAssignmentType(RoleAssignmentType assignmentType)
```

setCauseIdentities

Sets the cause identities DNs.

Syntax: Here is the method signature:

```
public void setCauseIdentities(IdentityTypeDnMapArray causeIdentities)
```

setEffectiveDate

Sets the effective date.

Syntax: Here is the method signature:

```
public void setEffectiveDate(java.util.Date effectiveDate)
```

setExpirationDate

Sets the expiration date.

Syntax: Here is the method signature:

```
public void setExpirationDate(java.util.Date expirationDate)
```

setExplicitIdentities

Sets the explicit identities DNs.

Syntax: Here is the method signature:

```
public void setExplicitIdentities(DNStringArray explicitIdentities)
```

setRole

Sets role associated with this assignment.

Syntax: Here is the method signature:

```
public void setRole(java.lang.String role)
```

15.2.26 RoleAssignmentArray

This section provides reference information on the RoleAssignmentArray class.

RoleAssignmentArray constructors

The RoleAssignmentArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
RoleAssignmentArray()
```

Syntax 2: Here is the syntax for a constructor that takes an array of Attribute objects as a parameter:

```
RoleAssignmentArray(RoleAssignment[] RoleAssignmentVal)
```

getRoleassignment

Returns an array of RoleAssignment objects.

Syntax: Here is the method signature:

```
RoleAssignment[] getRoleassignment()
```

setRoleassignment

Sets the array of RoleAssignment objects associated with the RoleAssignmentArray class.

Syntax: Here is the method signature:

```
void setRoleassignment (RoleAssignment[] RoleAssignmentVal)
```

15.2.27 RoleAssignmentActionType

An JAX-RPC friendly representation of com.novell.idm.nrf.RoleAssignmentActionType.

Table 15-6 Field Summary

Type	Name
static RoleAssignmentActionType	EXTEND
static RoleAssignmentActionType	GRANT
static RoleAssignmentActionType	REVOKE

RoleAssignmentActionType constructors

The RoleAssignmentActionType class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
RoleAssignmentActionType()
```

Syntax 2: Here is the syntax for a constructor that takes a String as a parameter:

```
RoleAssignmentActionType(java.lang.String value)
```

equals

Implementation of equals().

Syntax: Here is the method signature:

```
public boolean equals(java.lang.Object obj)
```

fromRPC

Reconstructs an API representation object from an RPC representation.

Syntax: Here is the method signature:

```
public com.novell.idm.nrf.RoleAssignmentActionType fromRPC()
```

fromValue

This method is for WSSDK serialization.

Syntax: Here is the method signature:

```
public static RoleAssignmentActionType fromValue(java.lang.String value)
```

getValue

Gets the type.

Syntax: Here is the method signature:

```
public java.lang.String getValue()
```

hashCode

This is an implementation of hashCode(). This implementation overrides the hashCode() method in java.lang.Object.

Syntax: Here is the method signature:

```
public int hashCode()
```

setValue

Sets the type.

Syntax: Here is the method signature:

```
public void setValue(java.lang.String type)
```

toRPC

Constructs an RPC friendly representation off of an API object.

Syntax: Here is the method signature:

```
public static RoleAssignmentActionType  
toRPC(com.novell.idm.nrf.RoleAssignmentActionType type)
```

toString

Implementation of toString() that returns a string representation of the class.

Syntax: Here is the method signature:

```
public java.lang.String toString()
```

15.2.28 RoleAssignmentRequest

Class to represent a role assignment request.

RoleAssignmentRequest

The RoleAssignmentRequest class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
RoleAssignmentRequest()
```

getActionType

Gets role assignment type (grant, revoke, extend).

Syntax: Here is the method signature:

```
public RoleAssignmentActionType getActionType()
```

getAssignmentType

Gets the role assignment type.

Syntax: Here is the method signature:

```
public RoleAssignmentType getAssignmentType()
```

getCorrelationID

Gets the correlation ID.

Syntax: Here is the method signature:

```
public java.lang.String getCorrelationID()
```

getEffectiveDate

Gets the effective date.

Syntax: Here is the method signature:

```
public java.util.Date getEffectiveDate()
```

getExpirationDate

Gets the expiration date.

Syntax: Here is the method signature:

```
public java.util.Date getExpirationDate()
```

getIdentity

Gets the identity to assign roles to.

Syntax: Here is the method signature:

```
public java.lang.String getIdentity()
```

getReason

Gets the reason for the role assignment.

Syntax: Here is the method signature:

```
public java.lang.String getReason()
```

getRoles

Gets the roles to assign to the identity.

Syntax: Here is the method signature:

```
public DNStringArray getRoles()
```

getSodOverridesRequested

Gets the SOD DNs and justification to override.

Syntax: Here is the method signature:

```
public SodJustificationArray getSodOverridesRequested()
```

setActionType

Sets the action type (grant, revoke, extend).

Syntax: Here is the method signature:

```
public void setActionType(RoleAssignmentActionType actionType)
```

setAssignmentType

Sets the role assignment type.

Syntax: Here is the method signature:

```
public void setAssignmentType(RoleAssignmentType assignmentType)
```


setCorrelationID

Sets the correlation ID.

Syntax: Here is the method signature:

```
public void setCorrelationID(java.lang.String correlationID)
```

setEffectiveDate

Sets the effective date.

Syntax: Here is the method signature:

```
public void setEffectiveDate(java.util.Date effectiveDate)
```

setExpirationDate

Sets the expiration date.

Syntax: Here is the method signature:

```
public void setExpirationDate(java.util.Date expirationDate)
```

setIdentity

Sets the identity to assign roles to.

Syntax: Here is the method signature:

```
public void setIdentity(java.lang.String identity)
```

setReason

Sets the reason for the role assignment.

Syntax: Here is the method signature:

```
public void setReason(java.lang.String reason)
```

setRoles

Sets the roles to assign to the identity.

Syntax: Here is the method signature:

```
public void setRoles(DNStringArray roles)
```

setSodOverridesRequested

Sets the SOD DNs and justification to override.

Syntax: Here is the method signature:

```
public void setSodOverridesRequested(SodJustificationArray sodOverridesRequested)
```

15.2.29 RoleAssignmentRequestStatus

This class represents the status of a role assignment.

RoleAssignmentRequestStatus

The RoleAssignmentRequestStatus class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
RoleAssignmentRequestStatus()
```

getCategory

Gets the request category.

Syntax: Here is the method signature:

```
public RequestCategoryType getCategory()
```

getCorrelationId

Gets the correlation ID.

Syntax: Here is the method signature:

```
public java.lang.String getCorrelationId()
```

getEffectiveDate

Gets the effective date.

Syntax: Here is the method signature:

```
public java.util.Date getEffectiveDate()
```

getEntityKey

Gets the entity key.

Syntax: Here is the method signature:

```
public java.lang.String getEntityKey()
```

getExpirationDate

Gets the expiration date.

Syntax: Here is the method signature:

```
public java.util.Date getExpirationDate()
```

getReason

Gets the reason for the role assignment.

Syntax: Here is the method signature:

```
public java.lang.String getReason()
```

getRequestDate

Gets the request date.

Syntax: Here is the method signature:

```
public java.util.Date getRequestDate()
```

getRequester

Gets the request DN.

Syntax: Here is the method signature:

```
public java.lang.String requester()
```

getSource

Gets the source Role DN.

Syntax: Here is the method signature:

```
public java.lang.String getSource()
```

getStatus

Gets the request status.

Syntax: Here is the method signature:

```
public RequestStatus getStatus()
```

getTarget

Gets the targeted identity DN.

Syntax: Here is the method signature:

```
public java.lang.String getTarget()
```

setCategory

Sets the request category.

Syntax: Here is the method signature:

```
public void setCategory(RequestCategoryType category)
```

setCorrelationId

Sets the correlation ID.

Syntax: Here is the method signature:

```
public void setCorrelationId(java.lang.String correlationId)
```

setEffectiveDate

Sets the effective date.

Syntax: Here is the method signature:

```
public void setEffectiveDate(java.util.Date effectiveDate)
```

setEntityKey

Sets the entity key.

Syntax: Here is the method signature:

```
public void setEntityKey(java.lang.String entityKey)
```

setExpirationDate

Sets the expiration date.

Syntax: Here is the method signature:

```
public void setExpirationDate(java.util.Date expirationDate)
```

setReason

Sets the reason for the role assignment.

Syntax: Here is the method signature:

```
public void setReason(java.lang.String reason)
```

setRequestDate

Sets the request date.

Syntax: Here is the method signature:

```
public void setRequestDate(java.util.Date requestDate)
```

setRequester

Sets the requester DN.

Syntax: Here is the method signature:

```
public void setRequester(java.lang.String requester)
```

setSource

Sets the source Role DN.

Syntax: Here is the method signature:

```
public void setSource(java.lang.String source)
```

setStatus

Sets the request status.

Syntax: Here is the method signature:

```
public void setStatus(RequestStatus status)
```

setTarget

Sets the identity targeted DN.

Syntax: Here is the method signature:

```
public void setTarget(java.lang.String target)
```

15.2.30 RoleAssignmentType

An JAX-RPC friendly representation of com.novell.idm.nrf.RoleAssignmentType.

Table 15-7 Field Summary

Type	Name
static RoleAssignmentType	CONTAINER_TO_ROLE
static RoleAssignmentType	CONTAINER_WITH_SUBTREE_TO_ROLE
static RoleAssignmentType	GROUP_TO_ROLE
static RoleAssignmentType	ROLE_TO_ROLE
static RoleAssignmentType	USER_TO_ROLE

RoleAssignmentType constructors

The CategoryKey class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
CategoryKey()
```

Syntax 2: Here is the syntax for a constructor that takes a String as a parameter:

```
CategoryKey(java.lang.String categoryKey)
```

convertToAPI

Reconstructs an API representation object from an RPC representation.

Syntax: Here is the method signature:

```
public com.novell.idm.nrf.RoleAssignmentType convertToAPI()
```

convertToRPC

Constructs an RPC friendly representation off of an API object.

Syntax: Here is the method signature:

```
public static RoleAssignmentType  
convertToRPC(com.novell.idm.nrf.RoleAssignmentType type)
```

equals

Implementation of equals().

Syntax: Here is the method signature:

```
public boolean equals(java.lang.Object obj)
```

fromValue

This method is for WSSDK serialization.

Syntax: Here is the method signature:

```
public static RoleAssignmentType fromValue(java.lang.String value)
```

getValue

Gets the type.

Syntax: Here is the method signature:

```
public java.lang.String getValue()
```

hashCode

This is an implementation of hashCode(). This implementation overrides the hashCode() method in java.lang.Object.

Syntax: Here is the method signature:

```
public int hashCode()
```

setValue

Sets the type.

Syntax: Here is the method signature:

```
public void setValue(java.lang.String type)
```

toString

Implementation of toString() that returns a string representation of the class.

Syntax: Here is the method signature:

```
public java.lang.String toString()
```

15.2.31 RoleAssignmentTypeInfo

An JAX-RPC friendly representation of the details of the com.novell.idm.nrf.RoleAssignmentType enumeration.

RoleAssignmentTypeInfo

The RoleAssignmentTypeInfo class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
RoleAssignmentTypeInfo()
```

convertToRPC

Constructs an RPC friendly representation from an API object.

Syntax: Here is the method signature:

```
public static RoleAssignmentTypeInfo  
convertToRPC(com.novell.idm.nrf.RoleAssignmentType type)
```

getIdentityType

Returns the JAX-RPC friendly identity type.

Syntax: Here is the method signature:

```
public IdentityType getIdentityType()
```

getSubtreeIncluded

Determines whether the sub tree is included.

Syntax: Here is the method signature:

```
public boolean getSubtreeIncluded()
```

getSupportsApproval

Determines whether the assignment supports approval.

Syntax: Here is the method signature:

```
public boolean getSupportsApproval()
```

getSupportsEffectiveDate

Determines whether the assignment supports an effective date.

Syntax: Here is the method signature:

```
public boolean getSupportsEffectiveDate()
```

getSupportsExpiration

Determines whether the assignment supports expiration.

Syntax: Here is the method signature:

```
public boolean getSupportsExpiration()
```

getSupportsSODApproval

Determines whether the assignment supports SOD approval.

Syntax: Here is the method signature:

```
public boolean getSupportsSODApproval()
```

setIdentityType

Sets the JAX-RPC friendly identity type.

Syntax: Here is the method signature:

```
public void setIdentityType(IdentityType type)
```

setSubtreeIncluded

Sets whether the sub tree is included.

Syntax: Here is the method signature:

```
public void setSubtreeIncluded(boolean bool)
```

setSupportsApproval

Sets whether the assignment supports approval.

Syntax: Here is the method signature:

```
public void setSupportsApproval(boolean bool)
```

setSupportsEffectiveDate

Sets whether the assignment supports effective date.

Syntax: Here is the method signature:

```
public void setSupportsEffectiveDate(boolean bool)
```

setSupportsExpiration

Sets whether the assignment supports expiration.

Syntax: Here is the method signature:

```
public void setSupportsExpiration(boolean bool)
```


setSupportsSODApproval

Sets whether the assignment supports SOD approval.

Syntax: Here is the method signature:

```
public void setSupportsSODApproval(boolean bool)
```

15.2.32 RoleInfo

Value class to hold main role information. This is a small subset of the role value class.

RoleInfo constructors

The RoleInfo class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
RoleInfo()
```

getDescription

Gets the role description.

Syntax: Here is the method signature:

```
public java.lang.String getDescription()
```

getEntityKey

Gets the role entity key.

Syntax: Here is the method signature:

```
public java.lang.String getEntityKey()
```

getName

Gets the role name.

Syntax: Here is the method signature:

```
public java.lang.String getName()
```

getRoleCategoryKeys

Gets the role category keys.

Syntax: Here is the method signature:

```
public CategoryKeyArray getRoleCategoryKeys()
```

getRoleLevel

Gets the role level object.

Syntax: Here is the method signature:

```
public RoleLevel getRoleLevel()
```

setDescription

Sets the role description.

Syntax: Here is the method signature:

```
public void setDescription(java.lang.String description)
```

setEntityKey

Sets the role entity key.

Syntax: Here is the method signature:

```
public void setEntityKey(java.lang.String entityKey)
```

setName

Sets the role name.

Syntax: Here is the method signature:

```
public void setName(java.lang.String name)
```

setRoleCategoryKeys

Sets the role category keys.

Syntax: Here is the method signature:

```
public void setRoleCategoryKeys(CategoryKeyArray roleCategoryKeys)
```

setRoleLevel

Sets role level object.

Syntax: Here is the method signature:

```
public void setRoleLevel(RoleLevel roleLevel)
```

15.2.33 RoleInfoArray

This section provides reference information on the RoleInfoArray class.

RoleInfoArray constructors

The RoleInfoArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
RoleInfoArray()
```

Syntax 2: Here is the syntax for a constructor that takes an array of Attribute objects as a parameter:

```
RoleInfoArray(RoleInfo[] RoleInfoVal)
```

getRoleinfo

Returns an array of RoleInfo objects.

Syntax: Here is the method signature:

```
RoleInfo[] getRoleinfo()
```

setRoleinfo

Sets the array of RoleInfo objects associated with the RoleInfoArray class.

Syntax: Here is the method signature:

```
void setRoleinfo (RoleInfo[] RoleInfoVal)
```

15.2.34 RoleLevel

This class represent a role level.

RoleLevel constructors

The RoleLevel class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
RoleLevel()
```

getContainer

Gets the role level container.

Syntax: Here is the method signature:

```
public java.lang.String getContainer()
```

getDescription

Gets the role level description.

Syntax: Here is the method signature:

```
public java.lang.String getDescription()
```

getLevel

Gets the role level.

Syntax: Here is the method signature:

```
public long getLevel()
```

getName

Gets the role level name.

Syntax: Here is the method signature:

```
public java.lang.String getName()
```

setContainer

Sets the role level container.

Syntax: Here is the method signature:

```
public void setContainer(java.lang.String container)
```

setDescription

Sets the role level description.

Syntax: Here is the method signature:

```
public void setDescription(java.lang.String description)
```

setLevel

Sets the role level.

Syntax: Here is the method signature:

```
public void setLevel(long level)
```

setName

Sets the role level name.

Syntax: Here is the method signature:

```
public void setName(java.lang.String name)
```

15.2.35 RoleLevelArray

This section provides reference information on the RoleLevelArray class.

RoleLevelArray constructors

The RoleLevelArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
RoleLevelArray()
```

Syntax 2: Here is the syntax for a constructor that takes an array of Attribute objects as a parameter:

```
RoleLevelArray(RoleLevel[] RoleLevelVal)
```

getRolelevel

Returns an array of RoleLevel objects.

Syntax: Here is the method signature:

```
RoleLevel[] getRolelevel()
```

setRolelevel

Sets the array of RoleLevel objects associated with the RoleLevelArray class.

Syntax: Here is the method signature:

```
void setRolelevel (RoleLevel[] RoleLevelVal)
```

15.2.36 RoleRequest

The Role Request class has been added to support the creation of roles. The Role Request class is a value class used to hold information about a request to create a role.

getName

Gets the role name.

Syntax: Here is the method signature:

```
public String getName()
```

getDescription

Gets the role description.

Syntax: Here is the method signature:

```
public String getDescription()
```

getEntityKey

Gets the entity key for the role.

Syntax: Here is the method signature:

```
public String getEntityKey()
```

getRoleLevel

Gets the role level object.

Syntax: Here is the method signature:

```
public long getRoleLevel()
```

getRoleCategoryKeys

Gets the role category keys.

Syntax: Here is the method signature:

```
public CategoryKey[] getRoleCategoryKeys()
```

getQuorum

Gets the quorum amount.

Syntax: Here is the method signature:

```
public String getQuorum()
```

getRequestDef

Gets the provisioning request definition for approval processing.

Syntax: Here is the method signature:

```
public String getRequestDef()
```

getApprovers

Gets the approvers for the role definition.

Syntax: Here is the method signature:

```
public Approver[] getApprovers()
```

getOwners

Gets the owner DNs.

Syntax: Here is the method signature:

```
public DNString[] getOwners()
```

getRoleAssignments

Gets the associated roles.

Syntax: Here is the method signature:

```
public String getRoleAssignments()
```

getSystemRole

Gets the system role flag, which indicates whether this is a system role.

Syntax: Here is the method signature:

```
public boolean getSystemRole()
```

getContainer

Gets the name of the role container.

Syntax: Here is the method signature:

```
public String getContainer()
```

setName

Sets the role name.

Syntax: Here is the method signature:

```
public void setName()
```

setDescription

Sets the role description.

Syntax: Here is the method signature:

```
public void setDescription()
```

setEntityKey

Sets the entity key for the role.

Syntax: Here is the method signature:

```
public void setEntityKey()
```

setRoleLevel

Sets the role level object.

Syntax: Here is the method signature:

```
public void setRoleLevel()
```

setRoleCategoryKeys

Sets the role category keys.

Syntax: Here is the method signature:

```
public void setRoleCategoryKeys()
```

setQuorum

Sets the quorum amount.

Syntax: Here is the method signature:

```
public void setQuorum()
```

setRequestDef

Sets the provisioning request definition for approval processing.

Syntax: Here is the method signature:

```
public void setRequestDef()
```

setApprovers

Sets the approvers for role approval processing.

Syntax: Here is the method signature:

```
public void setApprovers()
```

setOwners

Sets the owner DNs.

Syntax: Here is the method signature:

```
public void setOwners()
```

setSystemRole

Sets the system role flag, which determines whether this is a system role.

Syntax: Here is the method signature:

```
public void setSystemRole()
```

setContainer

Sets the role container.

Syntax: Here is the method signature:

```
public void setContainer()
```

15.2.37 RoleServiceDelegate

Delegate class to perform the actual call to the API layer. Should be used by all skeleton classes.

RoleServiceDelegate constructors

The RoleServiceDelegate class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
RoleServiceDelegate(com.novell.srvprv.spi.security.ISecurityContext ctx,  
java.util.Locale locale)
```

findSodByExample

Finds all SoD objects based on the search criteria in the given SOD object.

Syntax: Here is the method signature:

```
SodArray findSodByExample(Sod sod) throws NrfServiceException,  
java.rmi.RemoteException
```


findSodByExampleWithOperator

Finds all SoD objects based on the search criteria found in the given SOD object

Syntax: Here is the method signature:

```
SodArray findSodByExampleWithOperator(Sod searchCriteria, boolean useAndForMultiValueSearch) throws NrfServiceException, java.rmi.RemoteException
```

findSodById

Find by key.

Syntax: Here is the method signature:

```
Sod findSodById(java.lang.String entityKey) throws NrfServiceException, java.rmi.RemoteException
```

getAssignedIdentities

Returns a list of role assignments for a specified identity.

Syntax: Here is the method signature:

```
RoleAssignmentArray getAssignedIdentities(java.lang.String identityDn, IdentityType type, boolean direct) throws NrfServiceException, java.rmi.RemoteException
```

getConfiguration

Returns the role system configuration defined in the role vault root (nrfConfiguration)

Syntax: Here is the method signature:

```
Configuration getConfiguration() throws NrfServiceException, java.rmi.RemoteException
```

getContainer

Gets container and role information for a given container DN.

Syntax: Here is the method signature:

```
Container getContainer(java.lang.String containerDn) throws NrfServiceException, java.rmi.RemoteException
```

getExceptionList

Returns a list of Sod instances for all SOD violations found for a specific identity and type.

Syntax: Here is the method signature:

```
SodArray getExceptionsList(java.lang.String identity, IdentityType identityType) throws NrfServiceException, java.rmi.RemoteException
```

getGroup

Gets group and role information for a given group DN.

Syntax: Here is the method signature:

```
Group getGroup(java.lang.String groupDn) throws NrfServiceException,  
java.rmi.RemoteException
```

getIdentitiesInViolation

Returns a map of identities which are in violation of a given SoD.

Syntax: Here is the method signature:

```
IdentityTypeDnMapArray getIdentitiesInViolation(java.lang.String sodDn) throws  
NrfServiceException, java.rmi.RemoteException
```

getIdentityRoleConflicts

Returns a list of Sod instances for all SOD conflicts found for a given list of roles for a given identity.

Syntax: Here is the method signature:

```
SodArray getIdentityRoleConflicts(java.lang.String identity, IdentityType  
identityType, DNStringArray requestedRoles) throws NrfServiceException,  
java.rmi.RemoteException
```

getRole

Retrieves a role object defined by a role DN

Syntax: Here is the method signature:

```
Role getRole(java.lang.String roleDn) throws NrfServiceException,  
java.rmi.RemoteException
```

getRoleAssignmentRequestStatus

Returns a list of role assignment request status instances given a correlation ID.

Syntax: Here is the method signature:

```
RoleAssignmentRequestStatusArray getRoleAssignmentRequestStatus(java.lang.String  
correlationId) throws NrfServiceException, java.rmi.RemoteException
```

getRoleAssignmentRequestStatusByIdentityType

Returns a list of role assignment request status instances given an identity and an identity type.

Syntax: Here is the method signature:

```
RoleAssignmentRequestStatusArray  
getRoleAssignmentRequestStatusByIdentityType(java.lang.String identityDn,  
IdentityType identityType) throws NrfServiceException, java.rmi.RemoteException
```

getRoleAssignmentTypeInfo

Retrieves details about a RoleAssignmentType.

Syntax: Here is the method signature:

```
RoleAssignmentTypeInfo getRoleAssignmentTypeInfo(RoleAssignmentType type) throws  
NrfServiceException, java.rmi.RemoteException
```

getRoleCategories

Gets role categories.

Syntax: Here is the method signature:

```
CategoryArray getRoleCategories() throws NrfServiceException,  
java.rmi.RemoteException
```

getRoleConflicts

Returns a list of Sod instances found for all given roles. This method always returns a list.

Syntax: Here is the method signature:

```
SodArray getRoleConflicts(DNStringArray roles) throws NrfServiceException,  
java.rmi.RemoteException
```

getRoleLevels

Gets role levels.

Syntax: Here is the method signature:

```
RoleLevelArray getRoleLevels() throws NrfServiceException,  
java.rmi.RemoteException
```

getRolesInfo

Returns a list of RoleInfo instances given a list of role DNs.

Syntax: Here is the method signature:

```
RoleInfoArray getRolesInfo(DNStringArray roleDns) throws NrfServiceException,  
java.rmi.RemoteException
```

getRolesInfoByCategory

Returns a list of RoleInfo instances given a list of role category keys.

Syntax: Here is the method signature:

```
RoleInfoArray getRolesInfoByCategory(CategoryKeyArray roleCategoryKeys) throws  
NrfServiceException, java.rmi.RemoteException
```

getRolesInfoByLevel

Returns a list of RoleInfo instances given a list of role levels.

Syntax: Here is the method signature:

```
RoleInfoArray getRolesInfoByLevel(LongArray roleLevels) throws  
NrfServiceException, java.rmi.RemoteException
```

getTargetSourceConflicts

Returns a list of Sod instances for all SOD conflicts defined between the target role DN and the source role DN.

Syntax: Here is the method signature:

```
SodArray getTargetSourceConflicts(java.lang.String targetName, java.lang.String  
sourceName) throws NrfServiceException, java.rmi.RemoteException
```

getUser

Gets user info including all role assignments for a given user DN stored in a UserIdentity object.

Syntax: Here is the method signature:

```
User getUser(java.lang.String userDn) throws NrfServiceException,  
java.rmi.RemoteException
```

getVersion

Returns the version of this Web Service.

Syntax: Here is the method signature:

```
VersionVO getVersion() throws java.rmi.RemoteException
```

isUserInRole

Returns boolean flag; true if role has been assigned to a User identity

Syntax: Here is the method signature:

```
boolean isUserInRole(java.lang.String userDn, java.lang.String roleDn)
```

requestRoleAssignment

Returns a list of request DNs created by the role assignment

Syntax: Here is the method signature:

```
DNStringArray requestRolesAssignment(RoleAssignmentRequest roleAssignmentRequest)  
throws NrfServiceException, java.rmi.RemoteException
```

15.2.38 RoleServiceSkeletonImpl

Class to represent the skeleton server side implementation of the Role Based offered services.

RoleServiceSkeletonImpl

The RoleServiceSkeletonImpl class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
RoleServiceSkeletonImpl()
```

findSodByExample

Finds all SoD objects based on the search criteria in the given SOD object.

Syntax: Here is the method signature:

```
SodArray findSodByExample(Sod sod) throws NrfServiceException,  
java.rmi.RemoteException
```

findSodByExampleWithOperator

Finds all SoD objects based on the search criteria found in the given SOD object

Syntax: Here is the method signature:

```
SodArray findSodByExampleWithOperator(Sod searchCriteria, boolean  
useAndForMultiValueSearch) throws NrfServiceException, java.rmi.RemoteException
```

findSodById

Find by key.

Syntax: Here is the method signature:

```
Sod findSodById(java.lang.String entityKey) throws NrfServiceException,  
java.rmi.RemoteException
```

getAssignedIdentities

Returns a list of role assignments for a specified identity.

Syntax: Here is the method signature:

```
RoleAssignmentArray getAssignedIdentities(java.lang.String identityDn,  
IdentityType type, boolean direct) throws NrfServiceException,  
java.rmi.RemoteException
```

getConfiguration

Returns the role system configuration defined in the role vault root (nrfConfiguration)

Syntax: Here is the method signature:

```
Configuration getConfiguration() throws NrfServiceException,  
java.rmi.RemoteException
```

getContainer

Gets container and role information for a given container DN.

Syntax: Here is the method signature:

Container getContainer(java.lang.String containerDn) throws
NrfServiceException, java.rmi.RemoteException

getExceptionList

Returns a list of Sod instances for all SOD violations found for a specific identity and type.

Syntax: Here is the method signature:

```
SodArray getExceptionsList(java.lang.String identity, IdentityType identityType)  
throws NrfServiceException, java.rmi.RemoteException
```

getGroup

Gets group and role information for a given group DN.

Syntax: Here is the method signature:

```
Group getGroup(java.lang.String groupDn) throws NrfServiceException,  
java.rmi.RemoteException
```

getIdentitiesInViolation

Returns a map of identities which are in violation of a given SoD.

Syntax: Here is the method signature:

```
IdentityTypeDnMapArray getIdentitiesInViolation(java.lang.String sodDn) throws  
NrfServiceException, java.rmi.RemoteException
```

getIdentityRoleConflicts

Returns a list of Sod instances for all SOD conflicts found for a given list of roles for a given identity.

Syntax: Here is the method signature:

```
SodArray getIdentityRoleConflicts(java.lang.String identity, IdentityType  
identityType, DNStringArray requestedRoles) throws NrfServiceException,  
java.rmi.RemoteException
```

getRole

Retrieves a role object defined by a role DN

Syntax: Here is the method signature:

```
Role getRole(java.lang.String roleDn) throws NrfServiceException,  
java.rmi.RemoteException
```

getRoleAssignmentRequestStatus

Returns a list of role assignment request status instances given a correlation ID.

Syntax: Here is the method signature:

```
RoleAssignmentRequestStatusArray getRoleAssignmentRequestStatus(java.lang.String  
correlationId) throws NrfServiceException, java.rmi.RemoteException
```

getRoleAssignmentRequestStatusByIdentityType

Returns a list of role assignment request status instances given an identity and an identity type.

Syntax: Here is the method signature:

```
RoleAssignmentRequestStatusArray  
getRoleAssignmentRequestStatusByIdentityType(java.lang.String identityDn,  
IdentityType identityType) throws NrfServiceException, java.rmi.RemoteException
```

getRoleAssignmentTypeInfo

Retrieves details about a RoleAssignmentType.

Syntax: Here is the method signature:

```
RoleAssignmentTypeInfo getRoleAssignmentTypeInfo(RoleAssignmentType type) throws  
NrfServiceException, java.rmi.RemoteException
```

getRoleCategories

Gets role categories.

Syntax: Here is the method signature:

```
CategoryArray getRoleCategories() throws NrfServiceException,  
java.rmi.RemoteException
```

getRoleConflicts

Returns a list of Sod instances found for all given roles. This method always returns a list.

Syntax: Here is the method signature:

```
SodArray getRoleConflicts(DNStringArray roles) throws NrfServiceException,  
java.rmi.RemoteException
```

getRoleLevels

Gets role levels.

Syntax: Here is the method signature:

```
RoleLevelArray getRoleLevels() throws NrfServiceException,  
java.rmi.RemoteException
```

getRolesInfo

Returns a list of RoleInfo instances given a list of role DNs.

Syntax: Here is the method signature:

```
RoleInfoArray getRolesInfo(DNStringArray roleDns) throws NrfServiceException,  
java.rmi.RemoteException
```

getRolesInfoByCategory

Returns a list of RoleInfo instances given a list of role category keys.

Syntax: Here is the method signature:

```
RoleInfoArray getRolesInfoByCategory(CategoryKeyArray roleCategoryKeys) throws  
NrfServiceException, java.rmi.RemoteException
```

getRolesInfoByLevel

Returns a list of RoleInfo instances given a list of role levels.

Syntax: Here is the method signature:

```
RoleInfoArray getRolesInfoByLevel(LongArray roleLevels) throws  
NrfServiceException, java.rmi.RemoteException
```

getTargetSourceConflicts

Returns a list of Sod instances for all SOD conflicts defined between the target role DN and the source role DN.

Syntax: Here is the method signature:

```
SodArray getTargetSourceConflicts(java.lang.String targetName, java.lang.String  
sourceName) throws NrfServiceException, java.rmi.RemoteException
```

getUser

Gets user info including all role assignments for a given user DN stored in a UserIdentity object.

Syntax: Here is the method signature:

```
User getUser(java.lang.String userDn) throws NrfServiceException,  
java.rmi.RemoteException
```

getVersion

Returns the version of this Web Service.

Syntax: Here is the method signature:

```
VersionVO getVersion() throws java.rmi.RemoteException
```

isUserInRole

Returns boolean flag; true if role has been assigned to a User identity

Syntax: Here is the method signature:

```
boolean isUserInRole(java.lang.String userDn, java.lang.String roleDn)
```

requestRoleAssignment

Returns a list of request DNs created by the role assignment

Syntax: Here is the method signature:

```
DNStringArray requestRolesAssignment(RoleAssignmentRequest roleAssignmentRequest)
throws NrfServiceException, java.rmi.RemoteException
```

15.2.39 Sod

Value object to hold SOD information.

Sod constructors

The Sod class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
Sod()
```

getApprovalType

Gets the SOD approval type.

Syntax: Here is the method signature:

```
public SodApprovalType getApprovalType()
```

getApprovers

Gets SOD approvers.

Syntax: Here is the method signature:

```
public ApproverArray getApprovers()
```

getDescription

Gets the SOD description.

Syntax: Here is the method signature:

```
public java.lang.String getDescription()
```

getEntityKey

Gets the SOD entity key.

Syntax: Here is the method signature:

```
public java.lang.String getEntityKey()
```

getName

Gets the SOD name.

Syntax: Here is the method signature:

```
public java.lang.String getName()
```

getQuorum

Gets the SOD quorum amount.

Syntax: Here is the method signature:

```
public java.lang.String getQuorum()
```

getRequestDef

Gets the request definition for approval processing.

Syntax: Here is the method signature:

```
public java.lang.String getRequestDef()
```

getRoles

Gets the SOD roles.

Syntax: Here is the method signature:

```
public DNStringArray getRoles()
```

setApprovalType

Sets the SOD approval type.

Syntax: Here is the method signature:

```
public void setApprovalType(SodApprovalType approvalType)
```

setApprovers

Sets the SOD approvers.

Syntax: Here is the method signature:

```
public void setApprovers(ApproverArray approvers)
```

setDescription

Sets the SOD description.

Syntax: Here is the method signature:

```
public void setDescription(java.lang.String description)
```

setEntityKey

Sets the SOD entity key.

Syntax: Here is the method signature:

```
public void setEntityKey(java.lang.String entityKey)
```

setName

Sets the SOD name.

Syntax: Here is the method signature:

```
public void setName(java.lang.String name)
```

setQuorum

Sets the SOD quorum amount.

Syntax: Here is the method signature:

```
public void setQuorum(java.lang.String quorum)
```

setRequestDef

Sets the request definition for approval processing.

Syntax: Here is the method signature:

```
public void setRequestDef(java.lang.String requestDef)
```

setRoles

Sets the SOD roles.

Syntax: Here is the method signature:

```
public void setRoles(DNStringArray roles)
```

15.2.40 SodArray

This section provides reference information on the SodArray class.

SodArray constructors

The SodArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
SodArray()
```

Syntax 2: Here is the syntax for a constructor that takes an array of Attribute objects as a parameter:

```
SodArray(Sod[] SodVal)
```

getSod

Returns an array of Sod objects.

Syntax: Here is the method signature:

```
Sod[] getSod()
```

setSod

Sets the array of Sod objects associated with the SodArray class.

Syntax: Here is the method signature:

```
void setSod (Sod[] SodVal)
```

15.2.41 SodApprovalType

An JAX-RPC friendly representation of com.novell.idm.nrf.api.SodApprovalType.

Table 15-8 Field Summary

Type	Name
static SodApprovalType	ALLOW_WITH_WORKFLOW
static SodApprovalType	ALWAYS_ALLOW

SodApprovalType constructors

The SodApprovalType class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
SodApprovalType()
```

Syntax 2: Here is the syntax for a constructor that takes a String as a parameter:

```
SodApprovalType(java.lang.String value)
```

equals

Implementation of equals().

Syntax: Here is the method signature:

```
public boolean equals(java.lang.Object obj)
```

fromRPC

Reconstructs an API representation object from an RPC representation.

Syntax: Here is the method signature:

```
public com.novell.idm.nrf.api.SodApprovalType fromRPC() throws  
com.novell.idm.nrf.exception.NrfException
```

fromValue

This method is for WSSDK serialization.

Syntax: Here is the method signature:

```
public static SodApprovalType fromValue(java.lang.String value)
```

getValue

Gets the type.

Syntax: Here is the method signature:

```
public java.lang.String getValue()
```

hashCode

This is an implementation of hashCode(). This implementation overrides the hashCode() method in java.lang.Object.

Syntax: Here is the method signature:

```
public int hashCode()
```

setValue

Sets the type.

Syntax: Here is the method signature:

```
public void setValue(java.lang.String type)
```

toRPC

Reconstructs an API representation object from an RPC representation.

Syntax: Here is the method signature:

```
public com.novell.idm.nrf.api.SodApprovalType fromRPC() throws  
com.novell.idm.nrf.exception.NrfException
```

toString

Implementation of toString() that returns a string representation of the class.

Syntax: Here is the method signature:

```
public java.lang.String toString()
```

15.2.42 SodJustification

Class to represent an SOD DN to override with a justification. Used for assignment of roles to be able to pass in a justification for overrides of SODs.

SodJustification constructors

The SodJustification class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
SodJustification()
```

Syntax 2: Here is the syntax for a constructor that takes two String values as parameters:

```
SodJustification(java.lang.String sodDN, java.lang.String justification)
```

getJustification

Gets the SOD justification for override.

Syntax: Here is the method signature:

```
public java.lang.String getJustification()
```

getSodDN

Gets the SOD DN for override.

Syntax: Here is the method signature:

```
public java.lang.String getSodDN()
```

setJustification

Sets the justification for override.

Syntax: Here is the method signature:

```
public void setJustification(java.lang.String justification)
```

setSodDN

Sets the SOD DN for override.

Syntax: Here is the method signature:

```
public void setSodDN(java.lang.String sodDN)
```

15.2.43 SodJustificationArray

This section provides reference information on the SodJustificationArray class.

SodJustificationArray constructors

The SodJustificationArray class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
SodJustificationArray()
```

Syntax 2: Here is the syntax for a constructor that takes an array of Attribute objects as a parameter:

```
SodJustificationArray(SodJustification[] SodJustificationVal)
```

getSodjustification

Returns an array of SodJustification objects.

Syntax: Here is the method signature:

```
SodJustification[] getSodjustification()
```

setSodjustification

Sets the array of SodJustification objects associated with the SodJustificationArray class.

Syntax: Here is the method signature:

```
void setSodjustification (SodJustification[] SodJustificationVal)
```

15.2.44 User

Value class to hold user identity information.

User constructors

The User class supports a single constructor.

Syntax: Here is the syntax for the constructor:

```
User()
```

getAssociatedRoles

Gets the associated roles for this identity.

Syntax: Here is the method signature:

```
public DNStringArray getAssociatedRoles()
```

getCn

Gets the cn.

Syntax: Here is the method signature:

```
public java.lang.String getCn()
```

getContainerRoles

Gets the container roles.

Syntax: Here is the method signature:

```
public DNStringArray getContainerRoles()
```

getEmail

Gets the email address.

Syntax: Here is the method signature:

```
public java.lang.String getEmail()
```

getEntityKey

Gets the identity entity key.

Syntax: Here is the method signature:

```
public java.lang.String getEntityKey()
```

getExplicitAssignments

Gets the explicit role assignments.

Syntax: Here is the method signature:

```
public RoleAssignmentArray getExplicitAssignments()
```

getFirstName

Gets the first name.

Syntax: Here is the method signature:

```
public java.lang.String getFirstName()
```

getGroupRoles

Gets the group roles.

Syntax: Here is the method signature:

```
public DNStringArray getGroupRoles()
```

getIdentityType

Gets identity type.

Syntax: Here is the method signature:

```
public IdentityType getIdentityType()
```

getImplicitAssignments

Gets the implicit role assignments.

Syntax: Here is the method signature:

```
public RoleAssignmentArray getImplicitAssignments()
```

getInheritedAssignments

Gets the inherited role assignments.

Syntax: Here is the method signature:

```
public RoleAssignmentArray getInheritedAssignments()
```


getInheritedRoles

Gets the inherited roles.

Syntax: Here is the method signature:

```
public DNStringArray getInheritedRoles()
```

getLastName

Gets the last name.

Syntax: Here is the method signature:

```
public java.lang.String getLastName()
```

getRoleAssignments

Gets the role assignments for this identity.

Syntax: Here is the method signature:

```
public RoleAssignmentArray getRoleAssignments()
```

setAssociatedRoles

Sets the associated roles for this identity.

Syntax: Here is the method signature:

```
public void setAssociatedRoles(DNStringArray associatedRoles)
```

setCn

Sets the CN.

Syntax: Here is the method signature:

```
public void setCn(java.lang.String cn)
```

setContainerRoles

Sets the container roles.

Syntax: Here is the method signature:

```
public void setContainerRoles(DNStringArray containerRoles)
```

setEmail

Sets the email address.

Syntax: Here is the method signature:

```
public void setEmail(java.lang.String email)
```

setEntityKey

Sets the identity entity key.

Syntax: Here is the method signature:

```
public void setEntityKey(java.lang.String entityKey)
```

setExplicitAssignments

Sets the explicit role assignments.

Syntax: Here is the method signature:

```
public void setExplicitAssignments(RoleAssignmentArray explicitAssignments)
```

setFirstName

Sets the first name.

Syntax: Here is the method signature:

```
public void setFirstName(java.lang.String firstName)
```

setGroupRoles

Sets the group roles.

Syntax: Here is the method signature:

```
public void setGroupRoles(DNStringArray groupRoles)
```

setIdentityType

Sets the identity type.

Syntax: Here is the method signature:

```
public void setIdentityType(IdentityType identityType)
```

setImplicitAssignments

Sets the implicit role assignments.

Syntax: Here is the method signature:

```
public void setImplicitAssignments(RoleAssignmentArray implicitAssignments)
```

setInheritedAssignments

Sets the inherited role assignments.

Syntax: Here is the method signature:

```
public void setInheritedAssignments(RoleAssignmentArray inheritedAssignments)
```

setInheritedRoles

Sets the inherited roles.

Syntax: Here is the method signature:

```
public void setInheritedRoles(DNStringArray inheritedRoles)
```

setLastName

Sets the last name.

Syntax: Here is the method signature:

```
public void setLastName(java.lang.String lastName)
```

setRoleAssignments

Sets the role assignments for this identity.

Syntax: Here is the method signature:

```
public void setRoleAssignments(RoleAssignmentArray roleAssignments)
```

15.2.45 VersionVO

A value object for Version.

VersionVO constructors

The VersionVO class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
VersionVO()
```

Syntax 2: Here is the syntax for a constructor that takes a String as a parameter:

```
VersionVO(java.lang.String version)
```

getValue

Gets the version.

Syntax: Here is the method signature:

```
public java.lang.String getValue()
```

setValue

Sets the version.

Syntax: Here is the method signature:

```
public void setValue(java.lang.String version)
```

15.3 Role Web Service Examples

This section provides examples that demonstrate how you might use the Role service.

15.3.1 Retrieving Roles for a Group

This example shows how to retrieve the role assignments for a given group:

```
public void getGroupTestCase()
    throws Exception
{
    System.out.println("\n*****Calling
getGroupTestCase()*****");
    String groupDN = "cn=HR,ou=groups,ou=medical-idmsample,o=netiq";
    try
    {
        IRemoteRole stub = getRolesStub(username, password, acceptlanguage);
        Group group = stub.getGroup(groupDN);
        //Assert.assertNotNull("Group not found", group);
        if (group != null)
        {
            System.out.println("Group Found:");
            System.out.println("    entityKey          : " + group.getEntityKey());
            System.out.println("    identityType       : " +
group.getIdentityType().getValue());
            System.out.println("    description        : " + group.getDescription());

            DNString[] roles = group.getAssociatedRoles().getDnstring();
            if (roles != null)
            {
                System.out.println("no of associated roles: " + roles.length);
                for (int rIndex = 0; rIndex < roles.length; rIndex++)
                {
                    System.out.println("    role: " + rIndex);
                }
            }
            else
            {
                System.out.println("no of associated roles:0");
            }

            RoleAssignment[] assignments =
group.getRoleAssignments().getRoleassignment();
            PrintRoleUtils.getAssignments(assignments);
        }
        else
            System.out.println("Group not found");
    }
    catch (NrfServiceException nrf)
    {
        throw new Exception(nrf.getMessage());
    }
    catch (RemoteException re)
    {
        throw new Exception(re.getMessage());
    }
}
```

```

...
/**
 * Returns the Roles remote stub
 * @param username - user name
 * @param password - password
 * @param acceptLanguage - HTTP header Accept-Language
 * @return the Roles remote stub
 * @throws Exception - catch all exceptions
 */
public static IRemoteRole getRolesStub(String username,
                                       String password,
                                       String acceptLanguage)
    throws Exception
{
    Stub stub = null;
    String stubCacheKey = username + ":" + password;
    if (g_rolesStubCache.containsKey(stubCacheKey)) {
        g_log.debug("Using Cached Roles stub for [" + username + "]");
        stub = (Stub) g_rolesStubCache.get(stubCacheKey);
    } else {
        g_log.debug("Using New Roles stub");
        RoleService service = new RoleServiceImpl();
        stub = (Stub) service.getIRemoteRolePort();

        if (username != null && password != null) {
            stub._setProperty(Stub.USERNAME_PROPERTY, username);
            stub._setProperty(Stub.PASSWORD_PROPERTY, password);
        }

        stub._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY,
ServletParameters.getInstance().getUserAppUrl() + ROLES_SERVICE);
        stub._setProperty(Stub.SESSION_MAINTAIN_PROPERTY, Boolean.TRUE);

        g_rolesStubCache.put(stubCacheKey, stub);
    }

    Properties props = new Properties();
    props.setProperty("Accept-Language", acceptLanguage);
    stub._setProperty(Stub.HTTP_HEADERS, props);

    return (IRemoteRole) stub;
}

```

15.3.2 Retrieving Role Assignment Request Status

Returns a list of role assignment request status instances given a correlation ID.

```

public void getRoleAssignmentRequestStatusTestCase()
    throws Exception
{
    System.out.println("\n*****Calling

getRoleAssignmentRequestStatusTestCase()*****");
    String correlationId = "9a5feec728864b55ac443724a915e831";
    try
    {
        IRemoteRole stub = getRoleStub(url, username, password);
        RoleAssignmentRequestStatusArray reqArray =
stub.getRoleAssignmentRequestStatus(correlationId);
        RoleAssignmentRequestStatus[] reqStatus =
reqArray.getRoleassignmentrequeststatus();
        //Assert.assertNotNull("RoleAssignmentRequestStatus object is null for

getRoleAssignmentRequestStatus", reqStatus);
        if (reqStatus != null)
            System.out.println(PrintRoleUtils.getRequestStatus(reqStatus));
        else
            System.out.println("RoleAssignmentRequestStatus object is null for

getRoleAssignmentRequestStatus");

        //result += Util.getRequestStatus(reqStatus);
    }
    catch (NrfServiceException nrf)
    {
        throw new Exception(nrf.getMessage());
    }
    catch (RemoteException re)
    {
        throw new Exception(re.getMessage());
    }
}

```

15.3.3 Retrieving Type Information for a Role Assignment

This example shows how to retrieve the type for a role assignment:

```

public void getRoleAssignmentTypeInfoTestCase()
    throws Exception
{
    System.out.println("\n*****Calling

getRoleAssignmentTypeInfoTestCase()*****");
    try
    {
        IRemoteRole stub = getRoleStub(url, username, password);

        RoleAssignmentTypeInfo info =

stub.getRoleAssignmentTypeInfo(RoleAssignmentType.fromValue("ROLE_TO_ROLE"));
        //Assert.assertNotNull("Role Assignment Type Info Not Found for

```

```

getRoleAssignmentTypeInfo", info);
    if (info != null)
    {
        System.out.println("Role Assignment Type Info:");
        System.out.println("        identity type: " +
info.getIdentityType().getValue());
        System.out.println("        subtree included: " +
info.getSubtreeIncluded());
        System.out.println("        supports approvals: " +
info.getSupportsApproval());
        System.out.println("        supports effective date: " +
info.getSupportsEffectiveDate());
        System.out.println("        supports expiration: " +
info.getSupportsExpiration());
        System.out.println("        supports SOD Approval: " +
info.getSupportsSODApproval());
    }
    else
        System.out.println("Role Assignment Type Info Not Found for
getRoleAssignmentTypeInfo");
}
catch (NrfServiceException nrf)
{
    throw new Exception(nrf.getMessage());
}
catch (RemoteException re)
{
    throw new Exception(re.getMessage());
}
}
}

```

15.3.4 Retrieving Role Categories

This example shows how to retrieve the defined role categories:

```

public void getRoleCategoriesTestCase()
    throws Exception
{
    System.out.println("\n*****Calling
getRoleCategoriesTestCase()*****");
    try
    {
        IRemoteRole stub = getRoleStub(url, username, password);
        CategoryArray entriesArray = stub.getRoleCategories();
        Category[] entries = entriesArray.getCategory();
        Assert.assertNotNull("No categories found.", entries);
        if (entries != null)
        {
            System.out.println("no of categories:" + entries.length);

            for (int i = 0; i < entries.length; i++)
            {
                System.out.println("    category key   : " + entries[i].getCategoryKey());
                System.out.println("    category label: " + entries[i].getCategoryLabel());
            }
        }
    }
}

```

```

    }
    else
        System.out.println("No categories found.");
    }
    catch (NrfServiceException nrf)
    {
        throw new Exception(nrf.getMessage());
    }
    catch (RemoteException re)
    {
        throw new Exception(re.getMessage());
    }
}

```

15.3.5 Retrieving Role Levels

This example shows how to retrieve the defined role levels:

```

public void getRoleLevelsTestCase()
    throws Exception
{
    System.out.println("\n*****Calling
getRoleLevelsTestCase()*****");
    try
    {
        IRemoteRole stub = getRoleStub(url, username, password);
        RoleLevelArray roleLevelArray = stub.getRoleLevels();
        RoleLevel[] entries = roleLevelArray.getRolelevel();
        //Assert.assertNotNull("No role levels found.", entries);
        if (entries != null)
        {
            System.out.println("no of levels:" + entries.length);

            for (int index = 0; index < entries.length; index++)
            {
                System.out.println("    Level      : " + entries[index].getLevel());
                System.out.println("    Name       : " + entries[index].getName());
                System.out.println("    Description: " + entries[index].getDescription());
                System.out.println("    Container  : " + entries[index].getContainer());
            }
        }
        else
            System.out.println("No role levels found.");
    }
    catch (NrfServiceException nrf)
    {
        throw new Exception(nrf.getMessage());
    }
    catch (RemoteException re)
    {
        throw new Exception(re.getMessage());
    }
}

```


15.3.6 Verifying Whether a User Is In a Role

This example shows how to determine whether a user has been assigned to a role:

```
public void isUserInRoleTestCase()
    throws Exception
{
    System.out.println("\n*****Calling
isUserInRoleTestCase()*****");
    String[] DNS = {
        "cn=ablake,ou=users,ou=medical-idmsample,o=netiq",
        "cn=Doctor,cn=Level20,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=HajenDriver,cn=TestDrivers,o=netiq"
    };
    try
    {
        IRemoteRole stub = getRoleStub(url, username, password);
        boolean inRole = stub.isUserInRole(DNS[0], DNS[1]);

        String sInRole = "User Not In Role";
        if (inRole)
            sInRole = new String("User In Role");

        System.out.println(sInRole);
    }
    catch (NrfServiceException nrf)
    {
        throw new Exception(nrf.getMessage());
    }
    catch (RemoteException re)
    {
        throw new Exception(re.getMessage());
    }
}
```

16 Resource Web Service

This section describes the Resource Web Service, which allows SOAP clients to invoke a subset of actions that apply to resources.

16.1 About the Resource Web Service

The Resource Web Service exposes a small set of actions for the resource model. The service allows remote clients to request that a resource be granted or revoked, and also to check on the status of resource requests. By exposing these actions, the service makes it possible for a provisioning workflow to invoke resource requests through the Integration activity.

Calls to the Resource Web Service calls require HTTP authentication. By default, access to the resource service methods is restricted to Resource Administrators.

16.1.1 Accessing the Test Page

You can access the Resource Web Service endpoint using a URL similar to the following:

```
http://server:port/warcontext/resource/service?test
```

For example, if your server is named “myserver”, your User Application is listening on port 8080, and your User Application war file is named “IDMPROV”, the URL would be:

```
http://myserver:8080/IDMPROV/resource/service?test
```

WARNING: The test page is disabled by default. Since some of the methods allow data to be updated, the test page presents a potential security vulnerability and should not be allowed in a production environment.

Servlet declaration for the Resource Service

A SOAP service using WSSDK is deployed by adding the following declarations in the deployment descriptor (i.e. WEB-INF/web.xml):

```
<servlet>
  <servlet-name>Resource</servlet-name>
  <servlet-
class>com.novell.idm.nrf.soap.ws.role.impl.ResourceServiceSkeletonImpl</servlet-
class>

  <servlet-mapping>
    <servlet-name>Resource</servlet-name>
    <url-pattern>/resource/service</url-pattern>
  </servlet-mapping>
</servlet>
```

This follows the normal servlet declaration pattern. It indicates that the servlet `com.novell.idm.nrf.soap.ws.resource.impl.ResourceServiceSkeletonImpl` is deployed at `/resource/service`.

When a user reaches this servlet using a HTTP GET by entering `http://server-name/context/resource/service` (for example, `http://localhost:8080/IDMProv/resource/service`) in their browser, the WSSDK provides a page that exposes some information about the deployed service.

Enabling the Test Page

WARNING: The test page is disabled by default. Since some of the methods allow data to be updated, the test page presents a potential security vulnerability and should not be allowed in a production environment.

To enable the test page, you need to update the WEB-INF/web.xml file in the IDMProv.war file. Before you make your changes, the web.xml should look like this:

```
<servlet>
  <servlet-name>Resource</servlet-name>
  <servlet-
class>com.novell.idm.nrf.soap.ws.resource.impl.ResourceServiceSkeletonImpl</
servlet-class>
  <init-param>
    <param-name>com.novell.soa.ws.test.disable</param-name>
    <param-value>>true</param-value>
  </init-param>
</servlet>
```

Change the servlet declaration, as follows:

```
<servlet>
  <servlet-name>Resource</servlet-name>
  <servlet-
class>com.novell.idm.nrf.soap.ws.resource.impl.ResourceServiceSkeletonImpl</
servlet-class>
  <init-param>
    <param-name>com.novell.soa.ws.test.disable</param-name>
    <param-value>>false</param-value>
  </init-param>
</servlet>
```

16.1.2 Accessing the WSDL

You can access the WSDL for the Resource Web Service using a URL similar to the following:

```
http://server:port/warcontext/resource/service?wsdl
```

For example, if your server is named “myserver”, your User Application is listening on port 8080, and your User Application war file is named “IDMPROV”, the URL would be:

```
http://myserver:8080/IDMPROV/resource/service?wsdl
```

16.1.3 Generating the Stub Classes

Before using the Web Service, you need to use the WSSDK tool or another SOAP tool kit to generate the stub classes. To allow your code to find the stub classes, you also need to add the JAR that contains the stub classes to your classpath.

If you want to use the NetIQ WSSDK tool, you can generate the client stubs by extracting the WSDL and running the `wSDL2java` utility. For example, you could run this command to generate the stubs in a package called `com.novell.soa.af.resource.soap.impl`:

```
"C:\Program Files\Java\jdk1.6.0_31\bin\java" -cp "../lib/wssdk.jar;../lib/jaxrpc-api.jar;../lib/mail.jar;../lib/activation.jar";"c:\Program Files\Java\jdk1.6.0_31\lib\tools.jar"; com.novell.soa.ws.impl.tools.wSDL2java.Main -verbose -ds gensrc -d C:\ -noskel -notie -genclient -keep -package com.novell.soa.af.resource.soap.impl -javadoc resource.wsdl
```

You can change the `wSDL2java` parameters to suit your requirements.

16.1.4 Removing Administrator Credential Restrictions

The Resource Web Service supports two levels of security, one that restricts access to Resource Administrators, and another that restricts access to the authenticated user. The default setting restricts access to all operations to the Resource Administrator.

You can modify the settings for security configuration in the `ism-configuration.properties` file, located by default in the `/netiq/idm/apps/tomcat/conf` directory. Each property can be set to `true` or `false`. A value of `true` locks down the operation, whereas a value of `false` opens up the operation. After making your changes, you need to import the file back into the WAR file and redeploy.

You can open up the Resource Web Service to authenticated users by setting the `ResourceService/Resource/soap` property to `false`. To open up a particular operation to authenticated users, you need to set the property for that operation (`ResourceService/Resource/soap/operation`) to `false` as well. If you set all of the properties to `false`, you can open up all operations to authenticated users. The *operation* names are the same as the names of the methods supported by the service.

The following methods can be invoked by users without Resource Administrator credentials if the property `ResourceService/Resource/soap` is set to `false`:

- ◆ `requestResourceGrant`
- ◆ `requestResourceRevoke`
- ◆ `getResourceRequestStatusByCorrelationId`
- ◆ `getResourceRequestStatusForCurrentUser`
- ◆ `getResourceAssignmentsForCurrentUser`

If you wish to change the restriction for a particular operation, you can modify the property `ResourceService/Resource/soap/operation` for the method, setting its value to `true` to restrict access to administrators for the operation and `false` to remove the restriction. If the `ResourceService/Resource/soap` property is `true`, all methods are restricted to Resource Administrator credentials.

Example To ensure that the security configuration opens up all operations within the Resource Web Service, except for the `getResourceRequestStatusByIdentity` operation, which would only be accessible to the Resource Administrator, the `ism-configuration.properties` must have the following settings:

```
ResourceService/Resource/soap = false
ResourceService/Resource/soap/requestResourceGrant = false
ResourceService/Resource/soap/requestResourceRevoke = false
ResourceService/Resource/soap/getResourceRequestStatusByCorrelationId = false
ResourceService/Resource/soap/getResourceRequestStatusForCurrentUser = false
ResourceService/Resource/soap/getResourceRequestStatusByIdentity = true
```

16.2 Resource Web Service Interface

This section provides details about the methods available with the Resource Web service. This programming interface presumes you're using Java code generated by the WSSDK toolkit. The interface will be different if you're using another Web Service toolkit.

16.2.1 IRemoteResource

This section provides reference information for each method associated with the IRemoteResource interface.

checkCodeMapValueStatus

Checks to see if a particular value exists in the code map table for a specified entitlement and logical system. The method returns the status for the code map value as a CodeMapValueStatus object.

This method is one of three SOAP endpoints to help you keep the code map tables for the Roles Based Provisioning Module synchronized with the code map tables for the Role Mapping Administrator. The user interface for the Role Mapping Administrator can trigger a code map refresh if a mismatch is discovered while a user is creating mappings. In addition, the Roles Based Provisioning Module allows you to use the three SOAP endpoints to refresh selected entitlements within its code map tables.

In addition to checkCodeMapValueStatus, the Roles Based Provisioning Module includes the following endpoints to help with code map synchronization:

- ◆ getRefreshStatus
- ◆ refreshCodeMap

The **Entitlement Query Settings** section of the **Configure Roles and Resources Settings** page in the User Application allows you to specify how often the Roles Based Provisioning Module code map tables are refreshed and also start a manual refresh. However, this page does not allow to refresh selected entitlements. To control which entitlements are refreshed, you need to use the SOAP endpoints.

For additional information on the getRefreshStatus endpoint, see [“getRefreshStatus” on page 457](#). For additional information on the refreshCodeMap endpoint, see [“refreshCodeMap” on page 462](#).

For code samples that use the new methods for code map synchronization, see [Section 16.3.1, “Code Map Synchronization Code Samples,” on page 482](#).

Syntax: Here is the method signature:

```
public CodeMapValueStatus checkCodeMapValueStatus(String entitlementDN, String
connectionName, String codeMapValue)
    throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ♦ *entitlementDN* entitlement DN as a string.

For example:

```
cn=groups,cn=groupentitlementloopback,cn=driverset1,o=system
```

- ♦ *connectionName* connection (logical system) name. This is an optional parameter. Only fanout drivers need to specify the connection name.
- ♦ *codeMapValue* code map value to verify.

For example:

```
\\TEST1\data\groups\netiq\cambridge\rbpm\4AlphaGroup
```

SOAP Request: Here is the SOAP request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://www.netiq.com/resource/service">
<soapenv:Header/>
<soapenv:Body>
<ser:checkCodeMapValueStatusRequest>
<!--Optional:-->
<ser:entitlementDN>cn=groups,cn=groupentitlementloopback,cn=driverset1,o=system</
ser:entitlementDN>
<!--Optional:-->
<ser:connectionName/>
<!--Optional:-->
<ser:codeMapValue>\\WILLIAMS1\data\groups\netiq\cambridge\rbpm\4AlphaGroup</
ser:codeMapValue>
</ser:checkCodeMapValueStatusRequest>
</soapenv:Body>
</soapenv:Envelope>
```

SOAP Response: Here is the SOAP response:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
/XMLSchema-instance">
<SOAP-ENV:Body>
<ns1:checkCodeMapValueStatusResponse xmlns="http://www.netiq.com/resource/service"
xmlns:ns1="http://www.netiq.com/resource/service">
<result>
<refreshStatus>
<connectionName xsi:nil="1"/>
<entitlementDN>cn=groups,cn=groupentitlementloopback,cn=driverset1,o=system</
entitlementDN>
<guid>\\2d\\13\\d1\\a4\\7b\\99\\d6\\4c\\03\\9a\\2d\\13\\d1\\a4\\7b\\99</guid>
<lastRefresh>1329431650891</lastRefresh>
<status>SUCCESS</status>
</refreshStatus>
<upToDate>true</upToDate>
<value>\\WILLIAMS1\data\groups\netiq\cambridge\rbpm\4AlphaGroup</value>
</result>
</ns1:checkCodeMapValueStatusResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

createResource

Creates a new resource according to the specified parameters, and returns a DN of the created resource.

A correlation ID is generated automatically for this method that uses this format:

```
UserApp#RemoteResourceRequest#xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx
```

The correlation ID is used for auditing.

Syntax: Here is the method signature:

```
public String createResource(Resource resource)
    throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ◆ *resource* specifies the resource object to create.

createResourceAid

Creates a new resource, with a correlation ID that you provide. The correlation ID is used for auditing to link a set of related resources. This method creates the resource according to the specified parameters, and returns a DN of the created resource.

Syntax: Here is the method signature:

```
public String createResourceAid(Resource resource, String correlationId)
    throws NrfServiceException, RemoteException;
```

findResourceByExampleWithOperator

Finds all Resource objects based on the search criteria specified in the given Resource object.

Syntax: Here is the method signature:

```
public Resource[] findResourceByExampleWithOperator(Resource searchCriteria,
    boolean useAndForMultiValueSearch)
    throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ◆ *searchCriteria* specifies Query by Example (QBE) search criteria within a Resource object.
- ◆ *useAndForMultiValueSearch* determines whether AND or OR will be used for multi-value search expressions. If you specify a value of true, AND will be used for multi-value searches; if you specify a value of false, OR will be used.

getEntitlementCodeMap

Returns an array of ProvisioningCodeMap objects, which include code map information from the code map and code map label tables.

Syntax: Here is the method signature:

```
ProvisioningCodeMap[] getEntitlementCodeMap(java.lang.String codeMapKey, int type)
    throws com.novell.idm.nrf.soap.ws.resource.NrfServiceException,
    java.rmi.RemoteException;
```


The parameters are described below:

- ♦ *codeMapKey* specifies the code map key to retrieve values from. The codeMapKey is a GUID that acts as a unique identifier for the code map. For example:

```
\2d\13\d1\a4\7b\99\d6\4c\03\9a\2d\13\d1\a4\7b\99
```

- ♦ *type* specifies the code map type. A value of 0 filters the list to include entitlement code maps only.

getRefreshStatus

Gets the refresh status of a code map based on a specified entitlement DN. This method returns the status as an array of CodeMapRefreshStatus objects. The structure returned contains the DN, GUID, connection name status, and last refresh time.

This method is one of three SOAP endpoints to help you keep the code map tables for the Roles Based Provisioning Module synchronized with the code map tables for the Role Mapping Administrator. The user interface for the Role Mapping Administrator can trigger a code map refresh if a mismatch is discovered while a user is creating mappings. In addition, the Roles Based Provisioning Module allows you to use the three SOAP endpoints to refresh selected entitlements within its code map tables.

In addition to getRefreshStatus, the Roles Based Provisioning Module includes the following endpoints to help with code map synchronization:

- ♦ checkCodeMapValueStatus
- ♦ refreshCodeMap

The **Entitlement Query Settings** section of the **Configure Roles and Resources Settings** page in the User Application allows you to specify how often the Roles Based Provisioning Module code map tables are refreshed and also start a manual refresh. However, this page does not allow to refresh selected entitlements. To control which entitlements are refreshed, you need to use the SOAP endpoints.

For additional information on the checkCodeMapValueStatus endpoint, see [“checkCodeMapValueStatus” on page 454](#). For additional information on the refreshCodeMap endpoint, see [“refreshCodeMap” on page 462](#).

For code samples that use the new methods for code map synchronization, see [Section 16.3.1, “Code Map Synchronization Code Samples,” on page 482](#).

Syntax: Here is the method signature:

```
public CodeMapRefreshStatus[] getRefreshStatus(String entitlementDN)
    throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ♦ *entitlementDN* entitlement DN as a string

For example:

```
cn=groups,cn=groupentitlementloopback,cn=driverset1,o=system
```

SOAP Request: Here is the SOAP request:

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://www.netiq.com/resource/service">
<soapenv:Header/>
<soapenv:Body>
<ser:getRefreshStatusRequest>
<!--Optional:-->
<ser:entitlementDN>cn=groups,cn=grouppentitlementloopback,cn=driverset1,o=system</
ser:entitlementDN>
</ser:getRefreshStatusRequest>
</soapenv:Body>
</soapenv:Envelope>

```

SOAP Response: Here is the SOAP response:

```

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
/XMLSchema-instance">
<SOAP-ENV:Body>
<ns1:getRefreshStatusResponse xmlns="http://www.netiq.com/resource/service"
xmlns:ns1="http://www.netiq.com/resource/service">
<result>
<codemaprefreshstatus>
<connectionName xsi:nil="1"/>
<entitlementDN>cn=groups,cn=grouppentitlementloopback,cn=driverset1,o=system</
entitlementDN>
<guid>\2d\13\d1\a4\7b\99\d6\4c\03\9a\2d\13\d1\a4\7b\99</guid>
<lastRefresh>1329100366090</lastRefresh>
<status>SUCCESS</status>
</codemaprefreshstatus>
</result>
</ns1:getRefreshStatusResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

getResourceAssignmentsForCurrentUser

Returns the resource assignments for the current user.

Syntax: Here is the method signature:

```

ResourceAssignment[] getResourceAssignmentsForCurrentUser()
    throws com.novell.idm.nrf.soap.ws.resource.NrfServiceException,
java.rmi.RemoteException;

```

getResourceAssignmentsForUser

Returns the resource assignments for a particular user.

Syntax: Here is the method signature:

```

ResourceAssignment[] getResourceAssignmentsForUser(java.lang.String userDn)
    throws com.novell.idm.nrf.soap.ws.resource.NrfServiceException,
java.rmi.RemoteException;

```

The parameters are described below:

- ♦ *userDn* DN of the target user

getAssignmentsForResource

Returns the resource assignments for a particular resource.

Syntax: Here is the method signature:

```
ResourceAssignment[] getAssignmentsForResource(java.lang.String resourceDn)
    throws com.novell.idm.nrf.soap.ws.resource.NrfServiceException,
    java.rmi.RemoteException;
```

The parameters are described below:

- ♦ *resourceDn* DN of the target resource

getResourceRequestStatusByCorrelationId

Returns all resource request status items for a given correlation ID.

Syntax: Here is the method signature:

```
public ResourceAssignmentRequestStatus[]
    getResourceRequestStatusByCorrelationId
        (String correlationId, String locale)
        throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ♦ *correlationId* specifies a resource assignment request correlation ID.
- ♦ *locale* supplies an iso639 language code to format localized string values; if the parameter is null, the language defaults to the servlet request locale.

This method returns all resource request status instances for the specified *correlationId* parameter value. For more information on the `ResourceAssignmentRequestStatus` class, see [Section 16.2.9, "ResourceAssignmentRequestStatus," on page 479](#).

getResourceRequestsStatusForCurrentUser

Returns all resource request status items for the authenticated user.

Syntax: Here is the method signature:

```
public ResourceAssignmentRequestStatus[]
    getResourceRequestsStatusForCurrentUser(String locale)
    throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ♦ *locale* supplies an iso639 language code to format localized string values; if the parameter is null, the language defaults to the servlet request locale.

This method returns all resource request status instances for the specified *correlationId* parameter value. For more information on the `ResourceAssignmentRequestStatus` class, see [Section 16.2.9, "ResourceAssignmentRequestStatus," on page 479](#).

getResourceRequestStatusByIdentity

Returns all resource assignment request status items for a particular user identity.

Syntax: Here is the method signature:

```
public ResourceAssignmentRequestStatus[]
    getResourceRequestStatusByIdentity(String identity, String locale)
        throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ♦ *identity* specifies the DN for a user.
- ♦ *locale* supplies an iso639 language code to format localized string values; if the parameter is null, the language defaults to the servlet request locale.

This method returns all resource request status instances for the specified *correlationId* parameter value. For more information on the `ResourceAssignmentRequestStatus` class, see [Section 16.2.9, “ResourceAssignmentRequestStatus,” on page 479](#).

getCodeMapValues

Returns a list of code map values for a specified code map.

Syntax: Here is the method signature:

```
public CodeMapValue[] getCodeMapValues(String codeMapKey, String locale, dn,
    nativeValue, logicalSystemId)
    throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ♦ *codeMapKey* specifies the code map key to retrieve values from. The *codeMapKey* is a GUID that acts as a unique identifier for the code map. For example:

```
\2d\13\d1\a4\7b\99\d6\4c\03\9a\2d\13\d1\a4\7b\99
```

- ♦ *locale* supplies an iso639 language code to format localized string values; if the parameter is null, the language defaults to the servlet request locale.

getResource

Returns a resource object.

Syntax: Here is the method signature:

```
public Resource getResource(String dn, String locale)
    throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ♦ *dn* specifies the DN of the resource you want to retrieve.
- ♦ *locale* supplies an iso639 language code to format localized string values; if the parameter is null, the language defaults to the servlet request locale.

getResourceLocalizedStrings

Gets the localized strings for a resource, such as the names and descriptions. The type parameter lets you specify whether the names or descriptions should be retrieved.

Syntax: Here is the method syntax:

```
public LocalizedValue[] getResourceLocalizedStrings(String resourceDn, int
    type) throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ♦ *resourceDn* specifies the DN of the resource for which you want to get the localized strings.
- ♦ *type* specifies the type of localized strings you want to retrieve. A type value of 1 retrieves a list of names for the resource, whereas a type value of 2 retrieves a list of descriptions.

getResourceInfoByCategory

Returns a list of ResourceInfo instances given a list of category keys.

Syntax: Here is the method signature:

```
public ResourceInfo[] getResourceInfoByCategory(CategoryKey[]  
resourceCategoryKeys)  
    throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ♦ *resourceCategoryKeys* specifies the list of resource category keys to retrieve resource information objects for.

getResourceInfo

Returns a list of ResourceInfo instances given a list of resource DNs.

Syntax: Here is the method signature:

```
public ResourceInfo[] getResourceInfo(DNString[] resDns)  
    throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ♦ *resDns* provides a list of resource DNs for which you want to retrieve resource information objects.

modifyResource

Modifies a resource definition. This method does not perform a localized string modification update. To update the localized names or descriptions for a resource, you need to use the `setResourceLocalizedStrings` method.

A correlation ID is generated automatically for this method that uses this format:

```
UserApp#RemoteResourceRequest#xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

Syntax: Here is the method signature:

```
public Resource modifyResource(Resource resource)  
    throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ♦ *resource* specifies the resource object to modify.

modifyResourceAid

Modifies a resource definition, with a correlation ID that you provide. The correlation ID is used for auditing to link a set of related resources. This method does not perform a localized string modification update. To update the localized names or descriptions for a resource, you need to use the `setResourceLocalizedStrings` method.

Syntax: Here is the method signature:

```
public Resource modifyResourceAid(Resource resource, String correlationId)
    throws NrfServiceException, RemoteException;
```

refreshCodeMap

Refreshes the code map based on a specified entitlement DN. The method returns the status of the refresh operation in the form of an `EntitlementRefreshInfo` object. This structure includes the detailed status as an array of `CodeMapRefreshStatus` objects.

This method is one of three SOAP endpoints to help you keep the code map tables for the Roles Based Provisioning Module synchronized with the code map tables for the Role Mapping Administrator. The user interface for the Role Mapping Administrator can trigger a code map refresh if a mismatch is discovered while a user is creating mappings. In addition, the Roles Based Provisioning Module allows you to use the three SOAP endpoints to refresh selected entitlements within its code map tables.

In addition to `refreshCodeMap`, the Roles Based Provisioning Module includes the following endpoints to help with code map synchronization:

- ◆ `checkCodeMapValueStatus`
- ◆ `getRefreshStatus`

The **Entitlement Query Settings** section of the **Configure Roles and Resources Settings** page in the User Application allows you to specify how often the Roles Based Provisioning Module code map tables are refreshed and also start a manual refresh. However, this page does not allow you to refresh selected entitlements. To control which entitlements are refreshed, use the SOAP endpoints.

For additional information on the `checkCodeMapValueStatus` endpoint, see [“checkCodeMapValueStatus” on page 454](#). For additional information on the `getRefreshStatus` endpoint, see [“getRefreshStatus” on page 457](#).

For code samples that use the new methods for code map synchronization, see [Section 16.3.1, “Code Map Synchronization Code Samples,” on page 482](#).

Syntax: Here is the method signature:

```
public EntitlementRefreshInfo refreshCodeMap(String entitlementDN rbpm-refresh-
rate="value" freeform-param="value")
    throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ◆ *entitlementDN* entitlement DN to refresh the code map

For example:

```
cn=groups,cn=groupentitlementloopback,cn=driverset1,o=system
```

- ♦ `rbpm-refresh-rate` allows you to control the automatic or manual rate at which the entitlement is refreshed. Specify `0` to disable automatic refreshing. Specify `-111111` to disable both automatic and manual refreshing.
- ♦ `freeform-param` allows you to control whether the code map refresh removes entries from the database when the entitlement type is valued and the values were loaded directly into the database. Specify `false` if you do not want the refresh to remove the values from the database. Specify `true` if you want the refresh to remove the values from the database. The default value is `true`.

SOAP Request: Here is the SOAP request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://www.netiq.com/resource/service">
<soapenv:Header/>
<soapenv:Body>
<ser:refreshCodeMapRequest>
<!--Optional:-->
<ser:entitlementDN>cn=groups,cn=groupentitlementloopback,cn=driverset1,o=system</
ser:entitlementDN>
</ser:refreshCodeMapRequest>
</soapenv:Body>
</soapenv:Envelope>
```

SOAP Response: Here is the SOAP request:

```
<SOAP-ENV:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
<SOAP-ENV:Body>
<ns1:refreshCodeMapResponse xmlns="http://www.netiq.com/resource/service"
xmlns:ns1="http://www.netiq.com/resource/service">
<result>
<detailedStatus>
<codemaprefreshstatus>
<connectionName xsi:nil="1"/>
<entitlementDN>cn=groups,cn=groupentitlementloopback,cn=driverset1,o=system</
entitlementDN>
<guid>\2d\13\d1\a4\7b\99\d6\4c\03\9a\2d\13\d1\a4\7b\99</guid>
<lastRefresh>1329244784180</lastRefresh>
<status>SUCCESS</status>
</codemaprefreshstatus>
</detailedStatus>
<entitlementDN>cn=groups,cn=groupentitlementloopback,cn=driverset1,o=system</
entitlementDN>
<guid>\2d\13\d1\a4\7b\99\d6\4c\03\9a\2d\13\d1\a4\7b\99</guid>
<status>>true</status>
</result>
</ns1:refreshCodeMapResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

removeResource

Deletes a specified resource from the Resource Catalog. Returns the DN for the deleted resource as a confirmation.

A correlation ID is generated automatically for this method that uses this format:

UserApp#RemoteResourceRequest#xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx

The correlation ID is used for auditing.

Syntax: Here is the method signature:

```
public DNString removeResource(DNString resourceDn)
    throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ◆ *resourceDn* specifies the DN of the resource to delete.

removeResourceAid

Deletes a specified resource from the Resource Catalog, with a correlation ID that you provide. The correlation ID is used for auditing to link a set of related resources. This method returns the DN for the deleted resource as a confirmation.

Syntax: Here is the method signature:

```
public DNString removeResourceAid(DNString resourceDn, String correlation Id)
    throws NrfServiceException, RemoteException;
```

requestResourceGrant

Makes a grant resource request and returns a resource request correlation ID.

Syntax: Here is the method signature:

```
public String requestResourceGrant(String resourceTarget, String requester, String
userTarget, String reasonForRequest,
    ResourceRequestParam[] requestParams, String correlationId)
    throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ◆ *resourceTarget* specifies the target resource DN.
- ◆ *requester* supplies an identifier for the remote client application making the request to grant the resource.

The *requester* parameter on this SOAP endpoint identifies the originator of the request. This value is set in the resource request object *nrfOriginator* attribute, following this convention:

- ◆ For a SOAP call: "REMOTE_CLIENT:<*requester param value*>"
- ◆ For a workflow action: "WF:<*wf process id*>"
- ◆ For the user application user interface: "USER_APP"
- ◆ *userTarget* specifies the DN for the being granted the resource.
- ◆ *reasonForRequest* provides a reason for the request.
- ◆ *requestParams* provides the parameter values for the request.
- ◆ *correlationId* specifies a resource assignment request correlation ID; if the parameter is null, a correlation ID is generated.

The *requester* parameter is a client-supplied identifier for the agent making the request. For example, an identifier such as *IRemote-MyApplicationName* might be used to identify a request from *MyApplicationName*. The *requestParams* are the dynamic parameter values required by the resource to make a request. If no values are required, the parameter value can be null or an empty

array. The *correlationId* allows a client to group request for the purpose of checking the status. If the parameter value is null, the service generates a unique correlation id. The correlation id is returned to the caller.

requestResourceRevoke

Makes a revoke resource request and returns a resource request correlation ID.

The revoke invocation behavior mirrors the behavior for a grant operation, except that a revoke request for the resource is posted on the server.

Syntax: Here is the method signature:

```
public String requestResourceRevoke(String resourceTarget,
    String requester, String userTarget, String reasonForRequest,
    ResourceRequestParam[] requestParams, String instanceGuid, String
correlationId)
    throws NrfServiceException, RemoteException;
```

The parameters are described below:

- ♦ *resourceTarget* specifies the target resource DN.
- ♦ *requester* supplies an identifier for the remote client application making the request to revoke the resource.

The *requester* parameter on this SOAP endpoint identifies the originator of the request. This value is set in the resource request object *nrfOriginator* attribute, following this convention:

- ♦ For a SOAP call: "REMOTE_CLIENT:<*requester param value*>"
- ♦ For a workflow action: "WF:<*wf process id*>"
- ♦ For the user application user interface: "USER_APP"
- ♦ *userTarget* specifies the DN for the user being granted the resource.
- ♦ *reasonForRequest* provides a reason for the request.
- ♦ *requestParams* provides the parameter values for the request.
- ♦ *instanceGuid* provides a GUID identifier for the resource assignment instance. The resource assignment instance GUID supports revoking a single instance of a multi-value resource assignment, if not all instances are to be revoked.

IMPORTANT: If you do not specify the *instanceGuid* value, and the user has more than one value of that resource assigned, all instances of the resource assignment will be removed.

When you create a new resource assignment request, the *instanceGuid* is included just above the *correlationid* field:

```
<ser:instanceGuid></ser:instanceGuid>
```

You need to specify the instance of the resource you want to revoke by supplying the value in the *instanceGuid* parameter.

To find out which resources are assigned to a user, you need to use the *getResourceAssignmentsForUser* method. This method returns the following data structure, which also includes the *instanceGuid*:

```

<resourceassignment>
  <instanceGuid>1b335aa9f4a14bd4a2a802eb4ba092da</instanceGuid>
  <reason>3b-Test</reason>
  <recipientDn>cn=ablake,ou=users,o=data</recipientDn>
  <requestDate>2011-08-18T14:25:21</requestDate>
  <requestParams>
    <resourcerequestparam>
      <name>param1</name>
      <value>3a3a</value>
    </resourcerequestparam>
  </requestParams>
  <requesterDn>cn=uaadmin,ou=sa,o=data</requesterDn>

  <resourceDn>cn=Vodacom,cn=ResourceDefs,cn=RoleConfig,cn=AppConfig,cn=User
  Application Driver,cn=driverset1,o=system</resourceDn>
</resourceassignment>

```

- ♦ *correlationId* specifies a resource assignment request correlation ID; if the parameter is null, a correlation ID is generated.

setResourceLocalizedStrings

Sets the localized strings for a resource, such as the names and descriptions.

A correlation ID is generated automatically for this method that uses this format:

```
UserApp#RemoteResourceRequest#xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

Syntax: Here is the method signature:

```

public LocalizedValue[] setResourceLocalizedStrings(String resourceDn,
  LocalizedValue[] locStrings, int type)
  throws NrfServiceException, RemoteException;

```

The parameters are described below:

- ♦ *resourceDn* specifies the DN of the resource for which you want to set the localized strings.
- ♦ *locStrings* provides an array of localized strings you want to define.
- ♦ *type* specifies the type of localized strings you want to retrieve. A type value of 1 retrieves a list of names for the resource, whereas a type value of 2 retrieves a list of descriptions.

setResourceLocalizedStringsAid

Sets the localized strings for a resource, such as the names and descriptions, with a correlation ID that you provide. The correlation ID is used for auditing to link a set of related resources.

Syntax: Here is the method signature:

```

public LocalizedValue[] setResourceLocalizedStringsAid(String resourceDn,
  LocalizedValue[] locStrings, int type, String correlationId)
  throws NrfServiceException, RemoteException;

```

16.2.2 CodeMapRefreshStatus

Supporting class that provides details about the status of a code map refresh.

getConnectionName

Returns the name of the connected system.

Syntax: Here is the method signature:

```
public String getConnectionName()
```

getEntitlementDN

Returns the DN for the entitlement.

Syntax: Here is the method signature:

```
public String getEntitlementDN()
```

getGuid

Returns the GUID for the entitlement.

Syntax: Here is the method signature:

```
public String getGuid()
```

getLastRefresh

Returns the timestamp for the last refresh.

Syntax: Here is the method signature:

```
public long getLastRefresh()
```

getStatus

Returns the refresh status as a string indicating whether the refresh was successful.

Syntax: Here is the method signature:

```
public String getStatus()
```

setConnectionName

Sets the name of the connection system.

Syntax: Here is the method signature:

```
public void setConnectionName(final String connectionName)
```

setEntitlementDN

Sets the entitlement DN.

Syntax: Here is the method signature:

```
public void setEntitlementDN(String entitlementDN)
```

setGuid

Sets the GUID for the entitlement.

Syntax: Here is the method signature:

```
public void setGuid(String guid)
```

setLastRefresh

Sets the last refresh timestamp.

Syntax: Here is the method signature:

```
public void setLastRefresh(final long lastRefresh)
```

setStatus

Sets the refresh status.

Syntax: Here is the method signature:

```
public void setStatus(String status)
```

16.2.3 CodeMapValueStatus

Supporting class that provides details about the status of a refresh for a code map value.

getUpToDate

Returns true or false to indicate whether the status is up-to-date.

Syntax: Here is the method signature:

```
public boolean getUpToDate()
```

getRefreshStatus

Returns the refresh status as a CodeMapRefreshStatus object.

Syntax: Here is the method signature:

```
public CodeMapRefreshStatus getRefreshStatus()
```

getValue

Gets the code map value.

Syntax: Here is the method signature:

```
public String getValue()
```

setRefreshStatus

Sets the refresh as a CodeMapRefreshStatus object.

Syntax: Here is the method signature:

```
public void setRefreshStatus(final CodeMapRefreshStatus refreshStatus)
```

setUpToDate

Sets a boolean indicating whether the status is up-to-date.

Syntax: Here is the method signature:

```
public void setUpToDate(final boolean upToDate)
```

setValue

Sets the code map value.

Syntax: Here is the method signature:

```
public void setValue(final String value)
```

16.2.4 EntitlementRefreshInfo

Supporting class that provides refresh information for an entitlement after a code map refresh has been performed.

getDetailedStatus

Returns the detailed status as an array of CodeMapRefreshStatus objects.

Syntax: Here is the method signature:

```
public CodeMapRefreshStatus[] getDetailedStatus()
```

getEntitlementDN

Returns the DN for the entitlement.

Syntax: Here is the method signature:

```
public String getEntitlementDN()
```

getGuid

Returns the GUID for the entitlement.

Syntax: Here is the method signature:

```
public String getGuid()
```

getStatus

Returns the status of the refresh as a boolean flag.

Syntax: Here is the method signature:

```
public boolean getStatus()
```

setDetailedStatus

Sets the detailed status as an array of CodeMapRefreshStatus objects.

Syntax: Here is the method signature:

```
public void setDetailedStatus(final CodeMapRefreshStatus[] detailedStatus)
```

setEntitlementDN

Sets the DN for the entitlement.

Syntax: Here is the method signature:

```
public void setEntitlementDN(String entitlementDN)
```

setGuid

Sets the GUID for the entitlement.

Syntax: Here is the method signature:

```
public void setGuid(String m_guid)
```

setStatus

Sets the status as a boolean flag.

Syntax: Here is the method signature:

```
public void setStatus(boolean m_status)
```

16.2.5 ProvisioningCodeMap

Value class to hold code map information from the code map and code map label tables.

getDescription

Returns the description

```
public String getDescription()
```

getName

Returns the name.

```
public String getName()
```

getEntityKey

Returns the entity key.

```
public String getEntityKey()
```

getEntityType

Returns the entity type.

```
public int getEntityType()
```

getQueryKey

Returns the query key.

```
public String getQueryKey()
```

getViewId

Returns the view ID.

```
public String getViewId()
```

getLastRefreshed

Returns the timestamp for the last refresh.

```
public long getLastRefreshed()
```

setDescription

Sets the description.

```
public void setDescription(String description)
```

setName

Sets the name.

```
public void setName(String name)
```

setEntityKey

Sets the entity key.

```
public void setEntityKey(String entityKey)
```

setEntityType

Sets the entity type.

```
public void setEntityType(int entityType)
```

setQueryKey

Sets the query key.

```
public void setQueryKey(String queryKey)
```

setViewId

Sets the view ID.

```
public void setViewId(String viewId)
```

setLastRefreshed

Sets the timestamp for the last refresh.

```
public void setLastRefreshed(long lastRefreshed)
```

getLabels

Returns the code map labels.

```
public ProvisioningCodeMapLabel[] getLabels()
```

setLabels

Sets the code map labels.

```
public void setLabels(ProvisioningCodeMapLabel[] labels)
```

getEntitlementDn

Returns the DN for the entitlement.

```
public String getEntitlementDn()
```

setEntitlementDn

Sets the DN for the entitlement.

```
public void setEntitlementDn(String entitlementDn)
```

getDriverDn

Returns the DN for the driver.

```
public String getDriverDn()
```

setDriverDn

Sets the DN for the driver.

```
public void setDriverDn(String driverDn)
```

getDriverDisplayName

Returns the display name for the driver.

```
public String getDriverDisplayName()
```


setDriverDisplayName

Sets the display name for the driver.

```
public void setDriverDisplayName(String driverDisplayName)
```

16.2.6 Resource

Supporting class that provides information about resources.

getName

Returns the name of the resource.

```
public String getName()
```

setName

Sets the name of the resource.

```
public void setName(String name)
```

getDescription

Returns the description of the resource.

```
public String getDescription()
```

setDescription

Sets the description of the resource.

```
public void setDescription(String description)
```

getEntityKey

Returns the entity key for the resource.

```
public String getEntityKey()
```

setEntityKey

Sets the entity key for the resource.

```
public void setEntityKey(String entityKey)
```

getResourceCategoryKeys

Returns the keys for the resource categories.

```
public CategoryKey[] getResourceCategoryKeys()
```

setResourceCategoryKeys

Sets the keys for the resource categories.

```
public void setResourceCategoryKeys(CategoryKey[] resourceCategoryKeys)
```

getEntitlementRef

Returns the entitlement reference for the resource.

```
public NrfEntitlementRef[] getEntitlementRef()
```

setEntitlementRef

Sets the entitlement reference for the resource.

```
public void setEntitlementRef(NrfEntitlementRef[] entitlementRef)
```

getGrantApprovers

Returns the list of approvers for resource grant operations.

```
public Approver[] getGrantApprovers()
```

setGrantApprovers

Sets the list of approvers for resource grant operations.

```
public void setGrantApprovers(Approver[] grantApprovers)
```

getGrantQuorum

Returns the quorum condition for grant operations.

```
public String getGrantQuorum()
```

setGrantQuorum

Sets the quorum condition for grant operations.

```
public void setGrantQuorum(String grantQuorum)
```

getGrantRequestDef

Returns the provisioning request definition for grant operations.

```
public String getGrantRequestDef()
```

setGrantRequestDef

Sets the provisioning request definition for grant operations.

```
public void setGrantRequestDef(String grantRequestDef)
```

getRevokeQuorum

Returns the quorum condition for revoke operations.

```
public String getRevokeQuorum()
```

setRevokeQuorum

Sets the quorum condition for revoke operations.

```
public void setRevokeQuorum(String revokeQuorum)
```

getRevokeRequestDef

Returns the provisioning request definition for revoke operations.

```
public String getRevokeRequestDef()
```

setRevokeRequestDef

Sets the provisioning request definition for revoke operations.

```
public void setRevokeRequestDef(String revokeRequestDef)
```

getRevokeApprovers

Returns the list of approvers for revoke operations.

```
public Approver[] getRevokeApprovers()
```

setRevokeApprovers

Sets the list of approvers for revoke operations.

```
public void setRevokeApprovers(Approver[] revokeApprovers)
```

getOwners

Returns the list of owners for the resource.

```
public DNString[] getOwners()
```

setOwners

Sets the list of owners for the resource.

```
public void setOwners(DNString[] owners)
```

getParameters

Returns the list of entitlement parameters defined for the resource.

```
public ResourceParameter[] getParameters()
```

setParameters

Sets the list of entitlement parameters for the resource.

```
public void setParameters(ResourceParameter[] parameters)
```

getActive

Returns a boolean flag indicating whether the resource is still active, or has been approved or denied.

```
public boolean getActive()
```

setActive

Sets the boolean flag indicating whether the resource is still active.

```
public void setActive(final boolean active)
```

getAllowOverride

Returns a boolean flag indicating whether the approval process for the resource can be overridden by the approval process for a role.

```
public boolean getAllowOverride()
```

setAllowOverride

Sets the boolean flag indicating whether the approval process for the resource can be overridden by the approval process for a role.

```
public void setAllowOverride(final boolean allowOverride)
```

getAllowMulty

Returns a boolean indicating whether the resource allows a user to request multiple resource values.

```
public boolean getAllowMulty()
```

setAllowMulty

Sets the boolean indicating whether the resource allows a user to request multiple resource values.

```
public void setAllowMulty(final boolean allowedMulty)
```

16.2.7 ResourceAssignment

Supporting class that holds resource assignment information.

setResourceDn

Sets the DN for the resource.

```
public void setResourceDn(String resourceDn)
```

getResourceDn

Returns the DN for the resource.

```
public String getResourceDn()
```

setRequesterDn

Sets the DN for the requester.

```
public void setRequesterDn(String requesterDn)
```

getRequesterDn

Returns the DN for the requester.

```
public String getRequesterDn()
```

getRecipientDn

Returns the DN for the recipient of the assignment.

```
public String getRecipientDn()
```

setRecipientDn

Sets the DN for the recipient of the assignment.

```
public void setRecipientDn(String recipientDn)
```

getReason

Returns the reason for the assignment.

```
public String getReason()
```

setReason

Sets the reason for the assignment.

```
public void setReason(String reason)
```

getRequestDate

Returns the date of the assignment request.

```
public Date getRequestDate()
```

setRequestDate

Sets the date of the assignment request.

```
public void setRequestDate(Date requestDate)
```

setRequestParams

Sets the parameters for the request.

```
public void setRequestParams(ResourceRequestParam[] params)
```

getRequestParams

Returns the parameters for the request.

```
public ResourceRequestParam[] getRequestParams()
```

setInstanceGuid

Sets the instanceGuid for the resource assignment.

```
public void setInstanceGuid(String instanceGuid)
```

getInstanceGuid

Returns the instanceGuid for the resource assignment.

```
public String getInstanceGuid()
```

16.2.8 ResourceRequestParam

Supporting class that holds the name and value for a resource request parameter value.

ResourceRequestParam Constructors

The ResourceRequestParam class has two constructors.

Syntax 1: Here is the syntax for a constructor that takes no parameters:

```
public ResourceRequestParam()  
{  
}
```

Syntax 2: Here is the syntax for a constructor that takes two String parameters:

```
public ResourceRequestParam(String name, String value)  
{  
    m_name = name;  
    m_value = value;  
}
```

setName

Sets a parameter name.

Syntax: Here is the method signature:

```
public void setName(String name)
```

getName

Returns a parameter name.

Syntax: Here is the method signature:

```
public String getName()
```

setValue

Sets the value of a parameter.

Syntax: Here is the method signature:

```
public void setValue(String value)
```

getValue

Returns the value of a parameter.

Syntax: Here is the method signature:

```
public String getValue()
```

16.2.9 ResourceAssignmentRequestStatus

Supporting class that holds a resource request status item. The interface includes methods for getting and setting various request status properties. However, you will not need to call the methods for setting property values, since you are using this class to retrieve information about the request status. After calling the `requestResourceGrant()` or the `requestResourceRevoke()` methods, you can use the get methods to get the properties for each status object returned in the `ResourceAssignmentRequestStatus` array.

setEntityKey

Sets the entity key.

Syntax: Here is the method signature:

```
public void setEntityKey(String entityKey)
```

getEntityKey

Gets the entity key.

Syntax: Here is the method signature:

```
public String getEntityKey()
```

setReason

Sets the reason for the role assignment.

Syntax: Here is the method signature:

```
public void setReason(String reason)
```

getReason

Gets the reason for the role assignment.

Syntax: Here is the method signature:

```
public String getReason()
```

setStatusValue

Sets the status value for the request.

Syntax: Here is the method signature:

```
public void setStatusValue(int value)
```

setStatusDescription

Sets the status description for the request.

Syntax: Here is the method signature:

```
public void setStatusDescription(String description)
```

getStatusValue

Gets the status value for the request.

Syntax: Here is the method signature:

```
public int getStatusValue()
```

getStatusDescription

Gets the localized description for the request.

Syntax: Here is the method signature:

```
public String getStatusDescription()
```

setCorrelationId

Sets the correlation ID.

Syntax: Here is the method signature:

```
public void setCorrelationId(String correlationId)
```

getCorrelationId

Gets the correlation ID.

Syntax: Here is the method signature:

```
public String getCorrelationId()
```


setRequester

Sets the requester DN.

Syntax: Here is the method signature:

```
public void setRequester(String requester)
```

getRequester

Gets the requester DN.

Syntax: Here is the method signature:

```
public String getRequester()
```

setRequestDate

Sets the request date.

Syntax: Here is the method signature:

```
public void setRequestDate(Date requestDate)
```

getRequestDate

Gets the request date.

Syntax: Here is the method signature:

```
public Date getRequestDate()
```

setSource

Sets the source resource DN.

Syntax: Here is the method signature:

```
public void setSource(String source)
```

getSource

Gets the source resource DN.

Syntax: Here is the method signature:

```
public String getSource()
```

setTarget

Sets the DN for the target identity.

Syntax: Here is the method signature:

```
public void setTarget(String target)
```

getTarget

Gets the DN for the target identity.

Syntax: Here is the method signature:

```
public String getTarget()
```

setRequestParams

Sets the dynamic request parameters.

Syntax: Here is the method signature:

```
public void setRequestParams(ResourceRequestParam[] params)
```

getRequestParams

Gets the dynamic request parameters.

Syntax: Here is the method signature:

```
public ResourceRequestParam[] getRequestParams()
```

16.3 Resource Web Service Examples

This section provides examples of using the Resource Web Service.

16.3.1 Code Map Synchronization Code Samples

This section provides code samples for using the SOAP endpoints for code map synchronization.

```
public IRemoteResource stub;
stub=getResourcesStub(url,adminname,password);

//refreshCodeMap
EntitlementRefreshInfo refreshResult =
stub.refreshCodeMap("cn=Devices,cn=DevicesLoopback,cn=driverset1,o=system");
System.out.println(refreshResult .getDetailedStatus());
System.out.println(refreshResult .getEntitlementDN());
System.out.println(refreshResult .getGuid());
System.out.println(refreshResult .getStatus());

//getRefreshStatus
CodeMapRefreshStatus[] refreshStatus
=stub.getRefreshStatus("cn=Devices,cn=DevicesLoopback,cn=driverset1,o=system");
for (CodeMapRefreshStatus item : refreshStatus) {
    System.out.println("Connection Name is: " + item.getConnectionName());
    System.out.println("Entitlement DN is: " + item.getEntitlementDN());
    System.out.println("Entitlement GUID is: " + item.getGuid());
    System.out.println("Last Refresh of this Entitlement is: " +
item.getLastRefresh());
    System.out.println("Status is: " + item.getStatus());
}
```

```

//checkCodeMapValueStatus
String connectionName="SAP123";
CodeMapValueStatus checkStatus =
String codeMapValue=null;

stub.checkCodeMapValueStatus("cn=Devices,cn=DevicesLoopback,cn=driverset1,o=system
",connectionName, codeMapValue);

    System.out.println("Connection Name is: " +
checkStatus.getRefreshStatus().getConnectionName());
        System.out.println("Entitlement DN is: " +
checkStatus.getRefreshStatus().getEntitlementDN());
        System.out.println("Entitlement GUID is: " +
checkStatus.getRefreshStatus().getGuid());
        System.out.println("Last Refresh of this Entitlement is: " +
checkStatus.getRefreshStatus().getLastRefresh());
        System.out.println("Status is: " +
checkStatus.getRefreshStatus().getStatus());

        System.out.println(checkStatus.getUpToDate());
        System.out.println(checkStatus.getValue());

private static IRemoteResource getResourcesStub(String url,
String username, String password) throws ServiceException {
    Stub stub = null;

    ResourceService service = new ResourceServiceImpl();
    stub = (Stub) service.getIRemoteResourcePort();
    stub._setProperty(Stub.USERNAME_PROPERTY, username);
    stub._setProperty(Stub.PASSWORD_PROPERTY, password);

    stub._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY,url +"/resource/
service");
    stub._setProperty(Stub.SESSION_MAINTAIN_PROPERTY, Boolean.TRUE);

    return (IRemoteResource) stub;
}

```

17 Forgot Password Web Service

This section describes the Forgot Password Web Service, which allows SOAP clients to invoke a subset of the actions available through the Password Management system.

17.1 About the Forgot Password Web Service

The Forgot Password Web Service exposes a small set of actions from the Password Management system. The service allows remote clients to retrieve information about the forgot password configuration. In addition, it allows clients to retrieve information about the forgot password settings for a particular user, and perform challenge response and change password operations for a user.

The Forgot Password Web Service does not support the full range of password self-service operations. The Forgot Password Web Service is only for forgot password operations. If you want to create a custom user interface for performing password self service functions, such as answering or updating the user's hint or answer, or updating the challenge response questions, or checking on the password policy status, you need to use the REST endpoints that have been added to RBPM.

Calls to the Forgot Password Web Service require HTTP authentication.

17.1.1 Accessing the Service

You can access the Forgot Password Web Service endpoint using a URL similar to the following:

```
http://server:port/warcontext/pwdmgt/service
```

For example, if your server is named "myserver", your User Application is listening on port 8080, and your User Application war file is named "IDMPROV", the URL would be:

```
http://myserver:8080/IDMPROV/pwdmgt/service
```

NOTE: The URL for the Forgot Password Web Service can be changed on the **Forgot Password Settings** page on the Administration tab in the User Application. To change the URL, enter the new URL in the **Forgot Password Web Service URL** field at the bottom of the page.

17.1.2 Accessing the WSDL

You can access the WSDL for the Forgot Password Web Service using a URL similar to the following:

```
http://server:port/warcontext/pwdmgt/service?wsdl
```

For example, if your server is named "myserver", your User Application is listening on port 8080, and your User Application war file is named "IDMPROV", the URL would be:

```
http://myserver:8080/IDMPROV/pwdmgt/service?wsdl
```

17.1.3 Generating the Stub Classes

Before using the Web Service, you need to use the WSSDK tool or another SOAP tool kit to generate the stub classes. To allow your code to find the stub classes, you also need to add the JAR that contains the stub classes to your classpath.

If you want to use the NetIQ WSSDK tool, you can generate the client stubs by extracting the WSDL and running the `wsdl2java` utility. For example, you could run this command to generate the stubs in a package called `com.novell.soa.af.pwdmgt.soap.impl`:

```
"C:\Program Files\Java\jdk1.6.0_31\bin\java" -cp "../lib/wssdk.jar;../lib/jaxrpc-api.jar;../lib/mail.jar;../lib/activation.jar";c:\Program Files\Java\jdk1.6.0_31\lib\tools.jar"; com.novell.soa.ws.impl.tools.wsdl2java.Main -verbose -ds gensrc -d C:\ -noskel -notie -genclient -keep -package com.novell.soa.af.pwdmgt.soap.impl -javadoc pwdmgt.wsdl
```

You can change the `wsdl2java` parameters to suit your requirements.

17.2 Password Management Web Service Interface

This section provides reference information for each forgot password operation available through the Password Management interface.

17.2.1 processForgotConf

Gets the forgot password configuration parameters.

This method returns an object of type `ForgotPasswordConfWSBean`. This object contains the following information about the configuration:

Table 17-1 ForgotPasswordConfWSBean Data

Field	Description
Configured Return Link	Provides the forgot password return link.
Show Return Link	Indicates whether to show the forgot password return link.

Syntax: Here is the method signature:

```
public ForgotPasswordConfWSBean processForgotConf()  
    throws RemoteException;
```

17.2.2 processUser

Retrieves forgot password configuration information for a user.

This method returns an object of type `ForgotPasswordWSBean`. If no match is found for the the user name specified, an error message is returned in the `getUsers()` method of `ForgotPasswordWSBean`. If multiple matches are found, the `getUsers()` method is returned with a `String` array of users. If a single match is found, the `getUsers()` method has a length of 1, and the following methods in `ForgotPasswordWSBean` are set:

- ◆ `getConfiguredRtnLink()`

- ◆ getShowReturnLink()
- ◆ getShowHint()
- ◆ getHint()
- ◆ getShowFullDN()
- ◆ getUserDisplayDN()
- ◆ getUserDN()
- ◆ getUser()
- ◆ getMessage()
- ◆ getAction()
- ◆ getChallengeQuestions()
- ◆ getChaResInUser()
- ◆ getMessage()

When a single user match is found, the user should be presented with the Challenge Response screen. If `getChaResInUse()` returns false, then call `processChaRes()` and show the Forgot Success screen directly without presenting the Challenge Response screen.

Syntax: Here is the method signature:

```
public ForgotPasswordWSBean processUser(final String userName)
    throws RemoteException;
```

The parameters are described below:

- ◆ *userName* specifies the name of a user.

17.2.3 processChaRes

Processes one or more challenge response answers for a particular user.

If the challenge response operation is authenticated, the following events may occur:

- ◆ If the password policy action is EmailHint, the operation will send an email with the hint to the user, and set the message to indicate that the operation succeeded. Therefore, the caller of this method should go to the Forgot Password Change Success screen, and display the message.
- ◆ If the password policy action is ShowHint, the operation will set the message to the user's hint. Therefore, the caller of this method should go to the Forgot Password Change Success screen, and display the message with the hint on the page.
- ◆ If the password policy action is EmailPassword, the operation will set send the password to the user. Therefore, the caller of this method should go to the Forgot Password Change Success screen, and display the message.
- ◆ If the password policy action is ChangePassword, the operation will set the password rules and the password hint. Therefore, the caller of this method should go to the Forgot Password Change screen.

This method returns an object of type `ForgotPasswordWSBean`. After the `processCharRes` operation is called, the following methods are populated with values:

- ◆ getTimeout()
- ◆ getRules()
- ◆ getLocked()

- ♦ `getError()`
- ♦ `getMessage()`

If the `getAction()` method returned by the `processUser()` operation is `ChangePassword`, then present the user with the Password Change screen. Otherwise, go to the Forgot Success screen and present the user with the message returned from the `getMessage()` method.

Syntax: Here is the method signature:

```
public ForgotPasswordWSBean processChaRes(final String userDN, final String[]
chaAnswers) throws RemoteException;
```

The parameters are described below:

- ♦ *userDN* specifies the DN for a particular user.
- ♦ *chaAnswers* provides an array of challenge response answers. The answers are processed in the order in which they are presented.

17.2.4 processChgPwd

Resets the password for a particular user.

After the `processChgPwd` operation is called, the following events may occur:

- ♦ If the change password operation succeeds, the caller of this method should go to the Forgot Password Success screen, and display the success message.
- ♦ If the change password operation fails, the error field on the `ForgotPasswordWSBean` object is set to true, and the message field is populated with the corresponding error message. Therefore, the caller of this method should stay on the password screen and display the error message.

This method returns an object of type `ForgotPasswordWSBean`. After the `processChgPwd` operation is called, the following methods are populated with values:

- ♦ `getTimeout()`
- ♦ `getError()`

If the `getError()` method returns false, you need to present the user with the Password Change Success screen.

Syntax: Here is the method signature:

```
public ForgotPasswordWSBean processChgPwd(final String userDN, final String
newPassword, final String confirmPassword )
throws RemoteException;
```

The parameters are described below:

- ♦ *userDN* specifies the DN for a particular user.
- ♦ *newPassword* supplies a password for the user.
- ♦ *confirmPassword* repeats the password for confirmation.

17.3 ForgotPasswordWSBean

Here is the complete structure of the `ForgotPasswordWSBean` object:

Table 17-2 *ForgotPasswordWSBean Structure*

Field	Description
Users	Provides a list of the users that match the search criteria specified. When the wildcard feature is enabled, multiple matches may be found.
Challenge Questions	Supplies the challenge questions associated with the user.
Configured Return Link	Shows the Return link to be used after the user performs a forgot password operation.
Show Return Link	Indicates whether to show the Return link after the user performs a forgot password operation.
Show Hint	Indicates whether to show the user's password hint on the Forgot Password Change screen.
Show Full DN	Indicates whether to show the user's full DN or just the CN name after the user performs a forgot password operation.
User DN	Shows the user's DN.
User Display DN	Shows the user's display DN. For example, <code>cn=ablake,ou=users,o=netiq</code> or <code>workforceID=ablake,ou=users,o=netiq</code> .
User	Provides the user's display name.
Error	Returns true if an error occurs.
Message	Returns a message in the event that there is an application-specific error.
Action	Specifies the policy action, which is one of the following values: ShowHint, EmailHint, EmailPassword, ChangePassword.
Hint	Specifies the user's password hint.
Rules	Lists the password policy rules.
Is Challenge Response in User	Indicates whether the challenge response feature is enabled for this user. If challenge response in use is false, then the user can only perform the email hint and show hint functions.
Locked	Indicates whether the user account is locked.
Timeout	Indicates whether a session timeout occurred.
Login Attribute	Specifies the user's Login Attribute.

VI REST Services

These sections tell you how to use the REST services.

18 Introduction to Resource Information Services

This section describes the Work Items Service.

18.1 About RIS

This section describes the Resource Information Services (RIS) facility, which is a standalone component that interacts with the Identity Manager User Application. RIS is built on a Resource Oriented Architecture (ROA). The RIS implementation resides in a WAR file called RIS.WAR, where RIS refers to Resource Information Services. The REST resources exposed through RIS make SOAP calls to gather information from various RBPM systems.

The methodology used to define these ROA services is based on the steps described by Leonard Richardson & Sam Ruby in the RESTful Web Services by O'Reilly.

18.1.1 How it Works

The code for RIS is contained in a WAR outside of the User Application. This is a standalone WAR (RIS.war) that uses SOAP calls to extract the necessary work item data.

Language support is determined by the "Accept-Language" header parameter.

The media type is determined by the "Accept" header parameter and must be equal to "application/json".

The implementation does not support the use of extensions. It does not support the ability to enter a language or media extension at the end of a URI.

This implementation is based on the JSR-311 specification implemented by Sun's Jersey product.

You may see this error on the console:

```
09:52:52,684 ERROR [STDERR] Sep 30, 2008 9:52:52 AM
com.sun.jersey.api.core.ClasspathResourceConfig init
INFO: Root resource classes found:
    class com.novell.ris.spi.impl.Root
```

This is a Jersey message that is simply informational. The application should function normally. You can ignore the message.

NOTE: During the deployment of the RIS.war on WebSphere, you might see the following error message in the server log:

```
WebApp      W      Error while adding servlet mapping --> /* Please set
fileServingEnabled=false in the ibm-web-ext.xmi file which is under WEB-INF
folder.
```

This message does not affect the functions of RIS.

Caching the SOAP stubs The web.xml file in the RIS.war includes an element that allows you to control the size of the stub connection pool.

```
<init-param>
  <param-name>STUB_CONNECTION_POOL</param-name>
  <param-value>100</param-value>
</init-param>
```

The STUB_CONNECTION_POOL element defines the size of a pool for caching the SOAP stubs created by each user. The cache uses a Least Recently Used (LRU) eviction policy and defaults to a size of 10 if the element is not defined in web.xml.

Removing the administrator credential restrictions By default, the requirement for invoking the REST and SOAP services is that the HTTP session logged in user must have administrator credentials. The Provisioning and Directory Web Services require Provisioning Administrator credentials. The Roles Web Service requires Role Administrator credentials. The restrictions can be removed to allow a session with a logged in user who does not have administrator credentials to invoke the methods for the services by changing the configuration settings for the service. In order to do this, you must extract the configuration files from the User Application war, make the appropriate changes, and import the files back into the User Application WAR. The details for removing the restrictions is included with the documentation for each of the underlying SOAP services. For example, to remove the credential restriction for the Role Service, see [Section 15.1.4, “Removing Administrator Credential Restrictions,” on page 360](#).

Media Type Supported

The only media type supported is JSON (application/json). The service uses a JSON Array format for list of items and a single JSON object for detail information. The media type is determined by the “Accept” header parameter. The implementation uses the Jettison JSON APIs to create the JSON structures.

Digital Signatures Not Supported

The REST interfaces do not support digital signatures. If you attempt to process a digital signature workflow through REST, an internal server error message will appear.

API Version Optional in URIs

The REST URIs work with or without the API version. For example, to access the roles service, you could specify either of the following URIs:

```
/RIS/v1/roles
/RIS/roles
```

18.1.2 Configuring the RIS WAR

This section provides manual instructions for setting up the RIS WAR on JBoss and WebSphere. In this release, the RIS WAR is configured automatically, so these steps are not required in most environments.

JBoss Configuration

To configure the RIS WAR on JBoss:

- 1 Modify the host, port, and WAR context information for the RBPM deployment on JBoss in the web.xml of the RIS WAR.

- 1a Copy the RIS.war file to a test folder.

For example: /home/lab/RIS

- 1b Extract the web.xml from the RIS war, maintaining the folder structure.

This will create the following structure: /home/lab/RIS/WEB-INF/web.xml

- 1b1 Open the web.xml in a text editor.

- 1b2 Locate the following entry:

```
<init-param>
    <param-name>USER_APP_URL</param-name>
    <param-value>http://localhost:8080/IDMProv</param-value>
</init-param>
```

- 1b3 Modify the param-value as necessary. You need to use either the DNS name or the IP address of the server on which the RBPM war is deployed.

IMPORTANT: Do not use localhost if you plan to use the REST identity services to access user photos. The photo URL is dependent on this entry. The photo URL must point to the User Application to retrieve the photo. The REST identity service does not provide the binaries for the photo, but does provide a link, which is based on this entry.

- 1b4 Save the file.

- 1c Add the web.xml file back to the RIS war using the jar command from the SUN JDK.

For example: /home/lab/jdk1.6.0_11/bin/jar -uf RIS.war WEB-INF/web.xml

- 2 Copy the RIS war to the deployment directory of the JBoss server.
- 3 Extract the commons-codec.jar from the RBPM war into the %context%/lib directory of the JBoss server where the RIS WAR will be deployed. Make sure to not maintain folder structure when extracting the file.

For example: /home/lab/IDM370/idm/jboss/server/IDMProv/lib

- 4 Start JBoss.

WebSphere Configuration

To configure the RIS WAR on WebSphere:

- 1 Modify the host, port, and war context information for the RBPM deployment on WebSphere in the web.xml of the RIS WAR.

- 1a Copy the RIS war to a test folder.

For example: /home/lab/RIS

- 1b Extract the web.xml from the RIS war, maintaining the folder structure.

This will create the following structure: /home/lab/RIS/WEB-INF/web.xml

- 1b1 Open the web.xml in a text editor.

- 1b2 Locate the following entry:

```
<init-param>
    <param-name>USER_APP_URL</param-name>
    <param-value>http://localhost:8080/IDMProv</param-value>
</init-param>
```

- 1b3** Modify the param-value as necessary. You need to use either the DNS name or the IP address of the server on which the RBPM war is deployed.

IMPORTANT: Do not use localhost if you plan to use the REST identity services to access user photos. The photo URL is dependent on this entry. The photo URL must point to the User Application to retrieve the photo. The REST identity service does not provide the binaries for the photo, but does provide a link, which is based on this entry.

- 1b4** Save the file.

- 1c** Add the web.xml file back to the RIS war using the jar command from the IBM JDK.

For example: `/home/lab/WAS61/IBM/WebSphere/AppServer/java/bin/jar -uf RIS.war WEB-INF/web.xml`

- 2** Extract the commons-httpclient.jar and log4j.jar from the RBPM war that was created for WebSphere into your test folder/WEB-INF/lib directory.

For example: `/home/lab/RIS/WEB-INF/lib`

- 3** Add the two jars to the RIS war using the jar command from the IBM JDK.

For example:

```
/home/lab/WAS61/IBM/WebSphere/AppServer/java/bin/jar -uf RIS.war WEB-INF/lib/commons-httpclient.jar
/home/lab/WAS61/IBM/WebSphere/AppServer/java/bin/jar -uf RIS.war WEB-INF/lib/log4j.jar
```

- 4** Deploy the RIS WAR to WebSphere.

NOTE: You may see a warning message in the WebSphere logs when deploying the RIS.war. However, the deployment should complete successfully.

Error while adding servlet mapping --> /* Please set fileServingEnabled=false in the ibm-web-ext.xmi file which is under WEB-INF folder.

18.2 Security

This section describes the security model used for the REST services.

The security model attempts to satisfy these objectives:

- ◆ Protects against CSRF attacks
- ◆ Allows the client to pass in user credentials
- ◆ Uses an HTTP Authorization or an HTTP Session Secret header for the REST requests

18.2.1 Architecture

The security model supports two options for making requests. The first one (Option 1) consists of passing in the user credentials in an HTTP header (default: RESTAuthorization). The second approach (Option 2) consists of a two request approach. In the second option, an authorization request is required first and all subsequent requests pass in the session secret token in an HTTP header. The header defaults to RESTSessionSecret.

Both options require the passing of sensitive data on the wire. Therefore, NetIQ highly recommends that you run this application in a TLS/SSL environment (HTTPS). Otherwise the user credentials could be exposed to a man-in-the-middle attack.

Either approach will work. However, NetIQ recommends using Option 2 (the Session Secret Security Model approach) rather than Option 1 (the Authorization Security Model approach). Option 2 offers more protection against discovering the actual user credentials. The credentials are maintained by the RIS server and are discovered using a unique access token through the RESTSessionSecret HTTP header.

Option 1: Authorization Security Model

This model is the same as the Basic authorization model. This model is recommended for developers who do not use JavaScript for their client application.

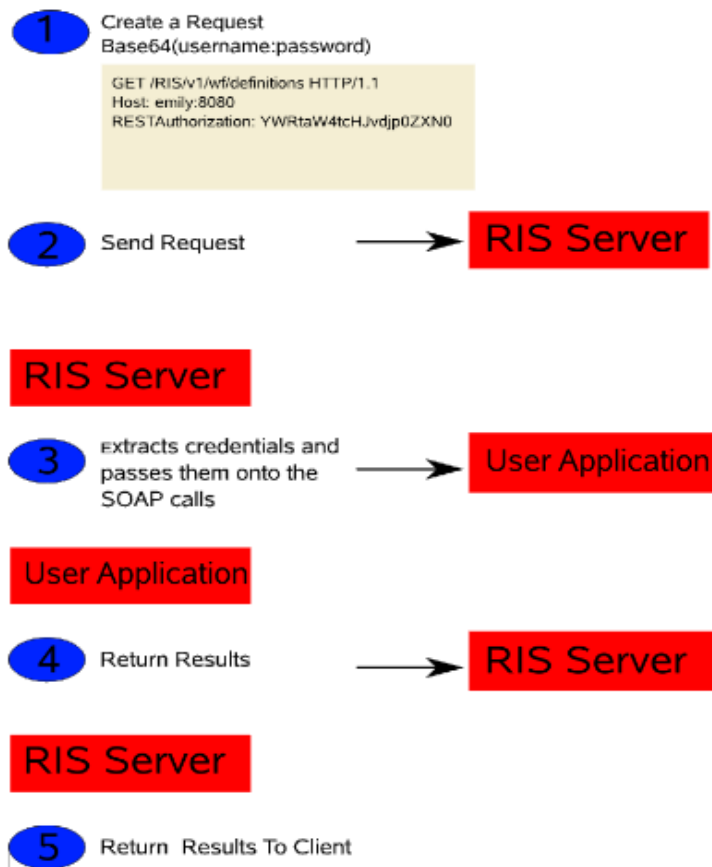
Here is the flow of control used with this option:

1. Developers must include the Base64(username:password)) string in an HTTP header (RESTAuthorization) before making the call to the REST service. The HTTP header name can be configured at installation time. The default name is:

`RESTAuthorization`
2. The client application sends the request to the RIS server.
3. The RIS server extracts the credentials from the header and passes those credentials onto the SOAP service. The actual authentication check is performed at the User Application server.
4. The User Application SOAP call is either granted or denied and the result is returned to the RIS server.
5. The RIS server returns the result to the client application.

The following picture illustrates the flow:

Figure 18-1 Option 1 Authorization Flow



Example 18-1 Example

Suppose you issue the following REST call:

```
/v1/wf/definitions
```

Here is the request:

```
GET /RIS/v1/wf/definitions HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.7) Gecko/2009022800
SUSE/3.0.7-1.4 Firefox/3.0.7
Accept: application/json
Accept-Language: en,it;q=0.8,fr;q=0.6,de;q=0.4,en-us;q=0.2
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
RESTAuthorization: YWRtaW4tcHJvdjp0ZXN0
```

Here is the response:

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1
Content-Type: application/json
Transfer-Encoding: chunked
Date: Tue, 31 Mar 2009 13:48:05 GMT

7d2

```
[
  {
    "Links": [
      {
        "Link": "/RIS/v1/wf/
processes?filter=Definition=cn=Change+Title+Single+Approval,cn=RequestDefs,cn=AppC
onfig,cn=PicassoDriver,cn=TestDrivers,o=netiq",
        "Type": "wf/processes",
        "Value": "Workflow Processes"
      },
      {
        "Link": "/RIS/v1/wf/
workitems?filter=Definition=cn=Change+Title+Single+Approval,cn=RequestDefs,cn=AppC
onfig,cn=PicassoDriver,cn=TestDrivers,o=netiq",
        "Type": "wf/workitems",
        "Value": "Workflow Workitems"
      }
    ],
    "DataItems": [],
    "DN": "cn=Change Title Single
Approval,cn=RequestDefs,cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=netiq",
    "GUID": "5a4f7af2142189430d935a4f7af21421",
    "Link": "/RIS/v1/wf/definitions/5a4f7af2142189430d935a4f7af21421",
    "Value": "Change Title Single Approval",
    "Category": "accounts",
    "DigitalSignatureType": "not-required",
    "Description": "Change Title",
    "Operation": "0",
    "Recipient": ""
  },
  {
    "Links": [
      {
        "Link": "/RIS/v1/wf/
processes?filter=Definition=cn=Change+Title,cn=RequestDefs,cn=AppConfig,cn=Picasso
Driver,cn=TestDrivers,o=netiq",
        "Type": "wf/processes",
        "Value": "Workflow Processes"
      },
      {
        "Link": "/RIS/v1/wf/
workitems?filter=Definition=cn=Change+Title,cn=RequestDefs,cn=AppConfig,cn=Picasso
Driver,cn=TestDrivers,o=netiq",
        "Type": "wf/workitems",
        "Value": "Workflow Workitems"
      }
    ]
  }
]
```

```

    ],
    "DataItems": [],
    "DN": "cn=Change
Title,cn=RequestDefs,cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=netiq",
    "GUID": "71e22c1cf4b4e74fdb871e22c1cf4b4",
    "Link": "/RIS/v1/wf/definitions/71e22c1cf4b4e74fdb871e22c1cf4b4",
    "Value": "Change Title",
    "Category": "accounts",
    "DigitalSignatureType": "not-required",
    "Description": "Change Title",
    "Operation": "0",
    "Recipient": ""
  }
]

```

Option 2: Session Secret Security Model

The session secret security model allows a developer to ask for an authorization unique id. This session secret ID is then used on all subsequent calls. This is to allow client developers to be more secured than passing user credentials for every call.

Here is the flow of control used with this option:

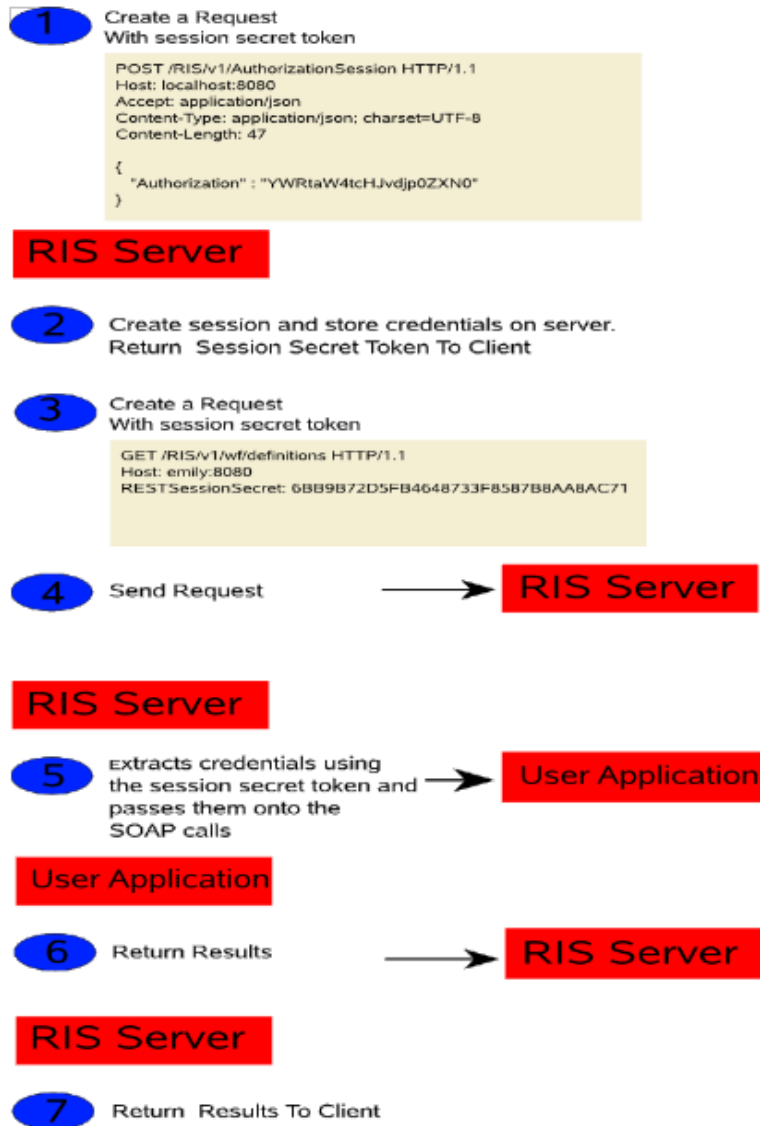
1. Developers must first make a call to the /v1/AuthorizationSession REST service to obtain a session token. This is a POST call with the credentials (Base64(username:password)) in the content of the message in a JSON object (see section 4).
2. The RIS server will then create a session object and stores the credentials in memory for the duration of the session. The session secret token is returned to the client application via a JSON object.
3. Developers must include the Session Secret token string in an HTTP header (RESTSessionSecret) before making any subsequent REST service calls. The HTTP header name can be configured at installation time. The default name is:

```
RESTSessionSecret
```

4. The client application sends the request to the RIS server.
5. The RIS server extracts the session secret token from the HTTP header and retrieves the credentials from memory based on the token for that session. The credentials are passed onto the SOAP service. The actual authentication check is performed at the User Application server.
6. The User Application SOAP call is either granted or denied and the result is returned to the RIS server.
7. The RIS server returns the result to the client application.

The following picture illustrates the flow:

Figure 18-2 Option 2 Authorization Flow



Example 18-2 Example

First, you issue the following call to the Authorization REST Service:

/v1/AuthorizationSession

Here is the request:

```
POST /RIS/v1/AuthorizationSession HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.7) Gecko/2009022800
SUSE/3.0.7-1.4 Firefox/3.0.7
Accept: application/json
Accept-Language: en,it;q=0.8,fr;q=0.6,de;q=0.4,en-us;q=0.2
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: application/json; charset=UTF-8
Content-Length: 47
Pragma: no-cache
Cache-Control: no-cache
```

```
{
  "Authorization" : "YWRtaW4tcHJvdjpwZXN0"
}
```

Here is the response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1
Set-Cookie: JSESSIONID=17B5528DEEC66610D0FBB456992E10ED; Path=/RIS
Content-Type: application/json
Transfer-Encoding: chunked
Date: Tue, 31 Mar 2009 13:54:26 GMT
```

```
35
{"SessionSecret": "17B5528DEEC66610D0FBB456992E10ED"}
```

Next, you issue the REST call:

```
/v1/wf/definitions
```

Here is the request:

```
GET /RIS/v1/wf/definitions HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.7) Gecko/2009022800
SUSE/3.0.7-1.4 Firefox/3.0.7
Accept: application/json
Accept-Language: en,it;q=0.8,fr;q=0.6,de;q=0.4,en-us;q=0.2
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
RESTSessionSecret: 17B5528DEEC66610D0FBB456992E10ED
```

Here is the response:

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1
Content-Type: application/json
Transfer-Encoding: chunked
Date: Tue, 31 Mar 2009 13:48:05 GMT

7d2

```
[
  {
    "Links": [
      {
        "Link": "/RIS/v1/wf/
processes?filter=Definition=cn=Change+Title+Single+Approval,cn=RequestDefs,cn=AppC
onfig,cn=PicassoDriver,cn=TestDrivers,o=netiq",
        "Type": "wf/processes",
        "Value": "Workflow Processes"
      },
      {
        "Link": "/RIS/v1/wf/
workitems?filter=Definition=cn=Change+Title+Single+Approval,cn=RequestDefs,cn=AppC
onfig,cn=PicassoDriver,cn=TestDrivers,o=netiq",
        "Type": "wf/workitems",
        "Value": "Workflow Workitems"
      }
    ],
    "DataItems": [],
    "DN": "cn=Change Title Single
Approval,cn=RequestDefs,cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=netiq",
    "GUID": "5a4f7af2142189430d935a4f7af21421",
    "Link": "/RIS/v1/wf/definitions/5a4f7af2142189430d935a4f7af21421",
    "Value": "Change Title Single Approval",
    "Category": "accounts",
    "DigitalSignatureType": "not-required",
    "Description": "Change Title",
    "Operation": "0",
    "Recipient": ""
  },
  {
    "Links": [
      {
        "Link": "/RIS/v1/wf/
processes?filter=Definition=cn=Change+Title,cn=RequestDefs,cn=AppConfig,cn=Picasso
Driver,cn=TestDrivers,o=netiq",
        "Type": "wf/processes",
        "Value": "Workflow Processes"
      },
      {
        "Link": "/RIS/v1/wf/
workitems?filter=Definition=cn=Change+Title,cn=RequestDefs,cn=AppConfig,cn=Picasso
Driver,cn=TestDrivers,o=netiq",
        "Type": "wf/workitems",
        "Value": "Workflow Workitems"
      }
    ]
  }
]
```

```

    ],
    "DataItems": [],
    "DN": "cn=Change
Title,cn=RequestDefs,cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=netiq",
    "GUID": "71e22c1cf4b4e74fbd871e22c1cf4b4",
    "Link": "/RIS/v1/wf/definitions/71e22c1cf4b4e74fbd871e22c1cf4b4",
    "Value": "Change Title",
    "Category": "accounts",
    "DigitalSignatureType": "not-required",
    "Description": "Change Title",
    "Operation": "0",
    "Recipient": ""
  }
]

```

18.2.2 Authorization REST Service

The Authorization REST Service lets you obtain a session token. When you make a call to the service, the RIS server creates a session object and stores the credentials in memory for the duration of the session. The session secret token is returned to the client application via a JSON object. The only media type supported is application/json, which uses a JSON Array format for the list of items and a single JSON object for detailed information.

The following table shows the complete URI syntax for all resource end points associated with the Authorization REST Service, along with a description for each URI and a list of supported HTTP methods:

Table 18-1 URI Syntax for the Authorization REST Service

URI	Description
/v1/AuthorizationSession	<p>Creates a new session authorization session and obtains a session token. The following HTTP methods are supported:</p> <p>GET – Not supported</p> <p>POST – Creates a new authorization session object and returns the session secret token in the response.</p> <p>Request JSON Object:</p> <pre>{ "Authorization" : Base64(username:password) }</pre> <p>Response JSON Object:</p> <pre>{ "SessionSecret": "session secret token" }</pre> <p>PUT – Not supported</p> <p>DELETE – Not supported</p>

URI	Description
/v1/AuthorizationSession/{session secret token}	<p>Deletes and invalidates the authorization session object. The following HTTP methods are supported:</p> <p>GET – Not supported</p> <p>PUT – Not supported</p> <p>POST – Not supported</p> <p>DELETE – Deletes and invalidates the authorization session object.</p> <p>POST with Matrix parameter DELETE – Same as DELETE because of limitations in browsers to set the DELETE method</p>

The following matrix parameters are available for debugging and displaying the schema:

Table 18-2 Matrix Parameters for Debugging and Displaying the Schema

URI	Description
/v1/AuthorizationSession;debug	This <code>debug</code> matrix parameter displays the JSON structure of the content type in human readable format, as opposed to compressed format. This matrix parameter can be put anywhere in the URI.
/v1/AuthorizationSession;schema	The <code>schema</code> matrix parameter can be put anywhere in the URI and in combination with the “Accept” header type will return the appropriate schema document for the content type. In this case a JSON schema document is returned for the workitems.

Example 18-1 Example

Here is an example of a call to the Authorization service that includes the debug and schema parameters:

```
/v1/AuthorizationSession;debug;schema
```

Here is the request:

```
POST /RIS/v1/AuthorizationSession;schema;debug HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.7) Gecko/2009022800
SUSE/3.0.7-1.4 Firefox/3.0.7
Accept: application/json
Accept-Language: en,it;q=0.8,fr;q=0.6,de;q=0.4,en-us;q=0.2
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: application/json; charset=UTF-8
Content-Length: 38
Pragma: no-cache
Cache-Control: no-cache
```

```
{
  "Authorization" : "YWRtaW4tcHJvdjpwOZXXN0"
}
```

Here is the response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1
Content-Type: application/json
Transfer-Encoding: chunked
Date: Tue, 31 Mar 2009 13:18:45 GMT
```

```
171
{
  "Request": {"SessionInfo": {
    "description": "schema for: /v1/AuthorizationSession",
    "type": "object",
    "properties": {"Authorization": {"type": "string"}}
  }},
  "Response": {"SessionSecret": {
    "description": "schema for: /v1/AuthorizationSession",
    "type": "object",
    "properties": {"SessionSecret": {"type": "string"}}
  }}
}
```

18.2.3 Configuration Parameters

The RIS.war uses the following filter parameters, all of which are set in the WEB.XML file.

Table 18-3 Filter Parameters

Parameter	Description
AUTHORIZATION_HEADER	<p>The AUTHORIZATION_HEADER filter parameter specifies the HTTP header name for option 1 - Authorization Security model. If not supplied, then the default will be:</p> <p>RESTAuthorization</p> <p>This HTTP header will hold the user credentials.</p> <p>Example:</p> <p>RESTAuthorization: Base64(username:password)</p>
SESSION_SECRET_HEADER	<p>The SESSION_SECRET_HEADER filter parameter specifies the HTTP header name to hold the session secret for Option 2 – Session Secret Security Model. If not supplied, then the default will be:</p> <p>RESTSessionSecret</p> <p>This HTTP header will hold the session secret returned from the RIS server when an access token is requested via the REST service:</p> <p>/RIS/v1/AuthenticationSession</p> <p>Example:</p> <p>RESTSessionSecret: <token></p>
USER_APP_URL	<p>The USER_APP_URL filter parameter will point to the User Application associated with the RIS server. All SOAP calls will use this URL for the SOAP end point.</p>
STUB_CONNECTION_POOL	<p>The STUB_CONNECTION_POOL filter parameter holds the number of connection that we want to maintain from the RIS server to the User Application server. This is to make the client perform better.</p>

In addition to these filter parameters, the configuration also supports the following session parameter:

Table 18-4 Session Parameter

Parameter	Description
Session Timeout	<p>This setting is used to control the length of the sessions. It is specified in minutes.</p>

Here is a sample Web.XML that illustrates the use of the configuration parameters:

```

<filter>
  <filter-name>Authorization Filter</filter-name>
  <filter-class>com.novell.ris.common.impl.ServletFilter</filter-class>
  <init-param>
    <param-name>AUTHORIZATION_HEADER</param-name>
    <param-value>RESTAuthorization</param-value>
  </init-param>
  <init-param>
    <param-name>SESSION_SECRET_HEADER</param-name>
    <param-value>RESTSessionSecret</param-value>
  </init-param>
  <init-param>
    <param-name>USER_APP_URL</param-name>
    <param-value>http://localhost:8080/IDMProv</param-value>
  </init-param>
  <!-- If not entered, Stub Connection size will default to 10 stub connections -->
  <init-param>
    <param-name>STUB_CONNECTION_POOL</param-name>
    <param-value>10</param-value>
  </init-param>
</filter>
...
<session-config>
  <session-timeout>30</session-timeout>
</session-config>

```

18.3 WADL Document

To see the Web Application Description Language (WADL) document for the RIS facility, enter the following URI on whatever server the RIS.war has been deployed to.

```
RIS/application.wadl
```

The WADL document shows the available resource paths for the RIS application, as shown below:

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application xmlns="http://research.sun.com/wadl/2006/10">
  <doc xmlns:jersey="http://jersey.dev.java.net/" jersey:generatedBy="Jersey:
1.0.3 04/15/2009 11:52 PM"/>
  <resources base="http://emily:8080/RIS/">
    <resource path="/v1">
      <method name="GET" id="getRootJSON">
        <response>
          <representation mediaType="application/json"/>
        </response>
      </method>
      <resource path="roles">
        <param xmlns:xs="http://www.w3.org/2001/XMLSchema" type="xs:string"
style="template" name="roles"/>
        <method name="GET" id="getListJSON">
          <request>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="query" name="filter"/>
          </request>
          <response>
            <representation mediaType="application/json"/>
          </response>
        </method>
        <resource path="{GUID}">

```

```

        <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
        <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
        <method name="GET" id="getRoleJSON">
            <response>
                <representation mediaType="application/json"/>
            </response>
        </method>
        <resource path="/sods">
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
            <method name="GET" id="getSoDListJSON">
                <response>
<representation mediaType="application/json"/>
                </response>
            </method>
        </resource>
        <resource path="/sods/{SODID}">
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="SODID"/>
            <method name="GET" id="getSoDJSON">
                <response>
<representation mediaType="application/json"/>
                </response>
            </method>
        </resource>
        <resource path="/assignments">
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
            <method name="GET" id="getAssignmentListJSON">
                <response>
<representation mediaType="application/json"/>
                </response>
            </method>
            <method name="PUT" id="postAssignment">
                <request>
<representation mediaType="*/*/">
                </request>
            </method>
        </resource>
        <resource path="/assignments/{ASSIGNMENTID}">
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="ASSIGNMENTID"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
            <method name="GET" id="getAssignmentJSON">
                <response>
<representation mediaType="application/json"/>
                </response>
            </method>
            <method name="DELETE" id="deleteAssignment"/>
        </resource>
    </resource>
</resource>
<resource path="identities">
    <param xmlns:xs="http://www.w3.org/2001/XMLSchema" type="xs:string"
style="template" name="identities"/>

```

```

        <method name="GET" id="getListJSON">
            <request>
                <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
default=" " type="xs:string" style="query" name="filter"/>
            </request>
            <response>
                <representation mediaType="application/json"/>
            </response>
        </method>
        <resource path="{GUID}">
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
            <method name="GET" id="getIdentityJSON">
                <response>
                    <representation mediaType="application/json"/>
                </response>
            </method>
            <resource path="{ATTRIBUTE}">
                <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
                <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="ATTRIBUTE"/>
                <method name="GET" id="getAttributeJSON">
                    <response>
<representation mediaType="application/json"/>
                    </response>
                </method>
                <method name="POST" id="updateAttribute">
                    <request>
<representation mediaType="*/*/>
                    </request>
                </method>
            </resource>
        </resource>
        <resource path="AuthorizationSession">
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" type="xs:string"
style="template" name="AuthorizationSession"/>
            <method name="POST" id="createAuthorizationSession">
                <request>
                    <representation mediaType="application/json"/>
                </request>
                <response>
                    <representation mediaType="application/json"/>
                </response>
            </method>
            <resource path="{GUID}">
                <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
                <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
                <method name="DELETE" id="terminateSession"/>
                <method name="POST" id="postTerminateSession"/>
            </resource>
        </resource>
        <resource path="wf/definitions">
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema" type="xs:string"
style="template" name="wf/definitions"/>

```

```

<method name="GET" id="getListJSON">
  <response>
    <representation mediaType="application/json"/>
  </response>
</method>
<resource path="{GUID}">
  <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
  <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
  <method name="POST" id="startProcess">
    <request>
      <representation mediaType="application/json"/>
    </request>
  </method>
  <method name="GET" id="getDefinitionJSON">
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
</resource>
</resource>
<resource path="wf/processes">
  <param xmlns:xs="http://www.w3.org/2001/XMLSchema" type="xs:string"
style="template" name="wf/processes"/>
  <method name="GET" id="getListJSON">
    <request>
      <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="query" name="filter"/>
    </request>
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
  <resource path="{GUID}">
    <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
    <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
    <method name="GET" id="getProcessJSON">
      <response>
        <representation mediaType="application/json"/>
      </response>
    </method>
    <method name="DELETE" id="terminateProcess"/>
    <method name="POST" id="postItem"/>
  </resource>
</resource>
<resource path="wf/workitems">
  <param xmlns:xs="http://www.w3.org/2001/XMLSchema" type="xs:string"
style="template" name="wf/workitems"/>
  <method name="GET" id="getListJSON">
    <request>
      <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="query" name="filter"/>
    </request>
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>

```

```

        <resource path="{GUID}">
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
            <param xmlns:xs="http://www.w3.org/2001/XMLSchema"
type="xs:string" style="template" name="GUID"/>
            <method name="GET" id="getItemJSON">
                <response>
                    <representation mediaType="application/json"/>
                </response>
            </method>
            <method name="PUT" id="putItemJSON">
                <request>
                    <representation mediaType="application/json"/>
                </request>
            </method>
            <method name="POST" id="postItemJSON">
                <request>
                    <representation mediaType="application/json"/>
                </request>
            </method>
        </resource>
    </resource>
</resources>
</application>

```

19 Identities Service

This section describes the Identities Service.

19.1 About the Identities Service

The Identities Service provides a REST endpoint for retrieving information about identities.

19.2 Accessing and Using the Identities Service

The Identities Service exposes resources to retrieve identity information. The service allows a user to retrieve lists of identities or access specific identities by filter expression or identity ID.

19.2.1 Available Resources

There are several types of resources available for the service.

Identities Service

The service provides a resource URI for every object exposed. The Identities resource supports two basic URI patterns:

- ♦ Resource for returning a collection of identities
- ♦ Resource for returning a specific identity instance

Services for Filtering, Debugging, and Displaying Schema Information

The service supports the following parameters to allow you to perform operations on the primary identities data set:

- ♦ A filter parameter to enable the filtering of result sets
- ♦ A debug matrix parameter to enable you to return the JSON structures in a human readable format
- ♦ A schema matrix parameter to enable you to return the schema for the data set

19.2.2 Complete URI Syntax

The following table shows the complete URI syntax for all resource end points associated with the Identities Service, along with a description for each URI and a list of supported HTTP methods:

Table 19-1 Resource URIs

URI	Description
/v1	Entry point for the service.
/vi/identities	<p>Will return a list of identities with minimum information and a list of identity URI links that include the identity GUID. The VDX services will be used to provide identity information. This information will be based on how the DAL user entity is defined. All attributes defined by the DAL entry will be made available to the identity JSON payload.</p> <p>The following HTTP methods are supported with this URI:</p> <p>GET - This will return a collection of identities (JSON Array).</p> <p>POST – Not supported</p> <p>PUT – Not supported</p> <p>DELETE – Not supported</p>
identities?filter={identity filter}	<p>GET - The Identity ID parameter will be a GUID to identify a specific identity within the LDAP realm. The payload will include links to the identity's roles, resources, and work items. The VDX services will be used to provide identity information. This information will be based on how the DAL user entity is defined. All attributes defined by the DAL entry will be made available to the identity JSON payload.</p> <p>POST – Not supported</p> <p>PUT – Not supported</p> <p>DELETE – Not supported</p>
identities/{Identity Id}	<p>If the “Accept” header is “image/gif”, then the identity photo will be returned.</p> <p>GET - The Identity ID parameter will be a GUID to identify a specific identity within the LDAP realm. The payload will include links to the identity's roles, resources, and work items. The VDX services will be used to provide identity information. This information will be based on how the DAL user entity is defined. All attributes defined by the DAL entry will be made available to the identity JSON payload. In the case of image type data, this might be just a link to this information (still needs to be flushed out)</p> <p>POST – Not supported</p> <p>PUT – Not supported</p> <p>DELETE – Not supported</p> <p>If the “Accept” header is “image/gif”, then the identity photo will be returned.</p>

URI	Description
identities/{Identity Id}/{attribute ID}	<p>GET - A specific attribute for a specific identity ID.</p> <p>POST – Not supported</p> <p>PUT – Not supported</p> <p>DELETE – Not supported</p> <p>Note: application/json will be returned for all attributes except for the attribute types of binary which an “image/gif” content type will be returned.</p>
identities/{GUID}	GET - Retrieves a user by GUID.
identities/{Fully qualified DN}	<p>GET - Retrieves a user by fully qualified DN.</p> <p>Example: /v1/identities/ cn=ablake,ou=users,ou=idmsample,o=netiq</p>
identities/{GUID}/{ATTRIBUTE}	GET - Retrieves a specific attribute for a user by GUID.
identities/{Fully qualified DN}/{ATTRIBUTE}	GET - Retrieves a specific attribute for a user by fully qualified DN.

Table 19-2 Matrix Parameters for Debugging and Displaying the Schema

URI	Description
identities;debug	This <code>debug</code> matrix parameter displays the identities JSON structure in human readable format, as opposed to compressed format. This matrix parameter can be put anywhere in the URI.
identities;schema	The <code>schema</code> matrix parameter can be put anywhere in the URI and in combination with the “Accept” header type will return the appropriate schema document for the content type. In this case a JSON schema document is returned for the workitems.

19.2.3 JSON Representations Received by the Client

This section shows the JSON structures received by the client for each resource. In each case, the HTTP method is GET.

Identities Endpoint

The identities end point (/identities) returns a collection (JSON Array) of identities available to the Directory Abstraction Layer (DAL).

```

[
  {
    "Managers": [{
      "DN": "cn=mmackenzie,ou=users,ou=medical-idmsample,o=netiq",
      "GUID": "1b9d83fa6f03b64e5bba1b9d83fa6f03",
      "Link": "/RIS/v1/identities/1b9d83fa6f03b64e5bba1b9d83fa6f03",
      "Value": "Margo MacKenzie"
    }],
    "DirectReports": [{}],
    "Groups": [{
      "DN": "cn=HR,ou=groups,ou=medical-idmsample,o=netiq",
      "GUID": "7f7f381d9cc3ad4694967f7f381d9cc3",
      "Link": "",
      "Value": "Human Resources"
    }],
    "Links": [
      {
        "Type": "wf/workitems",
        "Value": "Workitems",
        "Link": "/RIS/v1/wf/workitems?filter=Addressee%3Dcn%3Dablake%2Cou%3Dusers%2Cou%3Dmedical-idmsample%2Co%3Dnetiq"
      },
      {
        "Type": "roles",
        "Value": "Roles",
        "Link": "/RIS/v1/roles?filter=User%3Dcn%3Dablake%2Cou%3Dusers%2Cou%3Dmedical-idmsample%2Co%3Dnetiq"
      }
    ],
    "DN": "cn=ablake,ou=users,ou=medical-idmsample,o=netiq",
    "GUID": "26b65d8611075849e2b226b65d861107",
    "Link": "/RIS/v1/identities/26b65d8611075849e2b226b65d861107",
    "FirstName": "Allison",
    "LastName": "Blake",
    "Title": "Payroll",
    "Department": "HR",
    "Location": "Northeast",
    "Email": "mthibault@netiq.com",
    "TelephoneNumber": "(555) 555-1222",
    "City": "",
    "Country": "",
    "PostalCode": "",
    "State": "",
    "Street": ""
  },
  ... More identities
]

```

Identities/{Identity Id} Endpoint

The identities/{Identity Id} endpoint returns specific identity information from the Directory Abstraction Layer. This information will be based on how the DAL user entity is defined. All attributes defined by the DAL entry are made available to the identity JSON payload.

```

{
  "Managers": [{
    "DN": "cn=mmackenzie,ou=users,ou=medical-idmsample,o=netiq",
    "GUID": "1b9d83fa6f03b64e5bba1b9d83fa6f03",
    "Link": "/RIS/v1/identities/1b9d83fa6f03b64e5bba1b9d83fa6f03",
    "Value": "Margo MacKenzie"
  }],
  "DirectReports": [{}],
  "Groups": [{
    "DN": "cn=HR,ou=groups,ou=medical-idmsample,o=netiq",
    "GUID": "7f7f381d9cc3ad4694967f7f381d9cc3",
    "Link": "",
    "Value": "Human Resources"
  }],
  "Links": [
    {
      "Type": "wf/workitems",
      "Value": "Workitems",
      "Link": "/RIS/v1/wf/workitems?filter=Addressee%3Dcn%3Dablake%2Cou%3Dusers%2Cou%3Dmedical-idmsample%2Co%3Dnetiq"
    },
    {
      "Type": "roles",
      "Value": "Roles",
      "Link": "/RIS/v1/roles?filter=User%3Dcn%3Dablake%2Cou%3Dusers%2Cou%3Dmedical-idmsample%2Co%3Dnetiq"
    }
  ],
  "DN": "cn=ablake,ou=users,ou=medical-idmsample,o=netiq",
  "GUID": "26b65d8611075849e2b226b65d861107",
  "Link": "/RIS/v1/identities/26b65d8611075849e2b226b65d861107",
  "FirstName": "Allison",
  "LastName": "Blake",
  "Title": "Payroll",
  "Department": "HR",
  "Location": "Northeast",
  "Email": "mthibault@netiq.com",
  "TelephoneNumber": "(555) 555-1222",
  "City": "",
  "Country": "",
  "PostalCode": "",
  "State": "",
  "Street": ""
}

```

identities/{Identity Id}/{attribute ID} Endpoint

The identities/{Identity Id} endpoint returns a specific identity attribute.

```

{
  "DN": "cn=ablake,ou=users,ou=medical-idmsample,o=netiq",
  "GUID": "26b65d8611075849e2b226b65d861107",
  "Link": "/RIS/v1/identities/26b65d8611075849e2b226b65d861107",
  "FirstName": "Allison"
}

```

19.2.4 Event Status Codes

This section shows the event status codes for the available resources:

Table 19-3 Event Status Codes

URI	Status codes
identities	HTTP GET - Status 200 - OK
identities?filter={identity filter}	HTTP GET - Status 200 - OK
identities/{Identity Id}	HTTP GET - Status 200 - OK
identities/{Identity Id}/{attribute ID}	HTTP GET - Status 200 - OK

The Jersey implemented error conditions are used. When server errors are found, Jersey returns the appropriate 400 and 500 level codes.

20 Password Management Services

This section describes the Password Management Service.

20.1 About the Password Management Service

The Password Management Service provides REST endpoints for password management.

20.2 Accessing and Using the Password Management Service

The Password Management Service exposes resources to perform various password management functions.

NOTE: To test the REST service from remote client application, e.g. JAVA Swing, the client needs to pass a Base64 encoded user/password in the RESTAuthorization HTTP header. For example:

```
URLConnection passwordURL = (URLConnection) new
URL(password).openConnection();
passwordURL.setRequestMethod("GET");
passwordURL.setDoOutput(true);
passwordURL.setReadTimeout(20000);
passwordURL.setRequestProperty("RESTAuthorization", new
String(Base64.encodeBase64("username:password".getBytes("UTF-8")), "UTF-8"));
passwordURL.connect();
```

20.2.1 Available Resources

There are several types of resources available for the Password Management Service:

- ◆ Password Challenge Response
- ◆ Password Hint Change
- ◆ Change Password
- ◆ Password Policy Status
- ◆ Password Sync Status
- ◆ Locale Change

20.2.2 Complete URI Syntax

The following table shows the complete URI syntax for all resource end points associated with the Password Management Service, along with a description for each URI and a list of supported HTTP methods:

Table 20-1 Resource URIs

URI	Description
http(s)://host:port/war_context/roa/v1/pwdmgt/user/{userDN}/chares	<p>Password challenge response endpoint. Here is a sample URL:</p> <pre data-bbox="870 359 1442 432">http://localhost:9000/IDMProv/roa/v1/pwdmgt/user/cn=restuser,ou=Password,ou=medical-idmsample,o=netiq/chares</pre> <p>Supported HTTP methods:</p> <p>GET: Get all challenge response information related to the user DN specified in the URI in JSON format. - Authentication needed.</p> <p>POST: Post user answered challenge response answers to the server and save.</p>
http(s)://host:port/war_context/roa/v1/pwdmgt/user/{userDN}/hint	<p>Password hint change endpoint. Here is a sample URL:</p> <pre data-bbox="870 779 1442 852">http://localhost:9000/IDMProv/roa/v1/pwdmgt/user/cn=restuser,ou=Password,ou=medical-idmsample,o=netiq/hint</pre> <p>GET: Get all hint information related to the user DN specified in the URI in JSON format - Authentication needed.</p> <p>POST: Post user entered hint to the server and save.</p>
http(s)://host:port/war_context/roa/v1/pwdmgt/user/{userDN}/password	<p>Change password endpoint. Here is a sample URL:</p> <pre data-bbox="870 1094 1442 1167">http://localhost:9000/IDMProv/roa/v1/pwdmgt/user/cn=restuser,ou=Password,ou=medical-idmsample,o=netiq/password</pre> <p>Supported HTTP methods:</p> <p>GET: Get all password information related to the user DN specified in the URI in JSON format - Authentication needed.</p> <p>POST: Post user entered new password to the server and save.</p>
http(s)://host:port/war_context/roa/v1/pwdmgt/user/{userDN}/policy	<p>Password policy status endpoint. Here is a sample URL:</p> <pre data-bbox="870 1514 1442 1587">http://localhost:9000/IDMProv/roa/v1/pwdmgt/user/cn=restuser,ou=Password,ou=medical-idmsample,o=netiq/policy</pre> <p>Supported HTTP methods:</p> <p>GET: Get all password policy information related to the user DN specified in the URI in JSON format - Authentication needed.</p>

URI	Description
http(s)://host:port/war_context/roa/v1/pwdmgt/user/{userDN}/sync	<p>Password synchronization status endpoint. Here is a sample URL:</p> <pre>http://localhost:9000/IDMProv/roa/v1/pwdmgt/user/cn=restuser,ou=Password,ou=medical-idmsample,o=netiq/sync</pre> <p>Supported HTTP methods:</p> <p>GET: Get all password synchronization status information related to the user DN specified in the URI in JSON format - Authentication needed.</p>
http(s)://host:port/war_context/roa/v1/pwdmgt/user/{userDN}/locale	<p>Locale change endpoint. Here is a sample URL:</p> <pre>http://localhost:9000/IDMProv/roa/v1/pwdmgt/user/cn=restuser,ou=Password,ou=medical-idmsample,o=netiq/locale</pre> <p>Supported HTTP methods:</p> <p>GET: Get all locale information related to the user DN specified in the URI in JSON format - Authentication needed.</p> <p>POST: Post user entered new locale to the server and save.</p>

20.2.3 JSON Representations Received by the Client

This section provides details about the JSON representations returned by the client.

Password Challenge Response

Here is some sample return data:

```
[{"error_message":""}, {"have_stored_challenges":"true", "use_mask":"false"}, {"0":"What is your mother's maiden name?"}, {"1":"color1"}, {"use_grace_login":"true", "grace_login_remaining":"3"}]
```

Each element is described below:

- ♦ `error_message`: Any errors occurred during retrieving challenges data.
- ♦ `have_stored_challenges`: Indicates if the user has a stored challenge question.
- ♦ `use_mask`: Indicates if user's challenge answers should be marked when user enters challenge answers, this is configured through User Application password administration screen.
- ♦ `{"0":"What is your mother's maiden name?"}, {"1":"color1"}`: Challenge questions starting from index 0. First group is administrator defined questions, second group is user defined questions.
- ♦ `use_grace_login`: Indicates whether the user is in grace login.
- ♦ `grace_login_remaining`: Number of grace logins remaining.

Here is some sample POST data:

Table 20-2 Sample POST data

Field Name	Field Value
_answer0maiden	
_answer1redred	
_from_seq0	1
_from_seq1	2
_question0	What is your mother's maiden name?
_question1	color1

NOTE: _question0 _question1, _answer0, _answer1 are the challenge questions and answers sequence. _from_seq0 and _from_seq1 are corresponding to challenge questions and answers pair 0 and 1, this is because when form validation is kicked in, if validation failed, the error message sent back will refer to form fields starting from 1, instead of 0.

Here is some sample return data from the POST:

```
[{"success_message": "Challenge responses were saved successfully"}]
```

Each element is described below:

- ♦ success_message: Success message indicating that the save operation succeeded.
- ♦ error_message: Error message indicating that the save operation failed.

Password Hint Change

Here is some sample return data:

```
[{"hint": "new hint message", "hint_in_use": "Hint is not in use", "hint_invalid": "Hint is invalid", "message": "Any other error message"}, {"use_grace_login": "true", "grace_login_remaining": "3"}]
```

Each element is described below:

- ♦ hint: Password hint.
- ♦ hint_in_use: If this field exists in the returned JSON message, then it indicates the password hint is not in use with this user.
- ♦ hint_invalid: If this field exists in the returned JSON message, then it indicates the password hint is invalid.
- ♦ message: If this field exists in the returned JSON message, then it indicates some error occurred.
- ♦ use_grace_login: Indicates whether the user is in grace login.
- ♦ grace_login_remaining: Number of grace logins remaining.

Here is some sample POST data:

Table 20-3 Sample POST Data

Field Name	Field Value
hint	REST user name

Here is some sample return data from the POST:

```
[{"success_message": "Success"}]
```

Each element is described above:

- ◆ success_message: Indicates that the hint save operation was successful.
- ◆ error_message: Indicates that the hint save operation failed.

Change Password

Here is some sample return data:

```
[{"hintInUse": "true", "hint": "rest user name", "showSyncStatus": "false", "rules": "<tr><td class=\"nv-fontSmall \">Your password must have the following properties:</td></tr><tr><td class=\"nv-fontSmall nv-fontBold \"></td></tr><tr><td><ul><li class=\"nv-fontSmall\">Minimum number of characters in password: 4</li><li class=\"nv-fontSmall\">Maximum number of characters in password: 12</li></ul></td></tr><tr><td class=\"nv-fontSmall\">You may use numbers in your password.</td></tr><tr><td class=\"nv-fontSmall nv-fontBold\"></td></tr><tr><td><ul></ul></td></tr><tr><td class=\"nv-fontSmall\">The password is case sensitive.</td></tr><tr><td class=\"nv-fontSmall nv-fontBold\"></td></tr><tr><td><ul></ul></td></tr><tr><td class=\"nv-fontSmall\">You may use special characters in your password.</td></tr><tr><td class=\"nv-fontSmall nv-fontBold\"></td></tr><tr><td><ul></ul></td></tr><tr><td><ul></ul></td></tr>"}], {"error_message": "Error has occurred"}, {"use_grace_login": "true", "grace_login_remaining": "5"}]
```

Each element is described below:

- ◆ hint_in_use: Indicates whether password hint is in use with this user.
- ◆ hint: User's password hint.
- ◆ ShowSyncStatus: If to show password sync status screen after user changes password.
- ◆ rules: User's password rules.
- ◆ error_message: Error message if error has occurred.
- ◆ use_grace_login: Indicates whether the user is in grace login.
- ◆ grace_login_remaining: Number of grace logins remaining.

Here is some sample POST data:

Table 20-4 Sample POST Data

Field Name	Field Value
oldPassword	test
newPassword	netiq
retypeNewPassword	netiq

Here is some sample return data from the POST:

```
[{"pwdChgRtnPage":"","accessMgr":"false","pwd_chg_rtn_page":"Password Change Return Page","success_message":"Your password has been changed successfully."}]
```

Each element is described above:

- ◆ pwdChgRtnPage: Password change return page link - this is used when access manager is used when accessing REST service, so user knows where he can go after changing password.
- ◆ accessMgr: Indicates whether REST service is accessed through access manager.
- ◆ pwd_chg_rtn_page: This is just a localized label for pwdChgRtnPage link.
- ◆ success_message: Indicates that the password change is successful.
- ◆ error_message: Indicates that the password change failed.

Password Policy Status

Here is some sample return data:

```
[{"challengeresponse_status":"Valid","hint_status":"Invalid","password_status":"Valid"}, {"error":"error occurred"}, {"use_grace_login":"true","grace_login_remaining":"3"}]
```

Each element is described below:

- ◆ challengeresponse_status: Indicates if challenge response status is valid.
- ◆ hint_status: Indicates if hint status is valid.
- ◆ password_status: Indicates if password change is needed. If password change is needed, then this field will return "Invalid", else return "Valid".
- ◆ error: If exists, then error occurred.
- ◆ use_grace_login: Indicates whether the user is in grace login.
- ◆ grace_login_remaining: Number of grace logins remaining.

Password Synchronization Status

Here is some sample return data:

```
[{"0004":"Success","check_timeout":"3000","unchecked":"Unchecked","buffer_time":"0","0001":"Warning","0000":"Error","0003":"Fatal","0002":"Retry","image_servlet":"passwordSyncAppImage","error_occurred":"Error occurred, please contact administrator.","error_complete":"Application synchronization status check completed with errors. Contact your administrator for more details.","supervisor":"false","check_exceeded":"Check count exceeded.","timeout_message":"Application synchronization status check timeout.","success_complete":"Application synchronization status check completed successfully.","images_per_row":"4","processing":"Processing.....","no_app_configured":"There are no applications configured for Password Synchronization at this time.","process_count":"3","timeout":"300000","invalid-hash":"Hash Not Match"}, {"D37BBE8D658A344C4AADD37BBE8D658A":"App2"}, {"error_message":""}, {"use_grace_login":"true","grace_login_remaining":"2"}]
```

Each element is described below:

- ◆ Group 1: Returns a list of localized status message for displaying password sync status.

- ◆ Group 2: Returns a list of Configured Applications to check for password sync status. For each configured application, will return "App's GUID" and "App's Name" which is configured by User Application administrator.
- ◆ Group 3:
 - ◆ error_message: If an error occurs, this field shows the message.
- ◆ Group 4:
 - ◆ use_grace_login: Indicates whether the user is in grace login.
 - ◆ grace_login_remaining: Number of grace logins remaining.

Here is some sample POST data:

Table 20-5 Sample POST Data

Field Name	Field Value
guid	Application GUID
userDN	If checking user using Administrator credentials, then pass userDN, if user is self checking password sync status, then pass blank value.

Here is some sample return data from the POST:

```
[{"status": "0002"}, {"error_message": "Error occurred"}]
```

Each element is described above:

- ◆ Group 1: Returns user's password sync status on this application indicated by GUID.
- ◆ Group 2: If an error occurs, the error_message field shows the message.

Locale Change

Here is some sample return data:

```
[{"en": "English"}, {"NOT_AN_OPTION": "Select a locale to add..."}, {"zh-CN": "Chinese (China)", "zh-TW": "Chinese (Taiwan)", "nl": "Dutch", "fr": "French", "de": "German", "it": "Italian", "ja": "Japanese", "pt": "Portuguese", "ru": "Russian", "es": "Spanish", "sv": "Swedish"}, {"message": ""}, {"display_name": "FirstName LastName", "use_grace_login": "true", "grace_login_remaining": "5"}]
```

Each element is described below:

- ◆ Group 1: Returns a list of the user's configured locales.
- ◆ Group 2: Returns a list of available locales.
- ◆ Group 3: Any error messages are returned here.
- ◆ Group 4:
 - ◆ use_grace_login: Indicates whether the user is in grace login.
 - ◆ grace_login_remaining: Number of grace logins remaining.

Here is some sample POST data:

Table 20-6 Sample POST Data

Field Name	Field Value
locale	en fr

Here is some sample return data from the POST:

```
[{"message": "Locale Preferences Saved"}]
```

21 Resources Service

This section describes the Resources Services.

21.1 About the Resources Service

The Resources Service provides a REST endpoint for retrieving information about resources.

21.2 Accessing and Using the Resources Service

The Resources Service exposes resources to retrieve resources information. The service allows you to retrieve lists of resources or access specific resources by filter expression or resource ID.

IMPORTANT: To view resource detail and resource assignments by using the `resource/{resource id}` and `resource/{resource id}/assignments` end points, you need to be a Resource Administrator and a Provisioning Administrator. The Resource Administrator must have appropriate permissions on the Provisioning domain. To provide these permissions:

1. Log into the User Application as the Provisioning Administrator.
 2. Give all domain rights to the Resource Administrator including rights on the Provisioning domain.
 3. Login to iManager.
 4. Add Resource Administrator as a trustee of Resource-Config in iManager.
-

21.2.1 Available Resources

Resources Service

The service provides a resource URI for every object exposed. The Roles resource supports the following URI patterns:

- ◆ Resource for returning a collection of resources
- ◆ Resource for returning a specific resource instance
- ◆ Resource for returning all assignments for a specific resource instance
- ◆ Resource for returning a particular resource assignment

Services for Filtering, Debugging, and Displaying Schema Information

The service supports the following parameters to allow you to perform operations on the primary resources data set:

- ◆ A filter parameter to enable the filtering of result sets

- ♦ A debug matrix parameter to enable you to return the JSON structures in a human readable format
- ♦ A schema matrix parameter to enable you to return the schema for the data set

21.2.2 Complete URI Syntax

The following table shows the complete URI syntax for all resource end points associated with the Resources Service, along with a description for each URI and a list of supported HTTP methods:

Table 21-1 Resource URIs

URI	Description
/resources	<p>GET - Will return a list of resources with minimal information and a list of role URI links that include the role GUID</p> <p>POST – Not supported</p> <p>PUT – Not supported</p> <p>DELETE – Not Supported</p>
/resources/{ResourceId}	<p>GET - The Resource ID parameter is a GUID used to identify a specific resource within the LDAP realm</p> <p>POST – Not supported</p> <p>PUT – Not supported</p> <p>DELETE – Not Supported</p>
/resources/{ResourceId}/assignments	<p>GET - Will return a list of assignments for a specific resource. This will be a list of assignments to users.</p> <p>POST – Not supported</p> <p>PUT – Not supported</p> <p>DELETE – Not Supported</p>
/resources/{ResourceId}/assignments/{assignment id}	<p>GET - Displays information on a specific resource assignment.</p> <p>POST – Not supported</p> <p>PUT – Not supported</p> <p>DELETE – Not Supported</p>
/resources/refreshCodeMap	<p>GET - Not supported</p> <p>POST – Triggers a code map refresh for an entitlement. The request to trigger the refresh takes in an entitlement DN and the response includes a detailed status containing information about the codemap refresh, such as the last refresh time and refresh status.</p> <p>PUT – Not supported</p> <p>DELETE – Not Supported</p>

URI	Description
/resources/refreshStatus/{GUID}	<p>GET - Fetches the status of a code map refresh (RUNNING, COMPLETED, FAILED). The request to fetch the refresh status takes in an entitlement DN and returns details outlining the status of the code map refresh that was triggered previously.</p> <p>POST – Not supported</p> <p>PUT – Not supported</p> <p>DELETE – Not Supported</p>
/resources/checkCodeMapValueStatus	<p>GET - Not supported</p> <p>POST – Checks for the existence of an entitlement value in the RBPM code map.</p> <p>PUT – Not supported</p> <p>DELETE – Not Supported</p>
/resources/checkMultiCodeMapValueStatus	<p>GET - Not supported</p> <p>POST – Checks for the existence of multiple entitlement values in the RBPM code map.</p> <p>PUT – Not supported</p> <p>DELETE – Not Supported</p>

The following table lists the parameters for debugging and displaying the schema:

Table 21-2 Matrix Parameters for Debugging and Displaying the Schema

URI	Description
resources;debug	This <code>debug</code> matrix parameter displays the resources JSON structure in human readable format, as opposed to compressed format. This matrix parameter can be put anywhere in the URI.
resources;schema	The <code>schema</code> matrix parameter can be put anywhere in the URI and in combination with the “Accept” header type will return the appropriate schema document for the content type. In this case a JSON schema document is returned for the resources.

21.2.3 JSON Representations

This section provides the JSON representation for the resources endpoint.

Resources Endpoint

JSON Response The resources end point (/resources) returns a collection (JSON Array) of resources.

```

{
  "Link": "\\RIS\\resources\\9ca222fa9f942e4a7f879ca222fa9f94",
  "DN":
  "cn=BuildingAccessWest,cn=ResourceDefs,cn=RoleConfig,cn=AppConfig,cn=DoradoDriver,
  cn=TestDrivers,o=netiq",
  "GUID": "9ca222fa9f942e4a7f879ca222fa9f94"
  "Name": "West Building Access",
  "description" : "West Building Access",
  "CategoryKey": ["default"],
}

```

RefreshCodeMap Endpoint

JSON Request The refreshCodeMap endpoint (/resources/refreshCodeMap) sends a JSON request in this format.

```

{
  "DN" : "cn=Building Pass,cn=GroupEntitlementLoopback,cn=TestDrivers,o=netiq"
}

```

JSON Response The refreshCodeMap endpoint (/resources/refreshCodeMap) returns a detailed status containing information about the code map refresh.

```

{"GUID": "46d1b6b9aea3264b9ca946d1b6b9aea3", "Status": "true", "DN": "cn=Building
Pass,cn=GroupEntitlementLoopback,cn=TestDrivers,o=netiq", "DetailedStatus": [{"GUID
": "46d1b6b9aea3264b9ca946d1b6b9aea3", "Status": "SUCCESS", "Entitlement-
DN": "cn=Building Pass,cn=GroupEntitlementLoopback,cn=TestDrivers,o=netiq", "Last-
Refresh": "1329373072390", "Connection-Name": "1111"}]}

```

RefreshStatus Endpoint

JSON Response The refreshStatus endpoint (/resources/refreshStatus/{GUID}) returns details outlining the status of a previously triggered code map refresh.

```

{"CodeMapRefreshStatus": [{"GUID": "46d1b6b9aea3264b9ca946d1b6b9aea3", "Status": "1329
373072390", "Entitlement-DN": "cn=Building
Pass,cn=GroupEntitlementLoopback,cn=TestDrivers,o=netiq", "Last-
Refresh": "1329373072390", "Connection-Name": "1111"}]}

```

CheckCodeMapValueStatus Endpoint

JSON Request The checkCodeMapValueStatus endpoint (/resources/checkCodeMapValueStatus) sends a JSON request in this format:

```

{
  "entitlementDN" : "cn=Building
Pass,cn=GroupEntitlementLoopback,cn=TestDrivers,o=netiq",
  "connectionName" : "test",
  "codeMapValue" : "Waltham:Spring Street"
}

```

JSON Response The checkCodeMapValueStatus endpoint (/resources/checkCodeMapValueStatus) returns the status for an entitlement value in this format:

```
{ "UpToDate": "false", "Value": "Waltham:Spring
Street", "RefreshStatus": { "GUID": "46d1b6b9aea3264b9ca946d1b6b9aea3", "Status": "SUCCE
SS", "Entitlement-DN": "cn=Building
Pass,cn=GroupEntitlementLoopback,cn=TestDrivers,o=netiq", "Last-
Refresh": "1329712008390", "Connection-Name": "test" }}
```

CheckMultiCodeMapValueStatus Endpoint

JSON Request The checkMultiCodeMapValueStatus endpoint (/resources/checkMultiCodeMapValueStatus) sends a JSON request in this format:

```
{ "entitlementParams" : [
  {
    "entitlementDN" : "cn=Building
Pass,cn=GroupEntitlementLoopback,cn=TestDrivers,o=netiq",
    "connectionName" : "test",
    "codeMapValue" : "Waltham:Spring Street"
  },
  {
    "entitlementDN" : "cn=Parking
Permission,cn=GroupEntitlementLoopback,cn=TestDrivers,o=netiq",
    "connectionName" : "test1",
    "codeMapValue" : "Cambridge:Main Street"
  }
]
}
```

JSON Response The checkCodeMapValueStatus endpoint (/resources/checkMultiCodeMapValueStatus) returns the status for the specified entitlement values in this format:

```
{ "CodeMapValueStatus": [ { "UpToDate": "false", "Value": "Waltham:Spring
Street", "RefreshStatus": { "GUID": "46d1b6b9aea3264b9ca946d1b6b9aea3", "Status": "SUCCE
SS", "Entitlement-DN": "cn=Building
Pass,cn=GroupEntitlementLoopback,cn=TestDrivers,o=netiq", "Last-
Refresh": "1329712008390", "Connection-
Name": "test" } }, { "UpToDate": "false", "Value": "Cambridge:Main
Street", "RefreshStatus": { "GUID": "d33a80ae1cb82e4c80b9d33a80ae1cb8", "Status": "SUCCE
SS", "Entitlement-DN": "cn=Parking
Permission,cn=GroupEntitlementLoopback,cn=TestDrivers,o=netiq", "Last-
Refresh": "1329686444156", "Connection-Name": "test1" } } ] }
```

22 Roles Service

This section describes the Roles Service.

22.1 About the Role Service

The Roles Service provides a REST endpoint for retrieving information about roles.

22.2 Accessing and Using the Role Service

The Roles Service exposes resources to retrieve roles information. The service allows you to retrieve lists of roles or access specific roles by filter expression or role ID.

IMPORTANT: To view role detail and role assignments by using the `role/{role id}` and `role/{role id}/assignments` end points, you need to be a Role Administrator and a Provisioning Administrator. The Role Administrator must have appropriate permissions on the Provisioning domain. To provide these permissions:

1. Log into the User Application as the Provisioning Administrator.
 2. Give all domain rights to the Role Administrator including rights on the Provisioning domain.
 3. Login to iManager.
 4. Add Role Administrator as a trustee of Role-Config in iManager.
-

22.2.1 Available Resources

There are several types of resources available for the service.

Roles Service

The service provides a resource URI for every object exposed. The Roles resource supports the following URI patterns:

- ♦ Resource for returning a collection of roles
- ♦ Resource for returning a specific role instance
- ♦ Resource for returning assignments for a specific role instance
- ♦ Resource for returning SoDs for a specific role instance

Services for Filtering, Debugging, and Displaying Schema Information

The service supports the following parameters to allow you to perform operations on the primary roles data set:

- ♦ A filter parameter to enable the filtering of result sets

- ♦ A debug matrix parameter to enable you to return the JSON structures in a human readable format
- ♦ A schema matrix parameter to enable you to return the schema for the data set

22.2.2 Complete URI Syntax

The following table shows the complete URI syntax for all resource end points associated with the Roles Service, along with a description for each URI and a list of supported HTTP methods:

Table 22-1 Resource URIs

URI	Description
/roles	<p>GET - Will return a list of roles with minimum information and a list of role URI links that include the role GUID</p> <p>POST – Not supported</p> <p>PUT – Future. The PUT operation with the appropriate JSON structure will be used to create a new role. The Role JSON structure will be the same as the one used for the /roles/{RoleID} GET end point</p> <p>DELETE – Not Supported</p>
/roles?filter={role filter expression}	<p>GET - The above list can then be filtered by entering a role filter expression on the URI.</p> <p>The role filter expression must use the following syntax:</p> <p><i>Attribute%20Operator%20'Value'</i></p> <p><i>Attribute</i> must be one of the following:</p> <ul style="list-style-type: none"> ♦ Name ♦ Description <p><i>Operator</i> must be one of the following:</p> <ul style="list-style-type: none"> ♦ equal ♦ startwith ♦ endwith ♦ notequal <p>For example:</p> <p><i>/roles?filter=Description%20startwith%20'test'</i></p> <p>NOTE: The expression must include spaces between the <i>Attribute</i>, <i>Operator</i>, and <i>Value</i> elements, and the <i>Value</i> must be enclosed in single quotes (').</p>

URI	Description
/roles/{RoleId}	<p>GET - The Role ID parameter will be a GUID to identify a specific role within the LDAP realm.</p> <p>POST – Future. Provide ability to modify a specific role.</p> <p>PUT – Not supported</p> <p>DELETE – Future. Will remove the role.</p>
/roles/{RoleId}/assignments	<p>GET - Will return a list of assignments for a specific role. This will include assignments to other users, groups, and containers assigned to a role. It does not include role relationship information (information about roles assigned to another to a role).</p> <p>PUT – Future. The PUT operation with the appropriate JSON structure will be used to assign the role to another role, user, group, or container.</p> <p>POST – Not supported</p> <p>DELETE – Not supported</p>
/roles/{RoleId}/assignments/{assignmentID}	<p>GET – Display information on a specific role assignment. The assignment ID will be a GUID representing either a role, group, user or container.</p> <p>PUT – Not supported</p> <p>POST – Not supported</p> <p>DELETE – Future. Remove the specific assignment</p>
/roles/{RoleId}/sods	<p>GET - Will return a list of SODs for a specific role. This information will be crucial for clients that want to assign roles. Before the assignment, this REST end point should be executed to determine if any SODs exist for that particular role.</p> <p>PUT – Not supported</p> <p>POST – Not supported</p> <p>DELETE – Not supported</p>
/roles/{RoleId}/sods/{sodID}	<p>GET – Display information on a specific SOD.</p> <p>PUT – Not supported</p> <p>POST – Not supported</p> <p>DELETE – Not supported</p>

Table 22-2 Matrix Parameters for Debugging and Displaying the Schema

URI	Description
roles;debug	This <code>debug</code> matrix parameter displays the roles JSON structure in human readable format, as opposed to compressed format. This matrix parameter can be put anywhere in the URI.
roles;schema	The <code>schema</code> matrix parameter can be put anywhere in the URI and in combination with the "Accept" header type will return the appropriate schema document for the content type. In this case a JSON schema document is returned for the roles.

22.2.3 JSON Representations

This section shows the JSON structures received by the client for each resource. In each case, the HTTP method is GET.

Roles Endpoint

The roles end point (`/roles`) returns a collection (JSON Array) of roles.

```
[
  {
    "Links": [{
      "Link": "/RIS/v1/roles/372c47071345ae463db5372c47071345/assignments",
      "Type": "Assignments",
      "Value": "Assignments"
    }],
    {
      "Link": "/RIS/v1/roles/372c47071345ae463db5372c47071345/sods",
      "Type": "SODS",
      "Value": "SODS"
    }
  ],
  "DN": "cn=Scheduler System
Access,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=PicassoDriver,cn=TestD
rivers,o=netiq",
  "GUID": "372c47071345ae463db5372c47071345",
  "Link": "/POC/roa/v1/roles/372c47071345ae463db5372c47071345",
  "Name": "Scheduler System Access",
  "Description": "Scheduler System Access",
  "CategoryKey": ["system"],
  "RoleLevel": {
    "Name": "Permission Role",
    "Description": "Permission to connected systems",
    "level": "10"
  }
},
  {
    "Links": [{
      "Link": "/RIS/v1/roles/3059a9b358c8ba4f0ab53059a9b358c8/assignments",
      "Type": "Assignments",
      "Value": "Assignments"
    }],
    {
      "Link": "/RIS/v1/roles/3059a9b358c8ba4f0ab53059a9b358c8/sods",
```



```

        "Type": "SODS",
        "Value": "SODS"
    }],
    "DN": "cn=ER Access-
West,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=PicassoDriver,cn=TestDri
vers,o=netiq",
    "GUID": "3059a9b358c8ba4f0ab53059a9b358c8",
    "Link": "/RIS/v1/roles/3059a9b358c8ba4f0ab53059a9b358c8",
    "Name": "ER Access(West Campus)",
    "Description": "ER Access (West Campus)",
    "CategoryKey": ["building"],
    "RoleLevel": {
        "Name": "Permission Role",
        "Description": "Permission to connected systems",
        "level": "10"
    }
},
...
]

```

Roles/{RoleId} Endpoint

The roles/{RoleId} endpoint returns specific role information.

```

{
    "Links": [{
        "Link": "/RIS/v1/roles/372c47071345ae463db5372c47071345/assignments",
        "Type": "Assignments",
        "Value": "Assignments"
    }],
    {
        "Link": "/RIS/v1/roles/372c47071345ae463db5372c47071345/sods",
        "Type": "SODS",
        "Value": "SODS"
    }
}],
    "DN": "cn=Scheduler System
Access,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=PicassoDriver,cn=TestD
rivers,o=netiq",
    "GUID": "372c47071345ae463db5372c47071345",
    "Link": "/POC/roa/v1/roles/372c47071345ae463db5372c47071345",
    "Name": "Scheduler System Access",
    "Description": "Scheduler System Access",
    "CategoryKey": ["system"],
    "RoleLevel": {
        "Name": "Permission Role",
        "Description": "Permission to connected systems",
        "level": "10"
    }
}
}

```

Roles/{RoleId}/assignments Endpoint

This endpoint returns a list of assignments for a specific role.

```

{
  "Containers": [],
  "Groups": [],
  "Roles": [
    {
      "DN": "cn=Write
Prescriptions,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=PicassoDriver,c
n=TestDrivers,o=netiq",
      "GUID": "7cee8c618a2e8a436eb87cee8c618a2e",
      "Value": "Write Prescription",
      "link": "/RIS/v1/roles/7cee8c618a2e8a436eb87cee8c618a2e"
    },
    {
      "DN": "cn=Administer
Drugs,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=PicassoDriver,cn=TestDr
ivers,o=netiq",
      "GUID": "edcc5430f3e1a74041acedcc5430f3e1",
      "Value": "Administer Drug",
      "Link": "/RIS/v1/roles/edcc5430f3e1a74041acedcc5430f3e1"
    },
    {
      "DN": "cn=Order Medical
Tests,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=PicassoDriver,cn=TestDr
ivers,o=netiq",
      "GUID": "1a3381b0b849c04c1f981a3381b0b849",
      "Value": "",
      "Link": "/RIS/v1/roles/1a3381b0b849c04c1f981a3381b0b849"
    },
    {
      "DN": "cn=Perform Medical
Tests,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=PicassoDriver,cn=TestDr
ivers,o=netiq",
      "GUID": "4146615061202f4231b6414661506120",
      "Value": "",
      "Link": "/RIS/v1/roles/4146615061202f4231b6414661506120"
    }
  ],
  "Users": [{
    "DN": "cn=ablake,ou=users,ou=medical-idmsample,o=netiq",
    "GUID": "26b65d8611075849e2b226b65d861107",
    "Value": "Allison Blake",
    "Link": "/RIS/v1/identities/26b65d8611075849e2b226b65d861107"
  }],
  "DN":
  "cn=Doctor,cn=Level20,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=PicassoDriver,cn=T
estDrivers,o=netiq",
  "GUID": "80146f8473973942ceb380146f847397",
  "Link": "/RIS/v1/roles/80146f8473973942ceb380146f847397",
  "Name": "Doctor",
  "Description": "Doctor"
}

```

Roles/{RoleId}/assignments/{assignmentID} Endpoint

This endpoint returns information on a specific role assignment.

```

{
  "Users": [{
    "DN": "cn=ablake,ou=users,ou=medical-idmsample,o=netiq",
    "GUID": "26b65d8611075849e2b226b65d861107",
    "Value": "Allison Blake",
    "Link": "/RIS/v1/identities/26b65d8611075849e2b226b65d861107"
  }],
  "DN":
  "cn=Doctor,cn=Level20,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=netiq",
  "GUID": "80146f8473973942ceb380146f847397",
  "Link": "/RIS/v1/roles/80146f8473973942ceb380146f847397",
  "Name": "Doctor",
  "Description": "Doctor"
}

```

Roles/{RoleId}/sods Endpoint

This endpoint return a list of SoDs for a specific role. This information is crucial for clients that want to assign roles. Before the assignment, this REST end point should be executed to determine if any SODs exist.

```

[
  {
    "DN":
    "cn=SOD,cn=SodDefs,cn=RoleConfig,cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=netiq",
    "GUID": "80146f8473973942ceb380146f847397",
    "Link": "/RIS/v1/roles/3059a9b358c8ba4f0ab53059a9b358c8/sods/80146f8473973942ceb380146f847397",
    "Name": "sod name",
    "Description": "sod description",
    "Quorum": "50%",
    "ApprovalType": "SOD approval type",
    "Roles": [ { "DN":
    "cn=Doctor,cn=Level20,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=netiq",
      "GUID": "3059a9b358c8ba4f0ab53059a9b358c8",
      "Link": "/RIS/v1/roles/3059a9b358c8ba4f0ab53059a9b358c8",
      "Name": "Doctor",
      "Description": "Doctor" },
    { "DN":
    "cn=Nurse,cn=Level20,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=netiq",
      "GUID": "3059a9b358c8ba4f0ab53059a9b358c8",
      "Link": "/RIS/v1/roles/3059a9b358c8ba4f0ab53059a9b358c8",
      "Name": "Nurse",
      "Description": "Nurse" }
    ]
    "RequestDefinition": {
      "Value": "Role Approval",
      "DN": "cn=Role
Approval,cn=RequestDefs,cn=AppConfig,cn=CaribouDriver,cn=TestDrivers,o=netiq",

```

```

        "GUID": "c24dc77790ea497eb07617341c01e718",
        "Link": "/RIS/wf/definition/c24dc77790ea497eb07617341c01e718"
    },
    "Approvers" : [ { "DN": "cn=ablake,ou=users,ou=medical-idmsample,o=netiq",
        "GUID": "26b65d8611075849e2b226b65d861107",
        "Link": "/RIS/v1/identities/26b65d8611075849e2b226b65d861107",
        "Value" : "Allison Blake"},
        { "DN": "cn=mmackenzie,ou=users,ou=medical-idmsample,o=netiq",
        "GUID": "26b65d8611075849e2b226b65d861107",
        "Link": "/RIS/v1/identities/26b65d8611075849e2b226b65d861107",
        "Value" : "Margo Mackenzie"}
    ] ,
    ...
]

```

Roles/{RoleId}/sods/{sodID}

This endpoint returns information on a specific SoD.

```

{
  "DN":
  "cn=SOD,cn=SodDefs,cn=RoleConfig,cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=netiq",
  "GUID": "80146f8473973942ceb380146f847397",
  "Link": "/RIS/v1/roles/3059a9b358c8ba4f0ab53059a9b358c8/sods/80146f8473973942ceb380146f847397",
  "Name" : "sod name",
  "Description" : "sod description",
  "Quorum" : "50%",
  "ApprovalType" : "SOD approval type",
  "Roles" : [ { "DN":
  "cn=Doctor,cn=Level20,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=netiq",
    "GUID": "3059a9b358c8ba4f0ab53059a9b358c8",
    "Link": "/RIS/v1/roles/3059a9b358c8ba4f0ab53059a9b358c8",
    "Name": "Doctor",
    "Description": "Doctor"},
    { "DN":
  "cn=Nurse,cn=Level20,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=netiq",
    "GUID": "3059a9b358c8ba4f0ab53059a9b358c8",
    "Link": "/RIS/v1/roles/3059a9b358c8ba4f0ab53059a9b358c8",
    "Name": "Nurse",
    "Description": "Nurse"}
  ]
  "RequestDefinition" : {
    "Value": "Role Approval",
  }
}

```

```

        "DN": "cn=Role
Approval, cn=RequestDefs, cn=AppConfig, cn=CaribouDriver, cn=TestDrivers, o=netiq",
        "GUID": "c24dc77790ea497eb07617341c01e718",
        "Link": "/RIS/wf/definition/c24dc77790ea497eb07617341c01e718"
    },
    "Approvers" : [ { "DN": "cn=ablake, ou=users, ou=medical-idmsample, o=netiq",
        "GUID": "26b65d8611075849e2b226b65d861107",
        "Link": "/RIS/v1/identities/26b65d8611075849e2b226b65d861107",
        "Value" : "Allison Blake"},
    { "DN": "cn=mmackenzie, ou=users, ou=medical-idmsample, o=netiq",
        "GUID": "26b65d8611075849e2b226b65d861107",
        "Link": "/RIS/v1/identities/26b65d8611075849e2b226b65d861107",
        "Value" : "Margo Mackenzie"}
}

```

22.2.4 Event Status Codes

This section shows the event status codes for the available resources:

Table 22-3 Event Status Codes

URI	Status codes
roles	HTTP GET - Status 200 - OK HTTP PUT – Status 200 - OK
roles/{RoleId}	HTTP GET - Status 200 - OK HTTP POST – Status 200 – OK HTTP DELETE – Status 204 – No Content
roles/{RoleId}/assignments	HTTP GET - Status 200 - OK HTTP PUT – Status 200 - OK
roles/{RoleId}/assignments/{assignmentID}	HTTP GET - Status 200 - OK HTTP DELETE – Status 204 – No Content
roles/{RoleId}/sods	HTTP GET - Status 200 - OK
roles/{RoleId}/sods/{sodID}	HTTP GET - Status 200 - OK

The Jersey implemented error conditions are used. When server errors are found, Jersey returns the appropriate 400 and 500 level codes.

23 Work Items Service

This section describes the Work Items Service.

23.1 About the Work Items Service

The Work Items Service provides a REST endpoint for retrieving work items associated with provisioning workflows.

23.2 Accessing and Using the Work Items Service

The Provisioning Work Items Service exposes resources to retrieve provisioning work item information. The service allows a user is able to retrieve all work items related to himself or herself and then act upon a specific work item if so desired (Approve, Deny, Refuse).

Removing Administrator Credential Restrictions By default, the Provisioning Work Items Service requires that the HTTP session logged in user have administrator credentials. This restriction can be removed to allow a session with a logged in user who does not have administrator credentials to invoke the methods for the service. To allow non-administrative users to call workflow endpoints, you need to modify your configuration as described in [Section 11.1, “About the Provisioning Web Service,” on page 235](#) and [Section 14.1, “About the Directory Abstraction Layer \(VDX\) Web Service,” on page 331](#).

23.2.1 Available Resources

There are three types of resources available for the service.

Entry Point

The entry point URI for the Provisioning Work Items Service is:

```
/v1
```

The root entry point returns a list of all resources available.

Workitems Resource

The service provides a resource URI for every object exposed. The Work Items resource supports two basic URI patterns:

- ♦ Resource for returning a collection of work items
- ♦ Resource for returning a specific work item instance

Resources for Filtering, Debugging, and Displaying Schema Information

The service supports the following parameters to allow you to perform operations on the primary work items data set:

- ♦ A filter parameter to enable the filtering of result sets
- ♦ A debug matrix parameter to enable you to return the JSON structures in a human readable format
- ♦ A schema matrix parameter to enable you to return the schema for the data set

23.2.2 Complete URI Syntax

The following table shows the complete URI syntax for all resource end points associated with the Work Items Service, along with a description for each URI and a list of supported HTTP methods:

Table 23-1 Resource URIs

URI	Description
/v1	Entry point for the service.
/v1/wf/workitems	<p>Will return a collection (JSON Array) of work items available in the work flow sub system.</p> <p>Note: the URI is preceded with “wf/” this is to allow us to introduce other work flow related ROA services in the future such as “wf/processes” which will list all processes available in the work flow sub system.</p> <p>The following HTTP methods are supported with this URI:</p> <p>GET - This will return a collection of work items (JSON Array).</p> <p>POST – Not supported</p> <p>PUT – Not supported</p> <p>DELETE – Not supported</p>

URI	Description
/v1/wf/workitems?filter={parameter}={value}	<p>Return a collection of work items (JSON Array) for a specific addressee DN.</p> <p>The possible query parameters are listed below:</p> <p>Addressee Addressee DN</p> <p>DefinitionId Process DN</p> <p>ProcessId Process request instance ID (GUID)</p> <p>ActivityId Activity ID</p> <p>Status Status code (Integer)</p> <p>Owner Owner DN</p> <p>Priority Priority (Integer)</p> <p>CreationTime Creation time (date-time format) followed by operator (EQ, LT, LE, GT, GE). Example: 20080723044715000-0400EQ</p> <p>ExpTime Expiry time(date-time format) followed by operator (EQ, LT, LE, GT, GE). Example: 20080723044715000-0400EQ</p> <p>CompletionTime Completion time(date-time format) followed by operator (EQ, LT, LE, GT, GE). Example: 20080723044715000-0400EQ</p> <p>Recipient Recipient DN</p> <p>Initiator Initiator DN</p>
/v1/wf/workitems?filter=workid={work id}	<p>Return a specific work item (JSON Array) instance based on the work item GUID.</p> <p>Example:</p> <pre data-bbox="870 1209 1435 1285">/v1/wf/ workitems?filter=addressee%3dcn%3dadmin,ou%3 didmsample,o%3dnetiq</pre> <pre data-bbox="870 1444 1357 1495">/v1/wf/ workitems?filter=workid%3d456789afbc78</pre> <p>Note: the URI must be fully URL encoded.</p>

URI	Description
/v1/wf/workitems/{WorkId}	<p>This will return a single JSON Object with all the work item details. This will include the data items and actions allowed on the work item.</p> <p>The following HTTP methods are supported with this URI:</p> <p>GET - Return a specific work item instance (JSON Object) with all details. This will include the actions supported and data items available for that work item</p> <p>PUT – Forward the appropriate action on the work item.</p> <p>POST – Same as PUT because of limitations in browsers to set the PUT method</p> <p>DELETE – Not supported</p>

Table 23-2 Matrix Parameters for Debugging and Displaying the Schema

URI	Description
/v1/wf/workitems;debug	This <code>debug</code> matrix parameter displays the workitems JSON structure in human readable format, as opposed to compressed format. This matrix parameter can be put anywhere in the URI.
/v1/wf/workitems/{WorkId};debug	This <code>debug</code> matrix parameter displays the work ID JSON structure in human readable format, as opposed to compressed format. This matrix parameter can be put anywhere in the URI.
/v1/wf/workitems;schema	The <code>schema</code> matrix parameter can be put anywhere in the URI and in combination with the “Accept” header type will return the appropriate schema document for the content type. In this case a JSON schema document is returned for the workitems.
/v1/wf/workitems/{WorkId};schema	The <code>schema</code> matrix parameter can be put anywhere in the URI and in combination with the “Accept” header type will return the appropriate schema document for the content type. In this case a JSON schema document is returned for the work ID.

23.2.3 JSON Representations Received by the Client

This section shows the JSON structures received by the client for each resource. In each case, the HTTP method is GET.

In all JSON structures, date-time values follow this format:

yyyyMMddhhmssSSS-YYYY

Here is an example that illustrates the format:

20080723044715000-0400

Root entry point

The root entry point returns this structure:

```
[
  {
    "Link": "/RIS/v1/wf/workitems",
    "Type": "wf/workitems",
    "Value": "Workflow Workitems"
  }
]
```

Where:

Link – relative link to the service available

Type – Non localized internal code for the service.

Value – Localized value of the service.

Workitems end point

The workitems end point (/v1/wf/workitems) returns a collection (JSON Array) of work items available in the Workflow subsystem.

```
[
  {
    "AvailableActions": [
      {
        "Value": "Deny",
        "Code": "1"
      },
      {
        "Value": "Approve",
        "Code": "0"
      }
    ],
    "GUID": "6d8650ac062548fb84cb0e21bfc3cea6",
    "Link": "/RIS/v1/wf/workitems/6d8650ac062548fb84cb0e21bfc3cea6?filter=ProcessId%3Dc24dc77790ea497eb07617341c01e718",
    "ActivityId": "Activity",
    "ActivityName": "Approve Role Request (Serial)",
    "Addressee": {
      "DN": "cn=admin,ou=medical-idmsample,o=netiq",
      "Value": "Application Administrator Of Sample Data",
      "GUID": "",
      "Link": ""
    },
    "AssignmentType": "0",
    "Created": "20080723044715000-0400",
    "Owner": "Unclaimed",
    "Priority": "2",
    "Definition": {
      "Value": "Role Approval",
      "DN": "cn=Role Approval,cn=RequestDefs,cn=AppConfig,cn=CaribouDriver,cn=TestDrivers,o=netiq",
      "GUID": "",
      "Link": ""
    }
  }
]
```

```

    "Version": "8"
  },
  "Process": {
    "GUID": "c24dc77790ea497eb07617341c01e718",
    "Link":""
  },
  "Status": "0",
  "DigitalSignatureType": "not-required",
  "Initiator": {
    "DN": "CN=admin,OU=medical-idmsample,O=netiq",
    "Value": "Application Administrator Of Sample Data",
    "GUID": "",
    "Link":""
  },
  "Recipient": {
    "DN": "cn=admin,ou=medical-idmsample,o=netiq",
    "Value": "Application Administrator Of Sample Data",
    "GUID": "",
    "Link":""
  }
},
.....
]

```

Single workitem end point

The end point for accessing a single workitem (`/v1/wf/workitems/{work id}`) returns a single JSON Object with all the workitem details. This object includes the data items and actions allowed on the workitem.

```

{"AvailableActions":[{"Code":"0","Value":"Approve"}],"DataItems":[{"Name":"unmapped","Type":"string","Values":[{"Value":""}], "MultiValued":false}, {"Name":"reason","Type":"string","Values":[{"Value":"test"}], "MultiValued":false}, {"Name":"requester","Type":"dn","Values":[{"Value":"cn=uaadmin,ou=sa,o=data"}], "MultiValued":false}, {"Name":"multivaluedField","Type":"string","Values":[{"Value":"one"}, {"Value":"two"}], "MultiValued":true}], "GUID":"2417e36457f44a4d908aa3265e0f1e09","Link":"\\/RIS\\/wf\\/workitems\\/2417e36457f44a4d908aa3265e0f1e09","ActivityId":"Activity1","ActivityName":"Approval iPhone","Addressee":{"DN":"CN=uaadmin,OU=sa,O=data","GUID":"7f39d29c9e1de04b31bb7f39d29c9e1d","Link":"","Value":"aaron admin"},"AssignmentType":"0","Created":"20120402174027000-0400","Owner":"Unclaimed","Priority":"2","Definition":{"DN":"cn=startFromIphoneMultivalued,cn=RequestDefs,cn=AppConfig,cn=UserApplication,cn=DriverSet,o=netiq","GUID":"368c905cf6f3d048e2ac368c905cf6f3","Link":"\\/RIS\\/wf\\/definitions\\/368c905cf6f3d048e2ac368c905cf6f3","Value":"start from iphone multivalued"},"Process":{"GUID":"fa463317b85849bba3f6118bf4ff4dab","Link":"\\/RIS\\/wf\\/processes\\/fa463317b85849bba3f6118bf4ff4dab"},"Status":"0","Completed":"","Initiator":{"DN":"cn=uaadmin,ou=sa,o=data","GUID":"7f39d29c9e1de04b31bb7f39d29c9e1d","Link":"\\/RIS\\/identities\\/7f39d29c9e1de04b31bb7f39d29c9e1d","Value":"aaron admin"},"Recipient":{"DN":"cn=uaadmin,ou=sa,o=data","GUID":"7f39d29c9e1de04b31bb7f39d29c9e1d","Link":"\\/RIS\\/identities\\/7f39d29c9e1de04b31bb7f39d29c9e1d","Value":"aaron admin"},"Comment":""}

```

23.2.4 JSON Representations Sent by the Client

This section shows the JSON structure sent by the client for the single workitem end point when the HTTP PUT or HTTP POST method is used.

Single workitem end point

When the client uses the HTTP PUT or HTTP POST method with the `/v1/wf/workitems/{work id}` URI, the JSON Object structure is the same as the GET operation. However, the only information used by the server is the AvailableActions and the DataItems sections. The first available action found is taken as the forwarding action to perform on the work item.

```
{
  "AvailableActions": [
    {
      "Code": "0",
      "Value": "Approve"
    }
  ],
  "DataItems": [
    {
      "Name": "unmapped",
      "Type": "string",
      "Values": [
        {
          "Value": ""
        }
      ],
      "MultiValued": false
    },
    {
      "Name": "reason",
      "Type": "string",
      "Values": [
        {
          "Value": "test"
        }
      ],
      "MultiValued": false
    },
    {
      "Name": "requester",
      "Type": "dn",
      "Values": [
        {
          "Value": "cn=uaadmin,ou=sa,o=data"
        }
      ],
      "MultiValued": false
    },
    {
      "Name": "multivaluedField",
      "Type": "string",
      "Values": [
        {
          "Value": "one"
        },
        {
          "Value": "two"
        }
      ],
      "MultiValued": true
    }
  ],
  "GUID": "2417e36457f44a4d908aa3265e0f1e09",
  "Link": "\/RIS\/wf\/workitems\/2417e36457f44a4d908aa3265e0f1e09",
  "ActivityId": "Activity1",
  "ActivityName": "Approval",
  "Address": {
    "Type": "iPhone",
    "Addressee": {
      "DN": "CN=uaadmin,OU=sa,O=data",
      "GUID": "7f39d29c9e1de04b31bb7f39d29c9e1d",
      "Link": "",
      "Value": "aaron admin"
    },
    "AssignmentType": "0",
    "Created": "20120402174027000-0400",
    "Owner": "Unclaimed",
    "Priority": "2",
    "Definition": {
      "DN": "cn=startFromIphoneMultivalued,cn=RequestDefs,cn=AppConfig,cn=UserApplication,cn=DriverSet,o=netiq",
      "GUID": "368c905cf6f3d048e2ac368c905cf6f3",
      "Link": "\/RIS\/wf\/definitions\/368c905cf6f3d048e2ac368c905cf6f3",
      "Value": "start from iphone multivalued"
    },
    "Process": {
      "GUID": "fa463317b85849bba3f6118bf4ff4dab",
      "Link": "\/RIS\/wf\/processes\/fa463317b85849bba3f6118bf4ff4dab"
    },
    "Status": "0",
    "Completed": "",
    "Initiator": {
      "DN": "cn=uaadmin,ou=sa,o=data",
      "GUID": "7f39d29c9e1de04b31bb7f39d29c9e1d",
      "Link": "\/RIS\/identities\/7f39d29c9e1de04b31bb7f39d29c9e1d",
      "Value": "aaron admin"
    },
    "Recipient": {
      "DN": "cn=uaadmin,ou=sa,o=data",
      "GUID": "7f39d29c9e1de04b31bb7f39d29c9e1d",
      "Link": "\/RIS\/identities\/7f39d29c9e1de04b31bb7f39d29c9e1d",
      "Value": "aaron admin"
    },
    "Comment": ""
  }
}
```

23.2.5 Event Status Codes

This section shows the event status codes for the available resources:

Table 23-3 Event Status Codes

URI	Status codes
<code>/v1</code>	HTTP GET - Status 200 - OK
<code>/v1/wf/workitems</code>	HTTP GET - Status 200 - OK
<code>v1/wr/workitems/{work id}</code>	HTTP GET – Status 200 – OK
	HTTP PUT – Status 204 – OK with no content
	HTTP POST – Status 204 – OK with no content

The Jersey implemented error conditions are used. When server errors are found, Jersey returns the appropriate 400 and 500 level codes.

23.2.6 JSON Schema

The service supports the use of a schema matrix parameter to return the JSON schema for any returned data set. The JSON schema is based on the proposed schema for JSON as described at:

<http://www.json.com/json-schema-proposal/>

Root entry point schema

The schema for the root ROA entry point (/v1;schema) is as follows:

```
{
  "description" : "schema for: /v1",
  "type" : "array",
  "properties" :
  {
    "Link": {"type" : "string",
             "enum": ["/RIS/v1/wf/workitems"]}
    },
    "Type": {"type" : "string",
             "enum": ["wf/workitems"]}
    },
    "Value": {"type" : "string",
             "enum": ["Workflow Workitems"]}
    }
  }
}
```

Workitems end point schema

The schema for the workitems end point (/v1/wf/workitems;schema) is as follows:

```
{
  "description" : "schema for: /v1/wf/workitems",
  "type" : "array",
  "properties" :
  {
    "AvailableActions" :
    { "type" : "array",
      "properties" :
      {
        "Value" :
        { "type" : "string",
          "enum": ["Approve", "Deny", "Refuse"]}
        },
        "Code" :
        { "type" : "integer",
          "enum": [0, 1, 2]}
        }
      },
    "DataItems":
    { "type" : "array",
      "properties" :
      {
```

```

    "Name": { "type" : "string"},
    "Type": { "type" : "string"},
    "Value": { "type" : "string"}
  }
},
"GUID" : { "type" : "string"},
"Link" : { "type" : "string"},
"ActivityId" : { "type" : "string"},
"ActivityName" : { "type" : "string"},
"Addressee":
{ "type" : "object",
  "properties" :
  {
    "DN": { "type" : "string"},
    "Value": { "type" : "string"},
    "GUID": { "type" : "string"},
    "Link": { "type" : "string"}
  },
"AssignmentType": { "type" : "integer"},
"Created":
{ "type" : "string",
  "format" : "date-time"
},
"ExpiryDate":
{ "type" : "string",
  "format" : "date-time",
  "optional" : "true"
},
"Owner": { "type" : "string"},
"Priority": { "type" : "integer"},
"Definition":
{ "type" : "object",
  "properties" :
  {
    "Value": { "type" : "string"},
    "DN": { "type" : "string"},
    "GUID": { "type" : "string"},
    "Link": { "type" : "string"},
    "Version": { "type" : "string"}
  }
},
"Process":
{ "type" : "object",
  "properties" :
  {
    "GUID": { "type" : "string"},
    "Link": { "type" : "string"}
  }
},
"Status": { "type" : "integer",
  "minimum" : 0,
  "maximum" : 5
},
"LegalDisclaimer": { "type": "string",
  "optional": true
}
"DigitalSignatureType":
{ "type" : "string",
  "enum": ["data", "form", "not-required"]
},

```

```

"Completed":
{ "type" : "string",
  "format" : "date-time",
  "optional" : "true"
},
"Initiator":
{ "type" : "object",
  "properties" :
  {
    "DN": { "type" : "string"},
    "Value": { "type" : "string"},
    "GUID": { "type" : "string"},
    "Link": { "type" : "string"}
  }
},
"Recipient":
{ "type" : "object",
  "properties" :
  {
    "DN": { "type" : "string"},
    "Value": { "type" : "string"},
    "GUID": { "type" : "string"},
    "Link": { "type" : "string"}
  }
},
"ProxyFor": { "type": "object",
  "properties":
  {
    "DN": {"type": "string"},
    "Value": {"type": "string"},
    "GUID": {"type": "string"},
    "Link": {"type": "string"}
  },
  "optional": true
}
}
}
}

```

Single workitem end point schema

The schema for the single workitem end point (`/v1/wf/workitems/{work id};schema`) is as follows:

```

{
  "description" : "schema for: /v1/wf/workitems{work id}",
  "type" : "object",
  "properties" :
  {
    "AvailableActions" :
    { "type" : "array",
      "properties" :
      {
        "Value" :
        { "type" : "string",
          "enum": ["Approve", "Deny", "Refuse"]
        }
        "Code" :
        { "type" : "integer",
          "enum": [0, 1, 2]
        }
      }
    }
  }
}

```



```

},
"DataItems":
{ "type" : "array",
  "properties" :
  {
    "Name": { "type" : "string"},
    "Type": { "type" : "string"},
    "Value": { "type" : "string"}
  }
},
"GUID" : { "type" : "string"},
"Link" : { "type" : "string"},
"ActivityId" : { "type" : "string"},
"ActivityName" : { "type" : "string"},
"Addressee":
{ "type" : "object",
  "properties" :
  {
    "DN": { "type" : "string"},
    "Value": { "type" : "string"},
    "GUID": { "type" : "string"},
    "Link": { "type" : "string"}
  }
},
"AssignmentType": { "type" : "integer",
                    "minimum" : 0,
                    "maximum" : 17
                  },
"Created":
{ "type" : "string",
  "format" : "date-time"
},
"ExpiryDate":
{ "type" : "string",
  "format" : "date-time",
  "optional" : "true"
},
"Owner": { "type" : "string"},
"Priority": { "type" : "integer"},
"Definition":
{ "type" : "object",
  "properties" :
  {
    "Value": { "type" : "string"},
    "DN": { "type" : "string"},
    "GUID": { "type" : "string"},
    "Link": { "type" : "string"},
    "Version": { "type" : "string"}
  }
},
"Process":
{ "type" : "object",
  "properties" :
  {
    "GUID": { "type" : "string"},
    "Link": { "type" : "string"}
  }
},
"Status": { "type" : "integer",
            "minimum" : 0,
            "maximum" : 5
          }

```

```

    },
    "LegalDisclaimer": { "type": "string",
                        "optional": true
    }
  },
  "DigitalSignatureType":
  { "type" : "string",
    "enum": ["data", "form", "not-required"]
  },
  "Completed":
  { "type" : "string",
    "format" : "date-time",
    "optional" : "true"
  },
  },
  "Initiator":
  { "type" : "object",
    "properties" :
    {
      "DN": { "type" : "string"},
      "Value": { "type" : "string"},
      "GUID": { "type" : "string"},
      "Link": { "type" : "string"}
    }
  },
  "Recipient":
  { "type" : "object",
    "properties" :
    {
      "DN": { "type" : "string"},
      "Value": { "type" : "string"},
      "GUID": { "type" : "string"},
      "Link": { "type" : "string"}
    }
  },
  },
  "ProxyFor": { "type": "object",
               "properties":
               {
                 "DN": {"type": "string"},
                 "Value": {"type": "string"},
                 "GUID": {"type": "string"},
                 "Link": {"type": "string"}
               },
               "optional": true
  }
}

```

24 Workflow Process and Definition Service

This section describes the Workflow Process and Definition Service.

24.1 About the Workflow Process and Definition Service

The Workflow Process and Definition Service provides a REST interface for retrieving information about running workflow processes and provisioning request definitions (PRDs) available to the Workflow system.

The Workflow Process and Definition Service provides a REST interface for managing workflow processes. This will include REST end points for the identification and definitions of existing workflows and the ability to start an existing work flow. REST end points are also provided to allow you to display the status of current work flows in the Workflow system. Existing processes will also have a link to connect to the existing work items that are associated with that particular workflow process.

Workflow Processes Behave Differently in REST than in the User Application Workflows executed within the REST environment behave somewhat differently than the same processes running within the User Application. For example, a workflow request submitted through REST may succeed without providing any values for a mandatory field. This is because the User Application has access to the workflow form, whereas the REST interface does not. There is no way for the REST interface to know whether the data items are mandatory. It is up to the REST client to introspect the data items and enforce the business requirements.

24.2 Accessing and Using the Workflow Process and Definition Service

The Workflow Process and Definition Service exposes resources for managing workflow processes. The service allows a way for you to retrieve all provisioning request definitions in the system, or access a particular definition by ID. In addition, the service provides a way to retrieve all existing workflow processes, or access a particular process by ID.

24.2.1 Available Resources

There are four basic types of resources available with the Workflow Process and Definition Service:

Entry Point

The entry point URI for the Workflow Process and Definition Service is:

`/v1`

The root entry point returns a list of all resources available.

Definitions Resource

The Definitions resource supports two basic URI patterns:

- ◆ Resource for returning a collection of provisioning request definitions (PRDs)
- ◆ Resource for returning a specific PRD

Processes Resource

The Processes resource supports two basic URI patterns:

- ◆ Resource for returning a collection of existing workflow processes
- ◆ Resource for returning a specific process

Resources for Filtering, Debugging, and Displaying Schema Information

The service supports the following parameters to allow you to perform operations on the primary work items data set:

- ◆ A filter parameter to enable the filtering of result sets
- ◆ A debug matrix parameter to enable you to return the JSON structures in a human readable format
- ◆ A schema matrix parameter to enable you to return the schema for the data set

24.2.2 Complete URI Syntax

The following table shows the complete URI syntax for all resource end points associated with the Workflow Process and Definition Service, along with a description for each URI and a list of supported HTTP methods:

Table 24-1 Resource URIs

URI	Description
/v1	Entry point for the service.
/v1/wf/definitions	Returns a collection (JSON Array) of workflow definitions (provisioning request definitions) available in the Workflow system. The following HTTP methods are supported with this URI: GET - This will return a collection of workflow definitions in JSON Array. POST – Not supported PUT – Not supported DELETE – Not supported

URI	Description
/v1/wf/definitions/{definition ID}	<p>Returns a specific workflow definition (JSON Object) available in the Workflow system.</p> <p>The following HTTP methods are supported with this URI:</p> <p>GET - Returns a specific workflow definition (JSON Object) with all the details. This will include the data items associated with that definition.</p> <p>PUT – Not supported.</p> <p>POST – Start a work flow process.</p> <p>DELETE – Not supported</p> <p>Starting a workflow process To start a workflow process, you need to perform a GET operation to retrieve a provisioning request definition first. Once you have the JSON structure for the definition, you need to massage this JSON structure and perform a POST. For details on the JSON structure for GET operations, see Section 24.2.3, “JSON Representations Received by the Client,” on page 559. For details on the minimal JSON structure required for POST operations, see Section 24.2.4, “JSON Representations Sent by the Client,” on page 563.</p>
/v1/wf/processes	<p>Returns a collection of workflow processes in a JSON Array.</p> <p>The following HTTP methods are supported with this URI:</p> <p>GET - Returns a collection of workflow processes currently in the Workflow system in a JSON Array.</p> <p>POST – Not supported</p> <p>PUT – Not supported</p> <p>DELETE – Not supported</p>
/v1/wf/processes/{process id}	<p>Returns a single JSON Object with all the workflow process details.</p> <p>The following HTTP methods are supported with this URI:</p> <p>GET - Returns a specific work flow process (JSON Object) with all the details.</p> <p>PUT – Not supported.</p> <p>POST – Same as DELETE because of limitations in browsers to set the DELETE method</p> <p>DELETE – Terminates a work flow process</p>
/v1/wf/processes/{process id}/comments	<p>GET - Returns all comments for a specified workflow process.</p>

URI	Description
/v1/wf/processes?filter={process parameter}={process value}	<p>Returns a collection of work flow processes (JSON Array) for a specific process query parameter and value.</p> <p>The possible process query parameters are listed below:</p> <p>Definition Definition DN</p> <p>ProcessId Process request instance ID (GUID)</p> <p>EngineId Engine ID</p> <p>Recipient Recipient DN</p> <p>Initiator Initiator DN</p> <p>ApprovalStatus Approval status code (Integer)</p> <p>ProcessStatus Process status code (Integer)</p> <p>CreationTime Creation time (date-time format) followed by operator (EQ, LT, LE, GT, GE). Example: 20080723044715000-0400EQ</p> <p>CompletionTime Completion time(date-time format) followed by operator (EQ, LT, LE, GT, GE). Example: 20080723044715000-0400EQ</p> <p>CorrelationId Correlation ID</p> <p>Here is an example that illustrates filtering by recipient:</p> <pre>/v1/wf/processes?filter=recipient%3dcn%3dadmin,ou%3didmsample,o%3dnetiq</pre>

Table 24-2 Matrix Parameters for Debugging and Displaying the Schema

URI	Description
/v1/wf/definitions;debug	This <code>debug</code> matrix parameter displays the definitions JSON structure in human readable format, as opposed to compressed format. This matrix parameter can be put anywhere in the URI.
/v1/wf/definitions/{DefinitionID};debug	This <code>debug</code> matrix parameter displays the definition ID JSON structure in human readable format, as opposed to compressed format. This matrix parameter can be put anywhere in the URI.
/v1/wf/definitions;schema	The <code>schema</code> matrix parameter can be put anywhere in the URI and in combination with the “Accept” header type will return the appropriate schema document for the content type. In this case a JSON schema document is returned for the definitions.

URI	Description
/v1/wf/definitions/{Definition ID};schema	The <code>schema</code> matrix parameter can be put anywhere in the URI and in combination with the “Accept” header type will return the appropriate schema document for the content type. In this case a JSON schema document is returned for the specified definition ID.
/v1/wf/processes;debug	This <code>debug</code> matrix parameter displays the workflow processes JSON structure in human readable format, as opposed to compressed format. This matrix parameter can be put anywhere in the URI.
/v1/wf/processes/{Process ID};debug	This <code>debug</code> matrix parameter displays the process ID JSON structure in human readable format, as opposed to compressed format. This matrix parameter can be put anywhere in the URI.
/v1/wf/processes;schema	The <code>schema</code> matrix parameter can be put anywhere in the URI and in combination with the “Accept” header type will return the appropriate schema document for the content type. In this case a JSON schema document is returned for the workflow processes.
/v1/wf/processes/{Process ID};schema	The <code>schema</code> matrix parameter can be put anywhere in the URI and in combination with the “Accept” header type will return the appropriate schema document for the content type. In this case a JSON schema document is returned for the specified process ID.

24.2.3 JSON Representations Received by the Client

This section shows the JSON structures received by the client for each resource. In each case, the HTTP method is GET.

In all JSON structures, date-time values follow this format:

yyyyMMddhhmssSSS-YYYY

Here is an example that illustrates the format:

20080723044715000-0400

Root entry point

The root entry point returns this structure:

```
[
  {
    "Link": "/RIS/v1/wf/workitems",
    "Type": "wf/workitems",
    "Value": "Workflow Workitems"
  }
]
```

Where:

Link – relative link to the service available

Type – Non localized internal code for the service.

Value – Localized value of the service.

Definitions end point

The definitions end point (/v1/wf/definitions) returns a collection (JSON Array) of provisioning request definitions available in the Workflow System.

```
[
  {
    "Links": [
      {
        "Link": "/RIS/v1/wf/
processes?filter=Definition%3Dcn%3DPageWizardForm%2Ccn%3DRequestDefs%2Ccn%3DAppCon
fig%2Ccn%3DPicassoDriver%2Ccn%3DTestDrivers%2Co%3Dnetiq",
        "Type": "wf/processes",
        "Value": "Workflow Processes"
      },
      {
        "Link": "/RIS/v1/wf/
workitems?filter=Definition%3Dcn%3DPageWizardForm%2Ccn%3DRequestDefs%2Ccn%3DAppCon
fig%2Ccn%3DPicassoDriver%2Ccn%3DTestDrivers%2Co%3Dnetiq",
        "Type": "wf/workitems",
        "Value": "Workflow Workitems"
      }
    ],
    "DataItems": [{
      "Name": "recipient",
      "Type": "string",
      "Value": ""
    }],
    "Value": "PageWizardForm",
    "DN":
"cn=PageWizardForm,cn=RequestDefs,cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=n
etiq",
    "GUID": "8d64ba822ada934512968d64ba822ada",
    "Link": "/RIS/v1/wf/definitions/8d64ba822ada934512968d64ba822ada",
    "Category": "entitlements",
    "DigitalSignatureType": "not-required",
    "Description": "PageWizardForm",
    "Operation": "0"
  },
  .....
]
```

Single definition end point

The end point for accessing a single definition (/v1/wf/workitems/{work id}) returns a single JSON Object with all the details for the definition. The return payload also includes the data items required to start a work flow process.


```

{
  "Links": [
    {
      "Link": "/RIS/v1/wf/
processes?filter=Definition%3Dcn%3DPageWizardForm%2Ccn%3DRequestDefs%2Ccn%3DAppCon
fig%2Ccn%3DPicassoDriver%2Ccn%3DTestDrivers%2Co%3Dnetiq",
      "Type": "wf/processes",
      "Value": "Workflow Processes"
    },
    {
      "Link": "/RIS/v1/wf/
workitems?filter=Definition%3Dcn%3DPageWizardForm%2Ccn%3DRequestDefs%2Ccn%3DAppCon
fig%2Ccn%3DPicassoDriver%2Ccn%3DTestDrivers%2Co%3Dnetiq",
      "Type": "wf/workitems",
      "Value": "Workflow Workitems"
    }
  ],
  "DataItems": [{
    "Name": "recipient",
    "Type": "string",
    "Value": ""
  }],
  "Value": "PageWizardForm",
  "DN":
"cn=PageWizardForm,cn=RequestDefs,cn=AppConfig,cn=PicassoDriver,cn=TestDrivers,o=n
etiq",
  "GUID": "8d64ba822ada934512968d64ba822ada",
  "Link": "/RIS/v1/wf/definitions/8d64ba822ada934512968d64ba822ada",
  "Category": "entitlements",
  "DigitalSignatureType": "not-required",
  "Description": "PageWizardForm",
  "Operation": "0"
}

```

Workflow processes end point

The workflow processes end point (/v1/wf/processes) returns a collection (JSON Array) of running workflow processes available in the Workflow System.

```

[
  {
    "Links": [{
      "Type": "wf/workitems",
      "Value": "Workflow Workitems"
    }],
    "Link": "/POC/roa/v1/wf/
workitems?filter=ProcessId%3De11a2f10c90f489895f968d565b15091"
  },
  "GUID": "e11a2f10c90f489895f968d565b15091",
  "Link": "/RIS/v1/wf/processes/e11a2f10c90f489895f968d565b15091",
  "Initiator": {
    "DN": "CN=admin,OU=medical-idmsample,O=netiq",
    "Value": "Application Administrator Of Sample Data",
    "GUID": "",
    "Link": ""
  },
  "Recipient": {
    "DN": "cn=ablake,ou=users,ou=medical-idmsample,o=netiq",
    "Value": "Allison Blake",
    "GUID": ""
  }
}

```

```

    "Link": ""
  },
  "Definition": {
    "Value": "Role Approval",
    "DN": "cn=Role
Approval,cn=RequestDefs,cn=AppConfig,cn=CaribouDriver,cn=TestDrivers,o=netiq",
    "GUID": "d8c8e1a6d6432341fa84d8c8e1a6d643",
    "Link": "/RIS/v1/wf/definitions/e11a2f10c90f489895f968d565b15091",
    "Version": "8"
  },
  "Created": "20080723044715000-0400",
  "Completed": "",
  "ApprovalStatus": "Processing",
  "ProcessStatus": "Running",
  "Version": "8",
  "EngineId": "ENGINE"
}
]

```

Single workflow process end point

The end point for accessing a single process (/v1/wf/processes/{process id}) returns a single JSON Object with all the details for the process.

```

{
  "Links": [{
    "Type": "wf/workitems",
    "Value": "Workflow Workitems"
    "Link": "/POC/roa/v1/wf/
workitems?filter=ProcessId%3De11a2f10c90f489895f968d565b15091"
  }],
  "GUID": "e11a2f10c90f489895f968d565b15091",
  "Link": "/RIS/v1/wf/processes/e11a2f10c90f489895f968d565b15091",
  "Initiator": {
    "DN": "CN=admin,OU=medical-idmsample,O=netiq",
    "Value": "Application Administrator Of Sample Data",
    "GUID": "",
    "Link": ""
  },
  "Recipient": {
    "DN": "cn=ablake,ou=users,ou=medical-idmsample,o=netiq",
    "Value": "Allison Blake",
    "GUID": "",
    "Link": ""
  },
  "Definition": {

```

```

    "Value": "Role Approval",
    "DN": "cn=Role
Approval,cn=RequestDefs,cn=AppConfig,cn=CaribouDriver,cn=TestDrivers,o=netiq",
    "GUID": "d8c8e1a6d6432341fa84d8c8e1a6d643",
    "Link": "/RIS/v1/wf/definitions/e11a2f10c90f489895f968d565b15091",
    "Version": "8"
  },
  "Created": "20080723044715000-0400",
  "Completed": "",
  "ApprovalStatus": "Processing",
  "ProcessStatus": "Running",
  "Version": "8",
  "EngineId": "ENGINE"
}

```

24.2.4 JSON Representations Sent by the Client

This section shows the JSON structure sent by the client for the single workitem end point when the HTTP PUT or HTTP POST method is used.

Single definition end point

When the client uses the HTTP POST method with the `/v1/wf/definitions/{definition id}` URI, the following minimum information from a definition JSON structure is required to start a workflow process.

```

{
  "DataItems": [{
    "Name": "reason",
    "Type": "string",
    "Values": [{"Value": ""}],
    "MultiValued": false
  }],
  "Recipient": "cn=ablake,ou=users,ou=medical-idmsample,o=netiq"
}

```

All elements of the JSON object received from a GET call can be returned to the server. However, only the two items shown above are necessary. All other elements will be ignored by the server.

Single workflow process end point

When the client uses the HTTP DELETE method with the `/v1/wf/processes{process id}` URI, there is no need to send a JSON object to the server. This action terminates the specified workflow process. The key to the process instance is specified as part of the URI.

24.2.5 Event Status Codes

This section shows the event status codes for the available resources:

Table 24-3 Event Status Codes

URI	Status Codes
/v1	HTTP GET - Status 200 - OK

URI	Status Codes
/v1/wf/definitions	HTTP GET - Status 200 - OK
/v1/wf/definitions/{Definition ID}	HTTP GET – Status 200 – OK HTTP POST – Status 204 – OK with no content
/v1/wf/processes	HTTP GET - Status 200 - OK
/v1/wf/processes/{Process ID}	HTTP GET – Status 200 – OK HTTP DELETE – Status 204 – OK with no content HTTP POST – Status 204 – OK with no content

24.2.6 JSON Schema

The service supports the use of a schema matrix parameter to return the JSON schema for any returned data set. The JSON schema is based on the proposed schema for JSON as described at:

<http://www.json.com/json-schema-proposal/>

Root entry point schema

The schema for the root ROA entry point (/v1;schema) is as follows:

```
{ "Link": {
  "description": "schema for: /v1",
  "type": "array",
  "properties": {
    "Link": { "type": "string" },
    "Type": {
      "type": "string",
      "enum": [
        "root",
        "wf/definitions",
        "wf/processes",
        "wf/workitems"
      ]
    }
  },
  "Value": { "type": "string" }
}
```

Definitions end point schema

The schema for the definitions end point (/v1/wf/definitions;schema) is as follows:

```

{"DefinitionInfo": {
  "description": "schema for: /v1/wf/definitions",
  "type": "array",
  "properties": {
    "Links": {
      "type": "array",
      "properties": {
        "Link": {"type": "string"},
        "Type": {
          "type": "string",
          "enum": [
            "root",
            "wf/definitions",
            "wf/processes",
            "wf/workitems"
          ]
        }
      },
      "Value": {"type": "string"}
    },
    "DataItems": {
      "type": "array",
      "properties": {
        "Name": {"type": "string"},
        "Type": {"type": "string"},
        "Value": {"type": "string"}
      }
    },
    "Value": {"type": "string"},
    "DN": {"type": "string"},
    "GUID": {"type": "string"},
    "Link": {"type": "string"},
    "Category": {"type": "string"},
    "LegalDisclaimer": {"type": "string"},
    "DigitalSignatureType": {"type": "string"},
    "Description": {"type": "string"},
    "Operation": {"type": "string"},
    "Recipient": {"type": "string"}
  }
}}

```

Single definition end point schema

The schema for the single definition end point (/v1/wf/definitions/{Definition id};schema) is as follows:

```

{
  "description": "schema for: /v1/wf/definitions/{Definition ID}",
  "type": "object",
  "properties": {
    "Links": {
      "type": "array",
      "properties": {
        "Link": {"type": "string"},
        "Type": {
          "type": "string",
          "enum": [
            "root",
            "wf/definitions",
            "wf/processes",

```



```

    }
  },
  "Recipient": {
    "type": "object",
    "properties": {
      "DN": {"type": "string"},
      "Value": {"type": "string"},
      "GUID": {"type": "string"},
      "Link": {"type": "string"}
    }
  },
  "Definition": {
    "type": "object",
    "properties": {
      "Value": {"type": "string"},
      "DN": {"type": "string"},
      "GUID": {"type": "string"},
      "Link": {"type": "string"}
    }
  },
  "Created": {
    "type": "string",
    "format": "date-time"
  },
  "Completed": {
    "type": "string",
    "format": "date-time",
    "optional": true
  },
  "Proxy": {
    "type": "string",
    "optional": true
  },
  "CorrelationId": {
    "type": "string",
    "optional": true
  },
  "ApprovalStatus": {"type": "string"},
  "ProcessStatus": {"type": "string"},
  "Version": {"type": "string"},
  "EngineId": {"type": "string"}
}
}}

```

Single process end point schema

The schema for the single process end point (/v1/wf/processes/{Process ID};schema) is as follows:

```

{
  "description": "schema for: /v1/wf/processes/{Process ID}",
  "type": "object",
  "properties": {
    "Links": {
      "type": "array",
      "properties": {
        "Link": {"type": "string"},
        "Type": {
          "type": "string",
          "enum": [
            "root",
            "wf/definitions",
            "wf/processes",
            "wf/workitems"
          ]
        }
      }
    },
    "Value": {"type": "string"}
  },
  "GUID": {"type": "string"},
  "Link": {"type": "string"},
  "Initiator": {
    "type": "object",
    "properties": {
      "DN": {"type": "string"},
      "Value": {"type": "string"},
      "GUID": {"type": "string"},
      "Link": {"type": "string"}
    }
  },
  "Recipient": {
    "type": "object",
    "properties": {
      "DN": {"type": "string"},
      "Value": {"type": "string"},
      "GUID": {"type": "string"},
      "Link": {"type": "string"}
    }
  },
  "Definition": {
    "type": "object",
    "properties": {
      "Value": {"type": "string"},
      "DN": {"type": "string"},
      "GUID": {"type": "string"},
      "Link": {"type": "string"},
      "Version": {"type": "string"}
    }
  },
  "Created": {
    "type": "string",
    "format": "date-time"
  },
  "Completed": {
    "type": "string",
    "format": "date-time",
    "optional": true
  }
},

```



```

    "Proxy": {
      "type": "string",
      "optional": true
    },
    "CorrelationId": {
      "type": "string",
      "optional": true
    },
    "ApprovalStatus": {"type": "string"},
    "ProcessStatus": {"type": "string"},
    "Version": {"type": "string"},
    "EngineId": {"type": "string"}
  }
}

```

24.2.7 Testing the Client with the CURL Command

You can use the CURL command to test the REST client.

NOTE: To use the CURL command, you must also use either `RESTAuthorization` or `RESTSessionSecret`.

Table 24-4 *CURL Commands for Testing the Client*

URI	Command
<code>http://domain:port/RIS/v1</code>	<code>curl -v -H "Accept: application/json" http://domain:port/RIS/v1</code>
<code>wf/definitions</code>	<code>curl -v -H "Accept: application/json" http://domain:port/RIS/v1/wf/definitions</code>
<code>wf/definitions/{defintion id}</code>	<code>curl -v -H "Accept: application/json" http://domain:port/RIS/v1/wf/definitions/{definition id}</code>
<code>wf/definitions/{definition id}</code> To start a process, store the JSON structure in a file and pass it to the curl command.	<code>curl -v -H "Accept: application/json" -H "Content-Type: application/json" -X POST --data-binary @<filename of JSON structure> http://domain:port/RIS/v1/wf/definitions/{definition id}</code>
<code>wf/processes</code>	<code>curl -v -H "Accept: application/json" http://domain:port/RIS/v1/wf/processes</code>
<code>wf/processes/{request id}</code>	<code>curl -v -H "Accept: application/json" http://domain:port/RIS/v1/wf/processes/{request id}</code>
<code>wf/processes/{request id}</code> To terminate a process, store the JSON structure in a file and pass it to the curl command.	<code>curl -v -H "Accept: application/json" -X DELETE http://domain:port/RIS/v1/wf/processes/{request id}</code>

VII Appendixes

The following sections provide additional reference information and advanced topics for the Identity Manager User Application.

A Configuring the Identity Manager Approvals App

The NetIQ Identity Manager Approvals app allows managers and resource owners to approve or deny requests remotely, using an iPhone or iPad with the iOS operating system or any device with Android operating system installed. Your users can see and work with the same approval tasks in the app that they would normally see in the User Application interface. All changes are synchronized between the Approvals app and the User Application.

This appendix provides information about configuring your environment to allow users to use the new interfaces. These sections are intended to provide necessary information to administrators who want to enable and configure the Approvals app in their environment.

Most users should not need to refer to this document, but should instead be able to install, configure, and use the app without additional instructions. For information about installing or using the Approvals app, see “[Using the Identity Manager Approvals App](#)” in the *NetIQ User Application: User Guide*.

A.1 Product Requirements

The Approvals app has the following prerequisites:

- ◆ **On the Roles Based Provisioning Manager server:**
 - ◆ Identity Manager 4.5 Advanced Edition
 - ◆ Identity Manager Roles Based Provisioning Module 4.5
 - ◆ Designer for Identity Manager 4.5 or later with User Application driver and latest User Application Base package installed
 - ◆ SSL

NOTE: For detailed information on configuring and enabling SSL in your Identity Manager environment, see “[Enabling SSL](#)” on page 44.

- ◆ **On the device:** Apple iPhone or iPad with Apple iOS 5, iOS 6, or iOS 7 or any device with Android operating system

Enabling Non-Administrators to Use the Approvals App

If you want users who are not provisioning administrators on the Roles Based Provisioning Manager server to use the Approvals app, you must open the SOAP endpoints used by the server and app to non-provisioning administrator users.

NOTE: Opening SOAP endpoints to non-provisioning administrator users does not compromise security. Identity Manager continues to enforce all other existing security checks.

Complete the following steps to open the SOAP endpoints on the Roles Based Provisioning Module server:

- 1 Stop the server.
- 2 Create a backup of the existing `IDMProv.war` file.
- 3 Open the `IDMProv.war` file.
- 4 In `IDMProv.war`, open the file `WEB-INF/lib/IDMfw.jar`.
- 5 In `IDMfw.jar`, change the following configuration file properties to the specified values:

Configuration File	Property	Value
WorkflowService-conf/ config.xml	WorkflowService/SOAP- End-Points-Accessible- By-ProvisioningAdminOnly	false
	WorkflowService/soap/ addComment	false
	WorkflowService/soap/ getComments	false
VirtualDataService- conf/config.xml	VirtualDataService/soap	false

- 6 Save and close all files.
- 7 Restart the server.

A.2 Setting Up the Approvals App

Before your users can use the Approvals app, you must first configure your Identity Manager Roles Based Provisioning Module environment.

After installing the app, users can configure the app manually or automatically. Because manually configuring the Approvals app can be difficult, we recommend that administrators simplify the configuration process by providing users the necessary information as part of the provisioning process.

You provide configuration information to your users through a configuration link you customize for your Identity Manager Roles Based Provisioning Module environment. The structure of the configuration link is as follows:

```
idmapproval://settings/  
?userid=Username&passwordInKeychain=Password&host=HostName&port=PortNumber&  
rbpmContext=Context&userContainer=UserContainer&timeout=Timeout&vdxUserEntity=User  
Entity&  
vdxNameFormatAttribute=NameFormat&vdxFirstNameAttribute=FirstNameAttr&vdxLastNameA  
ttribute=  
LastNameAttr&vdxPhotoAttribute=UserPhotoAttr&vdxPhotoAttributeLdap=PhotoLDAPAttr&v  
dxPhoneAttribute=  
WorkPhoneAttr&vdxMobileAttribute=MobilePhoneAttr&vdxEmailAttribute=EmailAttr&namin  
gAttribute=  
NamingAttr&provAdminGetTasksWorkaroundInPlace=ProvisioningAdmin
```

The link must include settings specific to your environment, so that users can easily connect to the Roles Based Provisioning Module server from the Approvals app. However, none of the settings are explicitly required for the link. If you leave any setting values empty, each user must configure those settings on their device.

For example, if you want to provide a standard configuration link for all users in your environment, you would leave the `userid` and `passwordInKeychain` values empty:

```
idmapproval://settings/?userid=&passwordInKeychain=&host=123.112.20.109&port=8180&rbpmContext=IDMProv&userContainer=ou=users,o=data&timeout=5&vdxUserEntity=user&vdxNameFormatAttribute=FirstName%20LastName&vdxFirstNameAttribute=FirstName&vdxLastNameAttribute=LastName&vdxPhotoAttribute=UserPhoto&vdxPhotoAttributeLdap=photo&vdxPhoneAttribute=TelephoneNumber&vdxMobileAttribute=mobile&vdxEmailAttribute=Email&namingAttribute=cn&provAdminGetTasksWorkaroundInPlace=YES
```

For detailed information about the configuration settings, see [“Understanding Approvals App Settings” on page 575](#).

You can provide a configuration link to your users in one of the following ways:

- ◆ Customize and deploy the default “Request Mobile Approval App” process request definition (PRD) to your User Application. A user can log into the User Application and request access to the app using the PRD, which sends an email notification with a personalized configuration link that includes information specific to that user.
- ◆ Embed your custom configuration link in an HTML page hosted on a Web server in your environment. All users in your environment can navigate to the HTML page in a browser on their device and then click the configuration link.
- ◆ Create a QR code from the configuration link and embed the QR code in an HTML page. Users can use a QR code reader on their device to scan the code.

A.2.1 Understanding Approvals App Settings

An Approvals app configuration link can include the following settings:

Configuration Setting Name	Login Setting Description
<code>userid</code>	Specifies the user name the user uses to access the Roles Based Provisioning Module server.
<code>passwordInKeychain</code>	Specifies the password the user uses to access the Roles Based Provisioning Module server.
<code>host</code>	Specifies the fully qualified domain name or IP address of the Roles Based Provisioning Module server.
<code>port</code>	Specifies the HTTPS port the app uses to connect to the server.
<code>rbpmContext</code>	Specifies the context used when installing the User Application WAR file. The default value is <code>IDMProv</code> .

Configuration Setting Name	Login Setting Description
userContainer	<p>Specifies the full DN of the Identity Vault container that contains the user LDAP entry.</p> <p>NOTE: If you specify a user container to use, the Approvals app uses that container. If you do not specify a user container, the app attempts to detect the appropriate user container in the Identity Vault, searching all containers and subcontainers starting with the user container dn specified when running the User Application Configuration (configupdate) utility.</p> <p>If your Roles Based Provisioning Module environment includes a number of user containers, we recommend that you specify the container you want the app to use.</p> <p>You can configure a provisioning request definition (PRD) like the default “Request Mobile Approval App” definition to easily provision and configure your mobile end users. For more information about customizing the default PRD, see “Customizing and Using the Default Approvals App Provisioning Request Definition” on page 577.</p>
timeout	<p>Specifies the number of seconds the app waits when attempting to connect to the server before cancelling the connection. The default value is 5 seconds.</p>
vdxUserEntity	<p>Specifies the LDAP entity that represents a user in the Identity Vault. The default value is <code>user</code>.</p>
vdxNameFormatAttribute	<p>Specifies the DAL attribute representation the app uses to format a user's full name. The default value is <code>FirstName%20LastName</code>.</p>
vdxFirstNameAttribute	<p>Specifies the name of the DAL attribute that represents a user's first name. The default value is <code>FirstName</code>.</p>
vdxLastNameAttribute	<p>Specifies the name of the DAL attribute that represents a user's last name. The default value is <code>LastName</code>.</p>
vdxPhotoAttribute	<p>Specifies the name of the DAL attribute that contains a user's photo. The default value is <code>UserPhoto</code>.</p> <p>NOTE: If a user does not have a picture configured in the Identity Manager or has configured their Identity Manager settings to not display a picture, the app displays a generic image instead.</p>
vdxPhotoAttributeLdap	<p>Specifies the name of the LDAP attribute that contains the photo of the user. The default value is <code>photo</code>.</p>
vdxPhoneAttribute	<p>Specifies the name of the DAL attribute that represents a user's work phone number. The default value is <code>TelephoneNumber</code>.</p>
vdxMobileAttribute	<p>Specifies the name of the DAL attribute that represents a user's mobile phone number. The default value is <code>mobile</code>.</p>
vdxEmailAttribute	<p>Specifies the name of the DAL attribute that represents a user's email address. The default value is <code>Email</code>.</p>

Configuration Setting Name	Login Setting Description
namingAttribute	Specifies the naming DAL attribute used in the Identity Vault to describe a name. The default value is <code>cn</code> .
provAdminGetTasksWorkaroundInPlace	Specifies whether the user is a Provisioning Administrator on the Roles Based Provisioning Module server. The default value is <code>YES</code> .

A.2.2 Customizing and Using the Default Approvals App Provisioning Request Definition

As an administrator, you can use Designer to customize a generic “Request Mobile Approval App” PRD that your users can use, through the User Application, to request access to the Approvals application.

When a user requests access, Identity Manager then verifies that the user has the permissions required to access the mobile interface and that the Roles Based Provisioning Module server supports the application. If the server is not configured correctly or does not have the correct patch installed, the PRD generates a task for the provisioning administrator that lets the administrator know what needs to be fixed in order to enable use of the Approvals app.

After Identity Manager verifies the user and environment meet all requirements, the PRD triggers an email notification to the user. The user should open this email on the iPhone or iPad where the user has already installed the Approvals app.

This email notification includes a special `idmapproval://settings` link that automatically provides the settings the user needs to access the Approvals app from their device. The user clicks the link from their device and can then access their tasks through the Approvals app.

The default “Request Mobile Approval App” is included in the User Application Base package, which you can upgrade in Designer.

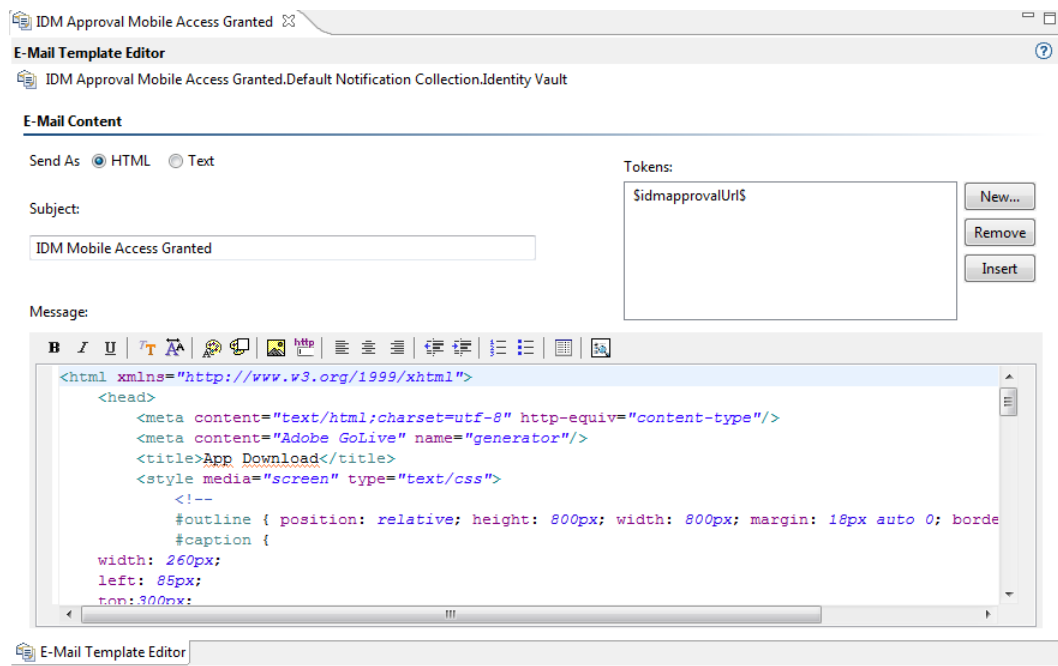
NOTE: The PRD and notification template provided in the User Application Base package are generic. Most administrators need to modify the generic PRD and template for their specific environments. However, we recommend that only users familiar with PRDs modify the default templates.

Customizing the Generic Notification Template

We recommend customizing the generic email notification template for your environment. To customize the default template to notify users they have access to the Approvals app and provide a link to automatically configure the app:

- 1 Ensure your Identity Manager environment meets all necessary requirements. For more information about prerequisites for using the Approvals app, see [“Product Requirements” on page 573](#).
- 2 In Designer, ensure you have a valid User Application driver in production. If a User Application driver does not exist in your Designer installation, install the driver before proceeding.

- 3 Upgrade the User Application Base package to the latest available version and install any dependent packages. For information about upgrading packages in Designer, see “Upgrading Installed Packages” (https://www.netiq.com/documentation/idm402/designer_admin/data/packman.html#packmanupgrade), in the *Designer for Identity Manager Administration Guide* (https://www.netiq.com/documentation/idm402/designer_admin/data/front.html).
- 4 In the Outline view, expand **Default Notification Collection**.
- 5 Right-click **IDM Approval Mobile Access Granted** and select **Edit**.
- 6 Modify the **Subject** field, if necessary.
- 7 In the **Message** field, modify the notification HTML as necessary for your environment. You can customize the email message text sent to your users, include graphics, or change the color and layout of the message to fit your company’s branding. The following image shows the default email message in the template editor:



WARNING: If you customize or modify the default notification template, do not remove or modify the token `$idmapprovalUrl$`, either in the Tokens list or in the HTML. The PRD uses the `$idmapprovalUrl$` token to provide the notification template a customized configuration link for the requesting user.

- 8 When finished making any customizations, close and save the notification template.
- 9 In the Outline view, right-click **IDM Approval Mobile Access Granted** and select **Live > Deploy**.
- 10 Click **Deploy**.
- 11 Click **OK**.

Customizing the Generic PRD

We recommend customizing the generic PRD for your environment. You can customize the category, workflow activities, entities, and forms. The PRD includes three forms by default:

- ♦ **request_form:** Request form users use to request access to the Approvals app.
- ♦ **approval_form:** Approval form managers use to approve or deny requests for access.

- ♦ **approval_form_prov_admin:** Approval form provisioning administrators use to fix issues with the provisioning server configuration.

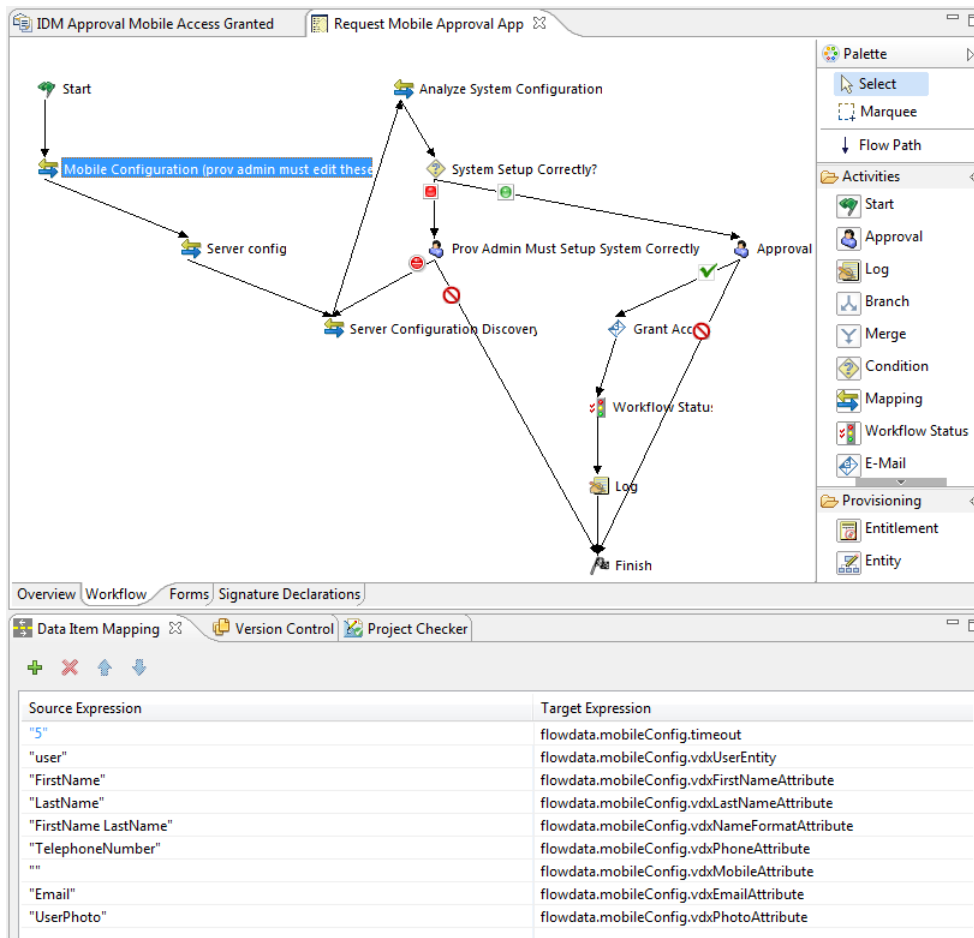
To create and customize a PRD to automatically configure your users' settings in the Approvals app:

- 1 In the Outline view, navigate to the User Application driver.
- 2 Expand **User Application Driver > Provisioning Request Definitions > Accounts**.
- 3 Right-click **Request Mobile Approval Access** and select **Edit**.
- 4 Modify the **Display Name** and **Description** fields, if necessary.
- 5 (Optional) If you want to move the PRD from the default Accounts category, click the **Category** drop-down menu and select the category you want to use.

NOTE: Most users do not need to modify the **Status**, **Flow Strategy**, and **Process Type** fields for the generic PRD. We recommend only advanced users modify these fields.

- 6 (Optional) By default, the User DAL entity does not have an attribute configured for a user's mobile telephone number. If you do not currently have a `mobile` attribute configured for User entities in your environment, you may need to add the attribute. Complete the following steps to add the attribute to the entity:
 - 6a In the Provisioning view, expand **User Application Driver > Directory Abstraction Layer > Entities**.
 - 6b Right-click **User** and select **Edit**.
 - 6c In the Data Abstraction Layer view, expand **Entities > User**.
 - 6d Right-click **User** and select **Add Attribute**.
 - 6e In the Add Attribute window, select the `mobile` attribute in the Available Attributes for Entity Class list.
 - 6f Click **Add Attribute** to move the attribute to the Entity Attributes list.
 - 6g Click **OK**.
 - 6h Close and save the Data Abstraction Layer.
- 7 Click the Workflow tab.
- 8 Click **Mobile Configuration (prov admin must edit these)**.
- 9 Click **Data Item Mapping**.
- 10 Edit the data item mapping expressions for the Mobile Configuration workflow activity. Ensure that the data item mapping matches the way your DAL User entity is configured.

The following image shows the workflow activity and data item mapping:



11 (Optional) If you want to modify the default **Trustee rights** for the PRD, complete the following steps:

11a Click the Overview tab.

11b Click the plus icon.

11c Select the group or user you want to be able to request access to the Approvals app.

NOTE: By default, the PRD trustee rights are set to [ROOT]. This default setting allows all users to request access to the Approvals app. Administrators can configure the trustee rights to limit access to only certain users, if necessary.

11d Click **OK**.

12 (Optional) If you want to customize the default PRD request and approval forms, complete the following steps:

12a In the Forms view, click the name of the form.

12b Modify the fields in the Form Controls window, as necessary.

12c Click the Preview icon to view the form.

12d Click **OK** when finished.

13 When finished, close and save the Request Mobile Approval App tab.

14 In the Outline view, right-click **Request Mobile Approval App** and select **Sync to Package**.

15 Right-click **Request Mobile Approval App** and select **Live > Deploy**.

16 Click **Deploy**.

17 Click **OK**.

A.2.3 Creating and Deploying a Custom Configuration Link

If you want to provide a “generic” set of configuration settings to any user who installs the Approvals app, you can embed a configuration link in an HTML page on a Web server your users can access.

Include the standard configuration link syntax in a link, as in the following example:

```
<a href="idmaproval://settings/  
?userid=&passwordInKeychain=&host=123.112.20.109&port=8180&rbpmContext=IDMProv&  
userContainer=ou=users,o=data&timeout=5&vdxUserEntity=user&vdxNameFormatAttribute=  
FirstName%20LastName&  
vdxFirstNameAttribute=FirstName&vdxLastNameAttribute=LastName&vdxPhotoAttribute=Us  
erPhoto&vdxPhotoAttributeLdap=  
photo&vdxPhoneAttribute=TelephoneNumber&vdxMobileAttribute=mobile&vdxEmailAttribut  
e=Email&namingAttribute=cn&  
provAdminGetTasksWorkaroundInPlace=YES">Configure Approvals App</a>
```

Unless you create a custom link for one specific user, most configuration links should leave the `userid` and `passwordInKeychain` values blank, providing the Roles Based Provisioning Module server information and Identity Vault information users need to be able to use the app.

A user clicks the link, and the link automatically configures the app with any settings you include in the link. The user then manually configures their Username and Password settings within the app.

A.2.4 Creating and Deploying a Custom Configuration QR Code

If your users cannot access their work email from their devices, you can create a QR code from the Approvals app configuration link and email that code to your users.

You can use any QR code generator you want to create the code, generating the code using a configuration link customized for your environment. Embed the code in an HTML page on a Web server your users can access.

For the example provided in [“Setting Up the Approvals App” on page 574](#), the QR code could look like the following image:



A user can then install the app, open the email on their work computer, and use a QR code reader on their device to scan the code displayed on the screen.

The QR code acts as a configuration link, automatically configuring the app with any settings you include in the link. In most environments, your users need to then manually configure their Username and Password settings within the app.

A.3 Optimizing Designer Forms for the Approvals App

The Approvals app renders Designer forms using either native iOS controls or HTML, depending on the complexity of each specific form. Native iOS controls provide a more standard look and feel to forms, while HTML-rendered forms look similar to forms in the User Application interface.

When creating new forms in Designer, we recommend simplifying forms as much as possible so that the app uses native iOS controls.

You can also configure your forms to display a more complex version of the form in the User Application and a less complex version in the Approvals app, using the suffix `_mobile`.

For example, if you have an Approval activity form called `approveLaptop`, you can create a new form called `approveLaptop_mobile` that acts as a simplified version of the original Approval activity form. In order for data item mapping to function correctly, the `_mobile` form must include the same fields as the original. We recommend you keep both versions of the form synchronized.

The following steps can help you optimize your forms so the app can render using iOS controls:

- 1 Ensure the Roles Based Provisioning Module server has the correct version and patch installed. The server must have version 4.0.2 Patch B or later installed.
- 2 Ensure the form has no scripts.
- 3 Ensure the form contains only fields with the following supported data types and control types:
 - ◆ `boolean`: any control type
 - ◆ `date`: any control type
 - ◆ `time`: any control type
 - ◆ `decimal` or `integer`: Text control type only
 - ◆ `dn`: `DNDisplay` or `read-only MVEditor` control types only
 - ◆ `string`: `Text`, `Password`, `Title`, `TextArea`, or `read-only MVEditor` control types only

For more detailed information about creating forms in Designer, see [“Creating Forms for a Provisioning Request Definition,”](#) in the *NetIQ User Application: Design Guide*.

A.4 Understanding Language Support in the Approvals App

The Approvals app includes localized text for all built-in text strings. For example, the titles displayed at the top of a view within the app are available in multiple languages, depending on the user’s locale. Approvals app strings are provided in the following languages, by default:

- ◆ Chinese (Simplified)
- ◆ Chinese (Traditional)
- ◆ Danish
- ◆ Dutch
- ◆ English
- ◆ French
- ◆ German
- ◆ Italian
- ◆ Japanese

- ♦ Portuguese (Brazilian)
- ♦ Russian
- ♦ Spanish
- ♦ Swedish

As an administrator, you can also localize the form text displayed in the Approvals app. For example, the Approvals app does not provide localized text for specific Approval tasks. You must localize text strings for each of your PRDs, including form text, using Designer. For information about localizing objects in Designer, see “[Localizing Provisioning Objects](#),” in the *NetIQ User Application: Design Guide*.

B Schema Extensions for the User Application

This section describes the schema extensions used by the User Application.

B.1 Attribute Schema Extensions

Attribute Name	Description
srvprvAllowMgrInitiate	A flag that indicates if the manager is allowed to initiate a provisioning request.
srvprvAllowMgrRetract	A flag to indicate if the manager is allowed to retract a provisioning request.
srvprvAllowMgrSetAvailability	A flag that indicates whether the manager can set a proxy for the team.
srvprvAllowMgrSetDelegate	A flag to indicate if the manager is allowed to set delegates for a provisioning request.
srvprvAllowMgrSetProxy	A flag to indicate if the manager is allowed to set a team proxy.
srvprvAllowMgrTaskClaim	A flag to indicate if the manager is allowed to claim a provisioning approval task.
srvprvAllowMgrTaskReassign	A flag to indicate if the manager is allowed to reassign a provisioning approval task.
srvprvAllRequests	A flag to indicate if the assignment covers all provisioning request definitions for a team.
srvprvAOLIMAddress	AOL IM address.
srvprvAssetRef	Representation of the aggregate asset properties for a named asset associated to a user via the <code>srvprvAssetRecipientAux</code> class.
srvprvAssignExpiration	Time at which a proxy or delegate assignment expires.
srvprvAssignFromContainer	Container subjects of a proxy or delegate assignment.
srvprvAssignFromGroup	Group subjects of a proxy or delegate assignment.
srvprvAssignFromUser	User subjects of a proxy or delegate assignment.
srvprvAssignStartTime	Time at which a delegation assignment takes effect.
srvprvAssignToRelationship	A target relationship of a delegate assignment.
srvprvAssignToUser	The User targets of a proxy or delegate assignment.
srvprvAutoDisplayTeam	Automatically display team members.

Attribute Name	Description
srvprvCapabilities1-5	Listing of skills for a user.
srvprvCategoryKey	Associates a given Provisioning Request Definition to a set of provisioning categories. Values are keys to a srvprvChoice instance.
srvprvCurrentDelegates	The delegations associated with a user.
srvprvCurrentDelegators	The delegations associated with a user.
srvprvDefaultTheme	The default theme.
srvprvDelegateDef	The delegates definition DN.
srvprvDelegationDef	The delegation definition DN.
srvprvDelegators	The users who are defined as delegators by this assignment.
srvprvEntitlementRef	Reference to a DirXML-Entitlement.
srvprvEntityType	Specifies Directory Abstraction Layer Entity definition type.
srvprvFlowStrategy	Specifies the flow invocation strategy to be used for the Provisioning Request Definition.
srvprvGrant	Flag which if true specifies that the Provisioning Request Definition supports a Grant operation.
srvprvGroupwiseIMAddress	Groupwise IM address.
srvprvHideAttributes	Flag indicating if certain attributes should be hidden and not displayed.
srvprvHideUser	Flag indicating if the user should be hidden when search list queries are executed.
srvprvIMAddress	Instant Messenger address.
srvprvIsTaskManager	Indicates if user is a task group manager.
srvprvLocalizedDescrs	Provides set of localized description strings for the provisioning web applications, Designers and iManager.
srvprvLocalizedNames	Provides set of localized display name strings for the provisioning web applications, Designers and iManager.
srvprvManager	Indicates users who are managers.
srvprvManagerGroup	Indicates a group containing managers.
srvprvManagerNotMember	Indicates that the manager is not a member of the team.
srvprvMember	Indicates users who are team members.
srvprvMemberContainer	The name of the container containing team members.
srvprvMemberGroup	The name of the group containing team members.

Attribute Name	Description
srvprvMemberRelationship	The name of the directory abstraction layer relationship that determines members based attribute in manager object.
srvprvModified	Flag to indicate changes to definitions object instances in the directory model container.
srvprvNotificationPrefs	Defines the set of notification types users want to receive.
srvprvPreferredLocale	Users preferred locale.
srvprvProcessXML	XML document representing a Provisioning process definition including Workflow and Provisioning Action.
srvprvQueryList	List of saved query/search criteria.
srvprvRelationship	Defines relationships between objects in the identity vault.
srvprvRequest	Exposes one item to be granted or revoked, including the workflow process which defines the run-time aspects of the Workflow and Provisioning Target.
srvprvRequestDefName	The provisioning request definition name associated with a delegate definition.
srvprvRequestScope	The scope of provisioning requests.
srvprvRequestXML	XML document representing the initial request form and its data bindings.
srvprvRevoke	If true, this flag specifies that the Provisioning Request Definition supports a Revoke operation.
srvprvStatus	Specifies the status of the Provisioning Object Supported values.
srvprvTaskGroups	Groups for which the user is a task manager.
srvprvTaskManager	Task manager of the task group.
srvprvTaskScopeAddressee	The addressee's task scope.
srvprvTaskScopeRecipient	The recipient's task scope.
srvprvTeam	The container for team definitions.
srvprvUser	The users associated with a delegation assignment.
srvprvUUID	Unique identifier for portlet.
srvprvYahooIMAddress	Yahoo* IM address.

B.2 Objectclass Schema Extensions

Objectclass Name	Description
srvprvAppConfig	Container for application configuration objects of the Provisioning System to which its DirXML-Driver parent connects.
srvprvAppDefs	Container for configuration objects used to initialize the Provisioning run-time environment, such as themes for the Identity Portal.
srvprvAssetRecipientAux	Records the provisioning of non-IT assets on a user.
srvprvChoice	Enumeration of values that can be assigned to a particular attribute, used in a query, for use in the Identity Portlets and other Web Application components.
srvprvChoiceDefs	Container for Directory Abstraction Layer Choice definitions, to be exposed by the Identity Portlets and Web Applications.
srvprvDelegateeAssignment	Delegates assignment definition.
srvprvDelegateeDefs	Container for delegates definitions.
srvprvDelegationAssignment	Delegation or availability assignment definition.
srvprvDelegationDefs	Container for delegation and delegators definitions.
srvprvDelegatorAssignment	Delegation or availability assignment definition.
srvprvDirectoryModel	Container for Directory Abstraction Layer meta-level objects, selected contents of the directory to be exposed by the Identity Portlets and Web Applications.
srvprvDirectoryModelConfig	Runtime Directory Abstraction Layer configuration parameters.
srvprvEntity	Defines a view of selected attributes for defined classes in the directory, used by the Identity Portlets and other Web Application components.
srvprvEntityAux	Standard ObjectClass.
srvprvEntityDefs	Container for Directory Abstraction Layer Entity definitions, to be exposed by the Identity Portlets and Web Applications.
srvprvProxyAssignment	Proxy assignment definition.
srvprvProxyDefs	Container for proxy definitions.
srvprvQuery	Directory abstraction layer query definition.
srvprvQueryDefs	Container for directory abstraction layer query definition.
srvprvRelationship	Defines relationships between objects in the directory, for use in the Identity Portlets and other Web Application components.

Objectclass Name	Description
srvprvRelationshipDefs	Container for Directory Abstraction Layer Relationship definitions, to be exposed by the Identity Portlets and Web Applications.
srvprvRequest	Exposes one item to be granted or revoked, including the workflow process which defines the run-time aspects of the Workflow and Provisioning Target.
srvprvRequestDefs	Container for Provisioning Request Definitions, the set of items to the Web Application run-time.
srvprvResource	Defines the set of directory assignments to execute for a provisioning fulfillment operation (either Grant or Revoke).
srvprvResourceDefs	Container for Provisioning Target definitions, including design-time descriptions plus any template or unused targets.
srvprvService	Describes how to invoke a specific Web Service from an Workflow This includes specification of input and return values.
srvprvServiceDefs	Container for Service Definition objects, which wrap Web Services called by Workflows.
srvprvTaskGroupAux	Service provisioning task group.
srvprvTeam	Team for provisioning request management.
srvprvTeamDefs	Container for team definitions.
srvprvTeamRequest	Team provisioning requests.
srvprvTheme	Theme Object.
srvprvUserAux	Service provisioning user entity.
srvprvWebAppConfig	Web Application configuration object.
srvprvWorkflow	Defines the network of activities including traversal conditions to be executed in order to obtain approval for a provisioning action.
srvprvWorkflowDefs	Container for Workflow objects, including design-time descriptions plus any template or unused flows.

B.3 Resource Definition Object (nrfResource)

The schema object that contains provisioning resource definitions.

Table B-1 Resource Definition Object Schema Definition

Attribute Name	Description
nrfLocalizedName	The localized name of the resource.
nrfLocalizedDescrs	The localized description of the resource.

Attribute Name	Description
Owner	The owner of the resource. It is the DN of an inetOrgPerson user.
nrfRequestDefGrant	Provisioning request definition used for approving the granting of a resource assignment.
nrfRequestDefRevoke	Provisioning request definition usef for approving the revocation of a resource assignment.
nrfEntitlementRef	IDM entitlement associated with the resource. Supports embedding of dynamic parameter macros to allow users to specify values at request time.
nrfApprovers	Resource approvers. Order of approvers is maintained by an integer in the second element.
nrfQuorum	Used to support quorum approvals in tempated PRDs. This is the quorum condition. Can be percentage or number of approvers required.
nrfDynamicParameters	XML document that describes allowable parameter values that can be specified at request time when the resource is being granted.
nrfCategoryKey	Used to categorize resource.
nrfAllowAprOverride	Allow requesting system (such as role provisioning) to override approval of the resource provisioning.
nrfAllowMulti	Allow the resource to be assigned to the same user multiple times.

B.4 Resource Request Object (nrfResourceRequest)

The schema object whose instances contain a resource request object. The resource request object is used by the resource driver to provision the resource.

Table B-2 Resource Request Object Sechema Definition

Attribute	Description
nrfRequestDate	Date-time resource request started.
nrfCategory	10-Resource To User Add 15 - Resource to User Remove
nrfResource	DN of resource to grant or revoke.
nrfEntitlementRef	Entitlement reference value of the resource being granted. This value is copied from the resource definition with parameter values populated at the time of the request.
nrfTargetDN	DN of user who will be granted the resource or from whom the resource will be revoked.
nrfRequester	DN of user or role that requested assignment.
nrfStatus	Status of request. Valid codes are described in Section B.4.1, "Resource Request Status Codes (nrfStatus)," on page 591.
nrfDescription	Description/Comment of the resource request.

Attribute	Description
nrfRequestDef	Provisioning request definitoin used for approving the role
nrfApprovers	Resource approvers. Order of approvers can be maintained by an integer in the second element.
nrfQuorum	Used to support quorum approvals in templated PRDs. The quorum condition can be percentage or numbers of approvers required.
nrfApprovalInfo	Holds approval data needed by resource view and reports.
nrfApprovalProcessid	Workflow process instance ID for resource assignment approval.

B.4.1 Resource Request Status Codes (nrfStatus)

Table B-3 Valid Resource Request (nrfStatus) Status Codes

Status Code	Key	Description
01	New Request	Initial value when request is created
12	Approval_Retry	
13	Pending_Approval_RETRY	
15	Approval Pending	Set by driver after successful assignment/revocation workflow.
20	Approved	Set by resource assignment/revocation workflow when approved.
30	Provision/Deprovision	Set by driver after all necessary approvals have been approved and role activation time has been reached.
50	Provisioned/Deprovisioned	Set by driver after role has been provisioned or deprovisioned.
70	Cancel	Request cancellation
75	Cancelled	Cancellation request completed.
80	Provisioning Error	Set by driver when an error occurred during provisioning or deprovisioning.
95	DeniedSet	Set by assignment/revocation workflow when approved.
100	CleanupSet	When nrfResourceRequest workflow should be deleted.

B.5 Role-Resource Configuration (nrfConfiguration)

Table B-4 Role-Resource Configuration Object Schema

Attribute	Definition
nrfResourceRequestContainer	Root container for resource requests.
nrfResourcesContainer	Root container for resource definitions.
nrfResourceRevokeRequestDef	Default PRD for approving resource revocations
nrfResourceGrantRequestDef	Default PRD for approving resource assignments.

B.6 Resource Binding to Users (nrfIdentity)

Table B-5 Resource Binding to Users Object Schema

Attribute	Description
nrfResource	Currently assigned and assigned resources. Attribute contains DN for the resource DN, the binding state of the resource, and the cause of the assignment and approval information.
nrfResourceHistory	Contains historical information about each resource grant, revocation, denial. Contains the resource as well as XML that contains the resource binding state, (0=inactive, 1=active, 2=pending, 3= deactivated). The XML also contains the entitlement reference value used to grant the entitlement, grant history (who and when), and revocation history (similar to approval information)

B.7 Resource Containers

ResourceRequests (nrfResourceRequests): A container objects that persists resource requests.

ResourceDefs (nrfResourceDefs): A container object that persists the definition of a resource.

C JavaScript Search API

The underlying framework for the Identity Manager User Application supports a JavaScript API for executing searches that access the Directory Abstraction Layer. This API lets you build, save, and execute queries from a JSP page running outside of the User Application itself. To run a query, you can invoke the services of the SearchListPortlet, passing parameters that specify the search criteria and formatting options. Alternatively, you can run a search by using the API directly without involving the SearchListPortlet.

C.1 Launching a Basic Search using the SearchListPortlet

To perform a basic search, you can specify a *deep link* to the SearchListPortlet from a JSP page. The URL for the portlet must either pass a simple set of request parameters that specify the search criteria, or pass a JSON-formatted query string. A basic search defines a single search criterion, such as the following:

```
First Name starts with A
```

To launch a search, you can call the single portlet render url for the SearchListPortlet. You must pass the request parameter `MODE=MODE_RESULTS_LIST`

C.1.1 Passing Request Parameters

You can pass a simple set of request parameters to the SearchListPortlet. These parameters specify an entity, an attribute to search on, an operator, and a search string. The following script shows the URL for the portlet, as well as the four request parameters you need to use:

```
<script type="text/javascript">
function openSearchResults(extraUrlParams) {
    var url = "/IDMProv/portal/portlet/SearchListPortlet?";
    url += "urlType=Render&novl-regid=SearchListPortlet";
    url += "&novl-inst=IDMProv.SearchListPortlet";
    url += "&wsrp-mode=view&wsrp-windowstate=normal";
    url += "&MODE=MODE_RESULTS_LIST&";
    url += extraUrlParams;
    var feat = "width=700,height=600";
    feat += ",menubar=no,resizable=yes,toolbar=no,scrollbars=yes";
    var win = window.open(url, "TestSearchPopup", feat);
    if (win) win.focus();
}

var search1a = "ENTITY_DEF=user";
search1a += "&COND_ROW_ATTR=FirstName";
search1a += "&COND_ROW_REL_OP=starts-with";
search1a += "&COND_ROW_VAL=A";
...

```

To call this function, you might have a button on the form with onclick event that looks like this:

```
<input type="button" value="GO" onclick="openSearchResults(search1a)"/>
```

The following table describes the request parameters:

Table C-1 Request Parameters for Basic Search

Request Parameter	Description
ENTITY_DEF	Specifies the key value for an entity in the Directory Abstraction Layer.
COND_ROW_ATTR	Specifies the attribute to search on.
COND_ROW_REL_OP	Specifies the operator to use in the search expression. The following operators are supported for attributes of type string, boolean, integer, time, dn_lookup, dynamic_list, and static_list: equals present not_equals not_present The following operators are supported for attributes of type string: starts_with ends_with contains not_starts_with not_ends_with not_contains The following operators are supported for attributes of type integer and time: greater greater_or_equal less less_or_equal not_greater not_greater_or_equal not_less not_less_or_equal
COND_ROW_VAL	The value to search on.

C.1.2 Using a JSON-formatted String to Represent a Query

If you prefer to format your query as a JSON string, you need to pass the QUERY parameter to the SearchListPortlet, instead of the request parameters described in the section above. The JavaScript variable shown below illustrates how the QUERY parameter is constructed:

```

var search1b = 'QUERY={"k":"Lastname starts with B","mxPg":"10",' ;
search1b += ' "mxRes": "0", "ptr": "1", "grp": [{"map": {"row": [{"map": {' ;
search1b += ' "rowRop": "starts-with", "rowVal": "B", "rowAttr": "LastName" ' ;
search1b += ' }]}], "rowLop": "and"}]} , ' ;
search1b += ' "orderBy": "LastName", "entDef": "user", ' ;
search1b += ' "sScope": " ", "sRoot": " ", "grpLop": "and", ' ;
search1b += ' "selAttr": ["FirstName", "LastName", ' ;
search1b += ' "Title", "Email", "TelephoneNumber"]}' ;

```

The JSON structure gives you a way to specify values for most of the settings and preferences associated with the SearchListPortlet.

The following table describes the JSON name/value pairs that define the QUERY parameter passed to the SearchListPortlet:

Table C-2 JSON Structure for Defining the QUERY Parameter

JSON Setting	Description
k	Specifies a name for the search. (Optional)
mxPg	Specifies the maximum number of rows per page. (Optional)
mxRes	Specifies the maximum number of total rows retrieved. (Optional)
ptr	Sets the scroll pointer, which defines the pagination offset. (Optional)
grp	Defines a condition group. You can specify one or more condition groups. For details on the settings for a condition group, see Table C-3 on page 595 .
orderBy	Specifies the attribute to sort on. (Optional)
entDef	Specifies an entity in the Directory Abstraction Layer.
sScope	Sets the search scope. (Optional)
sRoot	Sets the search root. (Optional)
grpLop	Defines the logical operator (<code>and</code> or <code>or</code>) for groups within this query.
selAttr	Lists the attributes to include in the search results.

The following table describes the JSON structure for defining a condition group:

Table C-3 JSON Structure for Defining a Condition Group

JSON Setting	Description
row	Defines a condition row. You can specify one or more condition rows. For details on the settings for a condition row, see Table C-4 on page 596 .
rowLop	Defines the logical operator (<code>and</code> or <code>or</code>) for rows within this group.

The following table describes the JSON structure for defining a condition row:

Table C-4 JSON Structure for Defining the Fields for a Condition Row

JSON Setting	Description
rowRop	Defines the relational operator. The relational operators supported in JSON are the same as those for basic searches using request parameters. For a complete list of the relational operators, see the description of COND_ROW_REL_OP in Table C-1 on page 594 .
rowVal	Sets the search value.
rowAttr	Specifies the attribute to search on.

C.2 Creating a New Query using the JavaScript API

As an alternative to using the basic search request parameters, or the JSON structure, you can call a JavaScript API to execute queries. This section describes some simple techniques for using the API, as well as reference documentation for the API.

The search API relies on the ajax framework embedded in the User Application component named JUICE. JUICE (JavaScript UI Controls and Extensions) is compliant with and uses the dojo library. JUICE is merged into the dojo release used in the User Application.

Therefore, to use JUICE on a custom page within the IDM User Application WAR file, you need to have a script reference to dojo.js (not to JUICE). After adding the reference to dojo.js, you can add a JavaScript line to tell dojo to download JUICE.

Before using the JavaScript API, you need to perform some setup steps on the page to make the dojo module available for use:

- 1 Add a script tag for dojo.js in the HTML header. The reference to dojo.js must be in the header (not the body), as shown below.

```
<html>
<head>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JavaScript Search</title>
<script type="text/javascript">
  if(typeof dojo=="undefined"){
    var djConfig={isDebug: false,
      baseScriptUri: "/IDMProv/javascript/dojo/"};
    var buf="<script type='text\/javascript' ";
    buf+="src='/IDMProv/UIQuery?js=dojo\/dojo.js'><\/script>";
    document.writeln(buf);
  }
</script>
</head>
```

- 2 Add this JavaScript statement to load JUICE into the browser's memory:

```
<script type="text/javascript">
  //This line must precede any code using JUICE.
  dojo.require("JUICE.*");
</script>
```

- 3 To take advantage of the JUICE.IDM services, which include entity searching, also add this JavaScript statement:

```
<script type="text/javascript">
  //This line must precede any code using JUICE.IDM services.
  dojo.require("JUICE.IDM.*");
</script>
```

To build the query, you need to call the create() method on the JUICE.IDM.Entities.Search object, passing in the name you want to give to the query. The create() method is a static method. Here's how you invoke it:

```
var newQuery = JUICE.IDM.Entities.Search.create("My New Search");
```

Once you've created the query object, you can call methods on this object to define the basic settings for the query, as well as the condition groups and condition rows. The query structure you create with the JavaScript API follows the model of the JSON representation. After you've created the query object you append it to the QUERY request parameter.

The JavaScript example shown below illustrates how you use the JavaScript API to build a query:

```
function buildQuery3() {
  var newQuery = JUICE.IDM.Entities.Search.create("My New Search");
  newQuery.setFrom("user");
  var selAttrs = ["FirstName", "LastName"];
  newQuery.setSelects(selAttrs);
  var newCondGrp1 = newQuery.addConditionGroup();
  var newCondRow1_1 = newCondGrp1.addConditionRow();
  newCondRow1_1.setRowAttr("FirstName");
  newCondRow1_1.setRowRop("contains");
  newCondRow1_1.setRowVal("C");
  openSearchResults("QUERY=" + newQuery);
}
```

C.2.1 JavaScript API

This section provides reference documentation for the JavaScript API for searching entities in the Directory Abstraction Layer.

The following table describes the static methods for the JUICE.IDM.Entities.Search object:

Table C-5 Static methods for JUICE.IDM.Entities.Search

Method	Description
<Query> create(searchName)	Creates a new Query with the searchName
<void> load(uuid)	Loads a user's saved search with the uuid
<Query> get(uuid)	Returns the user's saved search with uuid as a Query
<String[]> getNames()	Returns the names of all the logged in user's saved searches
<String> getUUID(searchName)	Returns the uuid of the saved search with the searchName

The following table describes the methods for the Query object:

Table C-6 *Methods for the Query object*

Method	Description
<void> setKey(searchName)	Sets the searchName
<void> setFrom(defKey)	Sets the from entity-definition
<void> setSelects(attrKey[])	Sets the selects (optional, if using SearchListPortlet)
<void> setSearchScope(scp)	Sets the search scope (optional)
<void> setSearchRoot(rt)	Sets the search root (optional)
<void> setMaxPage(int)	Sets the max rows per page (optional)
<void> setMaxResults(int)	Sets the max rows in total (optional)
<void> setOrderBy(attrKey)	Sets the sort (optional)
<void> setPointer(int)	Sets the pagination offset (optional)
<void> setGroupLop(lop)	Sets the inter-group logical operator
<String> getKey()	Gets the searchName
<String> getFrom()	Gets the from entity-definition
<String> getSelects()	Gets the selects
<String> getSearchScope()	Gets the search scope
<String> getSearchRoot()	Gets the search root
<int> getMaxPage()	Gets the max rows per page
<int> getMaxResults()	Gets the max rows in total
<String> getOrderBy()	Gets the sort
<int> getPointer()	Gets the pagination offset
<String> getGroupLop()	Gets the inter-group logical operator
<int> nbConditionGroups	Returns the number of condition groups
<CondGroup> addConditionGroup	Creates and returns a new condition group (CondGroup object) appended to the query
<void> removeConditionGroup(i)	Removes the condition group at i
<CondGroup> getConditionGroup(i)	Returns the condition group at i

The following table describes the methods for the CondGroup object:

Table C-7 *Methods for the CondGroup object*

Method	Description
<void> setRowLop(lop)	Sets the intra-group logical operator
<String> getRowLop()	Gets the intra-group logical operator
<int> nbConditionRows()	Returns the number of condition rows

C.4 Retrieving all Saved Queries for the Current User

You can use the JavaScript API to retrieve all saved queries for the user who is currently logged on. To do this, you need to call the `getNames()` static method on the `JUICE.IDM.Entities.Search` object.

The following JavaScript example illustrates the procedure for retrieving all saved queries for the current user:

```
function query4GetSavedQueries() {
    var searchNames = JUICE.IDM.Entities.Search.getNames();
    var replaceDiv = document.getElementById("savedQueryNames");
    replaceDiv.innerHTML = searchNames;
}
```

C.5 Running an Existing Saved Query

You can use the JavaScript API to execute a saved query. Before you execute a saved query, you need to perform the following JavaScript statement to retrieve the saved queries (as described in the previous section):

```
JUICE.IDM.Entities.Search.getNames();
```

You need to call `getNames()` first, even if you know the name of the saved search you want to run.

After calling the `getNames()` function, you need to perform these steps to execute the saved search:

- 1 Call the `getUUID()` method to access the UUID associated with the search name.
- 2 Call the `load()` method on the `JUICE.IDM.Entities.Search` object to load the saved query with the UUID.
- 3 Call the `get()` method to retrieve the saved query structure.

All of these methods are static methods.

Once you have the query structure, you can use it to construct a QUERY request parameter.

The following JavaScript example illustrates the procedure for launching a saved query:

```
function runQuery4() {
    var textField = document.getElementById("savedQueryToRun");
    var queryName = textField.value;
    var queryUUID = JUICE.IDM.Entities.Search.getUUID(queryName);
    JUICE.IDM.Entities.Search.load(queryUUID);
    var myQuery = JUICE.IDM.Entities.Search.get(queryUUID);

    openSearchResults("QUERY=" + myQuery);
}
```

C.6 Performing a Search on All Searchable Attributes

You can use the JavaScript API to search all of the searchable attributes for an entity. This type of search only applies to attributes that have a type of string. Therefore, it does not work with DN, date, integer, boolean, and so forth.

To perform a search on all searchable attributes, you create a query object in the same manner that you would using other search techniques (as described above). Then you need to get the list of attributes for an entity definition by calling `JUICE.IDM.Definition.load()`. Once you have the list of

attributes, you need to verify that each attribute is a string and is searchable. For each attribute that is a string and is searchable, you can now add a condition row by calling the `addConditionRow()` method on the condition group object. When all condition rows have been added, you can execute the search.

The following JavaScript example illustrates how to perform a search on all searchable attributes.

```
function buildQuery5() {
    var searchStr = document.getElementById("query5Text").value;
    if (searchStr == "") {
        alert("Enter a search string in the text field.");
        return;
    }
    var newQuery = JUICE.IDM.Entities.Search.create("My New Search");
    var entDef = "user";
    newQuery.setFrom(entDef);
    var selAttrs = new Array();
    selAttrs.push("FirstName");
    selAttrs.push("LastName");
    newQuery.setSelects(selAttrs);
    var newCondGrp1 = newQuery.addConditionGroup();
    newCondGrp1.setRowLop("or");

    //get all the searchable attributes of entity-definition user that are type
    string (excludes DN, date, integer, boolean, etc)
    JUICE.IDM.Definitions.load(entDef);
    var attrKeys = JUICE.IDM.Definitions.getAttributeKeys(entDef);
    for (var i = 0; i < attrKeys.length; i++) {
        var attrDef = JUICE.IDM.Definitions.getAttribute(entDef, attrKeys[i]);
        var attrType = attrDef.getType();
        var searchable = attrDef.isSearchable();

        if (attrType == "String" && searchable ) {
            var newCondRow = newCondGrp1.addConditionRow();
            newCondRow.setRowAttr(attrKeys[i]);
            newCondRow.setRowRop("contains");
            newCondRow.setRowVal(searchStr);
        }
    }
    openSearchResults("QUERY=" + newQuery);
}
```

D Trouble Shooting

This section describes tips for working around common errors.

D.1 Permgen Space Error

You might encounter the following error when you redeploy the User Application:

```
11:32:20,194 ERROR [[PortalAggregator]] Servlet.service() for servlet
PortalAggregator threw exception java.lang.OutOfMemoryError: PermGen space
```

To avoid this error, either:

- ◆ Restart the JBoss server.

or

- ◆ Or, increase the PermSpace value by passing `-XX:MaxPermSize` to the Java virtual machine by means of `JAVA_OPTS` in the `start-jboss` script, for example:

```
-XX:MaxpermSize=128m
```

D.2 Email Notification Templates

If your email notification templates are displaying in a single language and not in the user's default locale as you expect, check to see what notification template is selected. You can select a default template or a localized version of the template. When you select a localized template, the language of the localized template is used regardless of the user's default language. When you select the default template (the template without a locale code), the email is in the user's default language (if the default is a supported language).

D.3 Org Chart and Guest Access

If you encounter an error like this at runtime, then you must modify the service definitions in the User Application WAR:

```
error: "an error occurred Control instantiation of JUICE.OrgChartCtrl failed
(Object doesn't support this property or method). Please contact your system
administrator. Detailed information can be found in the console." when accessing
the portlet in a browser.
```

D.4 Provisioning Notification

If the **Notify Other Users of these Changes** check box does not display on the following pages:

- ◆ Edit Availability

- ◆ My Proxy Assignments
- ◆ My Delegate Assignments
- ◆ Team Proxy Assignments
- ◆ Team Delegate Assignments
- ◆ Team Availability

Verify that Email Notification templates have been defined. You define them through the **Administration > RBPM Provisioning and Security > Delegation and Proxy**.

D.5 javax.naming.SizeLimitExceededException

If you encounter a `javax.naming.SizeLimitExceededException` when you use the **Administration > Page Admin > Set As Default**, you might have encountered a maximum size limit. You can modify this limit in the `PortalGroupPageDefaults` portlet settings in the `portlet.xml` as follows:

```
<portlet>
  <portlet-name>PortalGroupPageDefaults</portlet-name>
  <portlet-class>
com.novell.afw.portal.portlet.core.permission.PortalGroupPageDefaults
</portlet-class>
  <init-param>
    <name>MIN_CACHE_SIZE</name>
    <value>20</value>
  </init-param>
  <init-param>
    <name>MAX_CACHE_SIZE</name>
    <value>200</value>
  </init-param>
  <init-param>
    <name>PAC_MAX_RESULTS</name>
    <value>2000</value>
  </init-param>
  ...
</portlet>
```

If you have more than 200 groups and want to assign groups to the View permissions for the Page Admin tab, you also need to update the settings for the `PortalUserGroupSelection` portlet. Modify this limit in the `portlet.xml` as follows:

```

<portlet>
  <portlet-name>PortalUserGroupSelection</portlet-name>
  <portlet-class>
com.novell.afw.portal.portlet.core.permission.PortalUserGroupSelection
</portlet-class>
  <init-param>
    <name>MIN_CACHE_SIZE</name>
    <value>20</value>
  </init-param>
  <init-param>
    <name>MAX_CACHE_SIZE</name>
    <value>200</value>
  </init-param>
  <init-param>
    <name>PAC_MAX_RESULTS</name>
    <value>2000</value>
  </init-param>
  ...
</portlet>

```

Redeploy the User Application after you make your changes.

D.6 Linux Open Files Error

If you run the User Application on Linux, you might encounter a **Too Many Open Files Error**.

Linux allows 1024 open files for each process, but the User Application often requires more. NetIQ suggests increasing the number of open files to 4096 to avoid the **Too Many Open Files** error.

Use the `ulimit` command to increase the number of open files. There are some restrictions on `ulimit` for non-root users. Here is an example of how you can use the `ulimit` command to increase the number of open files to 4096 for a non-root user:

- 1 Log in as root.
- 2 Edit the file `/etc/security/limits.conf`. Add an entry for the user named `smith` and allow `nofile` up to 4096:

```
smith    hard    nofile   4096
```

- 3 Log in as user `smith` and pass 4096 to the `ulimit -n` command. You can issue the command again with no argument to see the current value:

```
smith@myhost:~> ulimit -n 4096
smith@myhost:~> ulimit -n
```

You might want to specify `ulimit` in the user environment or the `start-jboss` script so that the new value is always used.

25 Troubleshooting

The following sections contain information about troubleshooting different components of identity applications:

- ♦ [Section 25.1, “Using Log Files for Troubleshooting,” on page 607](#)
- ♦ [Section 25.2, “Troubleshooting Open XDAS Logging,” on page 621](#)
- ♦ [Section 25.3, “Troubleshooting Self Service Password Reset Issues,” on page 622](#)
- ♦ [Section 25.4, “Entitlement Configuration Error During Codemap Refresh,” on page 623](#)
- ♦ [Section 25.5, “Mismatch of Certificates Used by Identity Manager Engine and User Application Causes Code \(-9205\) Error in vnd.nds.stream,” on page 623](#)

25.1 Using Log Files for Troubleshooting

The following sections provide information about how to use log files for troubleshooting problems:

- ♦ [Section 25.1.1, “Virtual Data Access Logging,” on page 607](#)
- ♦ [Section 25.1.2, “When a Code Map Refresh Is Triggered,” on page 610](#)
- ♦ [Section 25.1.3, “When Multiple Users Try to Authenticate From Different Interfaces,” on page 611](#)
- ♦ [Section 25.1.4, “When an E-Mail Approval Notification is Not Delivered,” on page 612](#)
- ♦ [Section 25.1.5, “When a Role Is Requested,” on page 613](#)
- ♦ [Section 25.1.6, “When a Role Is Listed in Role Catalog,” on page 615](#)
- ♦ [Section 25.1.7, “Checking the Status of Database Schema Validation,” on page 618](#)
- ♦ [Section 25.1.8, “Determining if Liquibase Changeset Has Executed,” on page 618](#)
- ♦ [Section 25.1.9, “When Assigning a Resource to a User That Does Not Exist,” on page 620](#)
- ♦ [Section 25.1.10, “When Checking the Workflow Engine Heartbeat,” on page 620](#)

25.1.1 Virtual Data Access Logging

The Virtual Data Access (VDA) trace is logged to the `catalina.out` file when you look for an entity definition.

The VDA issues all of the identity applications LDAP queries for identity data such as entity definition and attributes on the Identity Vault. First it queries the information from the local cache residing on the identity applications server. If the information is not found in the local cache, it queries the Identity Vault. After locating the object, the identity applications read the attributes of the object and map the attributes with the entity definition in the local cache. When the entity definition matches, the identity applications send the data to the client. The trace looks similar to the following:

```
2015-08-29 16:05:05,389 [http-bio-8443-exec-10] DEBUG
com.novell.idm.nrf.util.CacheUtil- [RBPM] Role object was found in cache: cache-
key-nrf-config
```

2015-08-29 16:05:05,389 [http-bio-8443-exec-10] DEBUG
com.novell.idm.nrf.util.CacheUtil- [RBPM] Role object RETRIEVED from cache: cache-
key-nrf-config

2015-08-29 16:05:05,392 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM]
VDA.getEntityResultList

2015-08-29 16:05:05,392 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataModel- [RBPM]
VDM.getEntityDefinition(String, Locale):sys-nrf-navitem

2015-08-29 16:05:05,392 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.spi.vdata.util.CacheUtil- [RBPM] DAL Definition is null/not
found in cache: VDM_ENTITY_DEFINITION_sys-nrf-navitem, Is object in cache?false

2015-08-29 16:05:05,392 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.definition.VirtualDataDefinition- [RBPM] Entity
Definition found: sys-nrf-navitem

2015-08-29 16:05:05,393 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.spi.vdata.util.CacheUtil- [RBPM] DAL definition was found in
cache: VDD_ENTITY_ATTR_sys-nrf-navitem

2015-08-29 16:05:05,393 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.spi.vdata.util.CacheUtil- [RBPM] DAL Definition RETRIEVED from
cache: VDD_ENTITY_ATTR_sys-nrf-navitem

2015-08-29 16:05:05,393 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.spi.vdata.util.CacheUtil- [RBPM] PUT new DAL Definition into
cache: VDM_DEFINITION_ATTRIBUTE_LIST_sys-nrf-navitem

2017-08-29 16:05:05,394 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataModel- [RBPM]
VDM.getEntityDefinition(String, Locale):sys-nrf-navitem

2015-08-29 16:05:05,394 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM]
VDA.getEntityResultList query filter: (&(objectClass=nrfNavItem))

2015-08-29 16:05:05,439 [http-bio-8443-exec-10] DEBUG
com.novell.idm.security.ui.UIProvSecurityUtil- [RBPM] not Admin or Compliance tab,
so checkAccess with resource =
cn=WorkDashBoard,cn=NavItems,cn=UIConfig,cn=AppConfig,cn=UserApplication,cn=driver
set,ou=idm,ou=services,o=system

2015-08-29 16:05:05,443 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataModel- [RBPM]
VDM.getEntityDefinition(String, Locale):sys-nrf-user

2015-08-29 16:05:05,444 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.definition.VirtualDataDefinition- [RBPM] Entity
Definition found: sys-nrf-user

2015-08-29 16:05:05,444 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.spi.vdata.util.CacheUtil- [RBPM] PUT new DAL Definition into
cache: VDM_ENTITY_DEFINITION_sys-nrf-user

2015-08-29 16:05:05,444 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] VDA.getEntity:
cn=mytestuser,dc=data

2015-08-29 16:05:05,444 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.spi.vdata.util.CacheUtil- [RBPM] DAL definition was found in
cache: VDD_ENTITY_ATTR_sys-nrf-user

2015-08-29 16:05:05,444 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.spi.vdata.util.CacheUtil- [RBPM] DAL Definition RETRIEVED from
cache: VDD_ENTITY_ATTR_sys-nrf-user

2015-08-29 16:05:05,445 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.spi.vdata.util.CacheUtil- [RBPM] PUT new DAL Definition into
cache: VDM_DEFINITION_ATTRIBUTE_LIST_sys-nrf-user

2015-08-29 16:05:05,446 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] VDA.getLdapAttributes
Attributes and values

2015-08-29 16:05:05,478 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] Attribute ID: mail

2015-08-29 16:05:05,478 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM]
mytestuser@acme.com

2015-08-29 16:05:05,478 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] Attribute ID:
modifyTimestamp

2015-08-29 16:05:05,478 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] 20150716124158Z

2015-08-29 16:05:05,478 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] Attribute ID:
givenName

2015-08-29 16:05:05,478 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] mytestuser

2015-08-29 16:05:05,478 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] Attribute ID:
objectClass

2015-08-29 16:05:05,478 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] inetOrgPerson

2015-08-29 16:05:05,478 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] organizationalPerson

2015-08-29 16:05:05,478 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] Person

2015-08-29 16:05:05,478 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] ndsLoginProperties

2015-08-29 16:05:05,478 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] Top

2015-08-29 16:05:05,478 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] srvprvEntityAux

2015-08-29 16:05:05,478 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] Attribute ID: sn

2015-08-29 16:05:05,478 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] mytestuser

```

2015-08-29 16:05:05,479 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] Attribute ID: cn

2015-08-29 16:05:05,479 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] VDA.checking if
object instance contains the required objectClass per DAL definition: sys-nrf-user

2015-08-29 16:05:05,479 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] VDA.does contain
required (search=true or auxilliary=false) objectClass:inetOrgPerson

2015-08-29 16:05:05,479 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] VDA.object instance
is correct type

2015-08-29 16:05:05,479 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataModel- [RBPM]
VDM.getEntityDefinition(String, Locale):sys-nrf-user

2015-08-29 16:05:05,479 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.spi.vdata.util.CacheUtil- [RBPM] DAL Definition RETRIEVED from
cache: VDM_ENTITY_DEFINITION_sys-nrf-user

2015-08-29 16:05:05,479 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] VDA.getEntity:
cn=mytestuser,dc=data

2015-08-29 16:05:05,480 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.spi.vdata.util.CacheUtil- [RBPM] DAL Definition RETRIEVED from
cache: VDM_DEFINITION_ATTRIBUTE_LIST_sys-nrf-user

2015-08-29 16:05:05,481 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] VDA.getLdapAttributes
Attributes and values

2015-08-29 16:05:05,481 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] Attribute ID: mail

2015-08-29 16:05:05,481 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM]
mytestuser@acme.com

2015-08-29 16:05:05,481 [http-bio-8443-exec-10] DEBUG
com.novell.srvprv.impl.vdata.model.VirtualDataAccess- [RBPM] 20150716124158Z

```

When an entity model is changed, you must clear the VDA cache for the changes to take effect. For example, entity changes occur when a new attributes is added or the existing attributes are modified or removed.

25.1.2 When a Code Map Refresh Is Triggered

When you initiate a code map refresh cycle, sometimes the refresh cycle is not successful. When this occurs, the trace prints messages similar to the following:

```

2015-11-14 12:28:12,045 [Timer-1] INFO com.novell.idm.nrf.service.CodeMapEngine-
[RBPM] Building the Entitlementment CODE MAP tables...

2015-11-14 12:28:12,450 [Timer-1] ERROR com.novell.idm.nrf.service.CodeMapEngine-
[RBPM] Unable to complete the CODE MAP refresh for entitlement:
cn=exchangemailbox,cn=ad,cn=dset,ou=idm,o=system.

```

```
2015-11-14 12:28:12,454 [Timer-1] INFO com.novell.idm.nrf.service.CodeMapEngine-
[RBPM] CODE MAP refresh on entitlement:
cn=exchangemailbox,cn=ad,cn=dset,ou=idm,o=system, processed, next refresh time:
20161115122812-0500.
```

```
2015-11-14 12:28:12,646 [Timer-1] ERROR com.novell.idm.nrf.service.CodeMapEngine-
[RBPM] Unable to complete the CODE MAP refresh for entitlement:
cn=group,cn=ad,cn=dset,ou=idm,o=system.
```

```
2015-11-14 12:28:12,835 [Timer-1] ERROR com.novell.idm.nrf.service.CodeMapEngine-
[RBPM] Unable to complete the CODE MAP refresh for entitlement:
cn=useraccount,cn=ad,cn=dset,ou=idm,o=system.
```

The first entry indicates that code map table refresh action is being initiated. The second entry specifies that it is unable to refresh the code map table for `cn=exchangemailbox,cn=ad,cn=dset,ou=idm,o=system` entitlement. The next line has the time interval when the entitlement will be refreshed. The next two lines specify that code map table was not refreshed for `cn=group,cn=ad,cn=dset,ou=idm,o=system` and `cn=useraccount,cn=ad,cn=dset,ou=idm,o=system` entitlements.

The code map refresh process can fail when either connected system or the Identity Vault is not up at the time of obtaining entitlement information. The trace logs the actual reason for failure.

25.1.3 When Multiple Users Try to Authenticate From Different Interfaces

OSP supports the OAuth2 specification and requires an LDAP authentication server. By default, Identity Manager uses Identity Vault (eDirectory) as an authentication server. When multiple users try to log in to OSP from different user interfaces of the identity applications, the users are redirected to the default landing page upon a successful login. When the access token expires within the login session, OSP validates the token and refreshes the session by generating a new access token without the user's involvement. Otherwise, it directs the user to the login page. Such a trace looks similar to the following:

```
2015-03-08 06:14:28,509 [http-bio-8443-exec-801] DEBUG
com.netiq.idm.auth.oauth.OAuthFilter- [RBPM] Original request going to RBPM is:
https://www.snet.acme.com:443/IDMProv/
workDashboard.do?apwaLeftNavItem=JSP_MENU_TASKS
```

```
2015-03-08 06:15:49,289 [http-bio-8443-exec-816] DEBUG
com.netiq.idm.auth.oauth.OAuthFilter- [RBPM] Original request going to RBPM is:
https://www.snet.acme.com:443/IDMProv/
```

```
2015-03-08 06:15:50,334 [http-bio-8443-exec-815] INFO
com.netiq.idm.auth.oauth.OAuthFilter- [RBPM] SSO Header issued by SSO Filter oauth
for User cn=Rob.Andrews,ou=Active,ou=People,o=acme.
```

```
2015-03-08 06:15:50,354 [http-bio-8443-exec-815] INFO
com.novell.common.auth.saml.AuthTokenGenerator- [RBPM] SAML Token is issued by the
request from SSO filter oauth
```

```
2015-03-08 06:15:50,414 [http-bio-8443-exec-815] INFO
com.novell.pwdmgt.util.PasswordHelper- [RBPM] [Login_Success]
cn=David.Scully,ou=Active,ou=People,o=acme successfully logged in.
```

```
2015-03-08 06:17:53,520 [http-bio-8443-exec-819] DEBUG
com.netiq.idm.auth.oauth.OAuthFilter- [RBPM] Original request going to RBPM is:
https://www.snet.acme.com:443/IDMProv/
```

```
2015-03-08 06:17:55,194 [http-bio-8443-exec-811] INFO
com.netiq.idm.auth.oauth.OAuthFilter- [RBPM] SSO Header issued by SSO Filter oauth
for User cn=neil.smith,ou=Active,ou=People,o=acme.
```

```
2015-03-08 06:17:55,204 [http-bio-8443-exec-811] INFO
com.novell.common.auth.saml.AuthTokenGenerator- [RBPM] SAML Token is issued by the
request from SSO filter oauth
```

```
2015-03-08 06:17:55,234 [http-bio-8443-exec-811] INFO
com.novell.pwdmgt.util.PasswordHelper- [RBPM] [Login_Success]
cn=neil.smith,ou=Active,ou=People,o=acme successfully logged in.
```

```
2015-03-08 06:20:56,616 [http-bio-8443-exec-813] DEBUG
com.netiq.idm.auth.oauth.OAuthFilter- [RBPM] Original request going to RBPM is:
https://www.snet.acme.com:443/IDMProv/
```

```
2015-03-08 06:21:02,129 [http-bio-8443-exec-823] INFO
com.netiq.idm.auth.oauth.OAuthFilter- [RBPM] SSO Header issued by SSO Filter oauth
for User cn=dean.gardner,ou=Active,ou=People,o=acme.
```

```
2015-03-08 06:21:02,149 [http-bio-8443-exec-823] INFO
com.novell.common.auth.saml.AuthTokenGenerator- [RBPM] SAML Token is issued by the
request from SSO filter oauth
```

```
2015-03-08 06:21:02,216 [http-bio-8443-exec-823] INFO
com.novell.pwdmgt.util.PasswordHelper- [RBPM] [Login_Success]
cn=dean.gardner,ou=Active,ou=People,o=acme successfully logged in.
```

```
2015-03-08 06:24:28,626 [http-bio-8443-exec-830] DEBUG
com.netiq.idm.auth.oauth.OAuthFilter- [RBPM] Original request going to RBPM is:
https://www.snet.acme.com:443/IDMProv/
workDashboard.do?apwaLeftNavItem=JSP_MENU_TASKS
```

```
2015-03-08 06:24:40,547 [http-bio-8443-exec-814] WARN
com.netiq.idm.auth.oauth.OAuthManager- [RBPM] Token validation failed. HTTP status
code: 401 Detail message from authentication server: The access token is expired.
```

This trace indicates that the user is accessing the application after some idle time. The last message indicates that the token has expired. When the user tried to log in again, the token failed the validation and as a result the user cannot be logged in.

25.1.4 When an E-Mail Approval Notification is Not Delivered

Sometimes an e-mail notification is not delivered due to errors in the connection between the client and the mail server. When this occurs, the trace looks similar to the following:

```
2015-03-08 07:42:41,575 [NOTIFICATION THREAD] ERROR
com.novell.soa.notification.impl.NotificationThread- [RBPM] Error sending email.
com.novell.soa.notification.impl.NotificationException: Error sending email.
at com.novell.soa.notification.impl.MailEngine.send(MailEngine.java:347)
at
com.novell.soa.notification.impl.NotificationThread.run(NotificationThread.java:96
)
Caused by: javax.mail.MessagingException: 421 Too many errors on this connection--
-closing

at com.sun.mail.smtp.SMTPTransport.issueCommand(SMTPTransport.java:879)
at com.sun.mail.smtp.SMTPTransport.mailFrom(SMTPTransport.java:599)
at com.sun.mail.smtp.SMTPTransport.sendMessage(SMTPTransport.java:319)
at com.novell.soa.notification.impl.MailEngine.send(MailEngine.java:344)
... 1 more
```

When an e-mail approval notification is not delivered, the first step should be to look at the logs and determine whether the connection is proper, mail server is running and accessible. Sometimes the e-mail fails to comply with e-mail template and fails to deliver.

25.1.5 When a Role Is Requested

When a role is requested in the identity applications, Identity Manager creates the role object in the Identity Vault. The Role and Resource Service driver checks users for assigning this role, and then provisions the role to the assigned users. When this occurs, the trace prints messages similar to the following:

```
2015-03-08 08:43:10,660 [pool-2-thread-5] INFO
com.novell.idm.nrf.service.RoleManagerService- [RBPM] [Role_Request] Requested by
cn=David.Scully,ou=Active,ou=People,o=acme, Target DN:
CN=Kaitlin.Demore,OU=active,OU=People,O=acme, Source
DN:cn=PennDOT_Vehicle_Certification,cn=Application
Access,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm
361,ou=services,o=acme, Request DN:cn=20160308084310-
15da49b28ddf4ee1b7d71b4ce220c080-
0,cn=Requests,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm361,ou=services,
o=acme, Request Category: 10, Request Status: 0, Original Request Status: 0,
Correlation ID: UserApp#UserStartWorkflow#082080ec-5b10-498f-b87d-28825ab63278
```

```
2015-03-08 08:43:10,669 [pool-2-thread-5] INFO
com.novell.idm.nrf.service.RoleManagerService- [RBPM] [Role_Request] Requested by
cn=David.Scully,ou=Active,ou=People,o=acme, Target DN:
CN=Kaitlin.Demore,OU=active,OU=People,O=acme, Source DN:cn=PennDOT History and
Photos Users,cn=Application
Access,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm
361,ou=services,o=acme, Request DN:cn=20160308084310-
3c5b20b79cc046bb8267a41cad88a96a-
1,cn=Requests,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm361,ou=services,
o=acme, Request Category: 10, Request Status: 0, Original Request Status: 0,
Correlation ID: UserApp#UserStartWorkflow#082080ec-5b10-498f-b87d-28825ab63278
```

```
2015-03-08 08:43:10,678 [pool-2-thread-5] INFO
com.novell.idm.nrf.service.RoleManagerService- [RBPM] [Role_Request] Requested by
cn=David.Scully,ou=Active,ou=People,o=acme, Target DN:
CN=Kaitlin.Demore,OU=active,OU=People,O=acme, Source DN:cn=JWL Eligible
Agency,cn=Application
Access,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm
361,ou=services,o=acme, Request DN:cn=20160308084310-
fc24ac874aca4fc8b1db0e1d7662d9b3-
2,cn=Requests,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm361,ou=services,
o=acme, Request Category: 10, Request Status: 0, Original Request Status: 0,
Correlation ID: UserApp#UserStartWorkflow#082080ec-5b10-498f-b87d-28825ab63278
```

```
2015-03-08 08:43:10,712 [pool-2-thread-5] INFO
com.novell.idm.nrf.service.RoleManagerService- [RBPM] [Role_Request] Requested by
cn=David.Scully,ou=Active,ou=People,o=acme, Target DN:
CN=Kaitlin.Demore,OU=active,OU=People,O=acme, Source DN:cn=DPW Recipient Address
Eligible Agency,cn=Application
Access,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm
361,ou=services,o=acme, Request DN:cn=20160308084310-
411abble8f6f488182c37c8629275245-
3,cn=Requests,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm361,ou=services,
o=acme, Request Category: 10, Request Status: 0, Original Request Status: 0,
Correlation ID: UserApp#UserStartWorkflow#082080ec-5b10-498f-b87d-28825ab63278
```

2015-03-08 08:43:10,737 [pool-2-thread-5] INFO
com.novell.idm.nrf.service.RoleManagerService- [RBPM] [Role_Request] Requested by
cn=David.Scully,ou=Active,ou=People,o=acme, Target DN:
CN=Kaitlin.Demore,OU=active,OU=People,O=acme, Source DN:cn=cj-users,cn=Application
Access,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm
361,ou=services,o=acme, Request DN:cn=20160308084310-
211a591e09b04fbbb195fb14d7f4df07-
4,cn=Requests,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm361,ou=services,
o=acme, Request Category: 10, Request Status: 0, Original Request Status: 0,
Correlation ID: UserApp#UserStartWorkflow#082080ec-5b10-498f-b87d-28825ab63278

2015-03-08 08:43:10,790 [pool-2-thread-5] INFO
com.novell.idm.nrf.service.RoleManagerService- [RBPM] [Role_Request] Requested by
cn=David.Scully,ou=Active,ou=People,o=acme, Target DN:
CN=Kaitlin.Demore,OU=active,OU=People,O=acme, Source DN:cn=JTS Eligible
Agency,cn=Application
Access,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm
361,ou=services,o=acme, Request DN:cn=20160308084310-
5339699630814a91ac44530a244a02ba-
5,cn=Requests,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm361,ou=services,
o=acme, Request Category: 10, Request Status: 0, Original Request Status: 0,
Correlation ID: UserApp#UserStartWorkflow#082080ec-5b10-498f-b87d-28825ab63278

2015-03-08 08:43:10,799 [pool-2-thread-5] INFO
com.novell.idm.nrf.service.RoleManagerService- [RBPM] [Role_Request] Requested by
cn=David.Scully,ou=Active,ou=People,o=acme, Target DN:
CN=Kaitlin.Demore,OU=active,OU=People,O=acme, Source DN:cn=Sentencing Guidelines
Eligible Agency,cn=Application
Access,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm
361,ou=services,o=acme, Request DN:cn=20160308084310-
df2398cf36a042f0ac2241e693efb93c-
6,cn=Requests,cn=RoleConfig,cn=AppConfig,cn=UserApplication,cn=idm361,ou=services,
o=acme, Request Category: 10, Request Status: 0, Original Request Status: 0,
Correlation ID: UserApp#UserStartWorkflow#082080ec-5b10-498f-b87d-28825ab63278

2015-03-08 08:43:10,800 [pool-2-thread-5] INFO
com.novell.soa.af.impl.LogEvent- [RBPM] [Role_Request_Submitted] Initiated by
cn=David.Scully,ou=Active,ou=People,o=acme, Process ID:
95f25f5f31224efcab1611fe0fc2471f, Process Name:
cn=newuserinvitation,cn=RequestDefs,cn=AppConfig,cn=UserApplication,cn=idm361,ou=s
ervices,o=acme, Activity: Activity6, Recipient:
CN=Kaitlin.Demore,OU=active,OU=People,O=acme, Correlation
ID:UserApp#UserStartWorkflow#082080ec-5b10-498f-b87d-28825ab63278, Submitted
Request:<?xml version="1.0" encoding="UTF-8"?><wfRoleRequest>
<attr name="sod-override-request">
<value>true</value>
</attr>
<attr name="target">
<value>CN=Kaitlin.Demore,OU=active,OU=People,O=acme</value>
</attr>
<attr name="action">
<value>GRANT</value>
</attr>
<attr name="targetType">
<value>USER</value>
</attr>
<attr name="roles">
<value>cn=PennDOT_Vehicle_Certification,cn=Application
Access,cn=Level10,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=UserApplication,ou=ser
vices,o=acme</value>
<value>cn=PennDOT History and Photos Users,cn=Application

```

Access, cn=Level10, cn=RoleDefs, cn=RoleConfig, cn=AppConfig, cn=UserApplication, ou=ser
vices, o=acme</value>
<value>cn=JWL Eligible Agency, cn=Application
Access, cn=Level10, cn=RoleDefs, cn=RoleConfig, cn=AppConfig, cn=UserApplication, ou=ser
vices, o=acme</value>
<value>cn=DPW Recipient Address Eligible Agency, cn=Application
Access, cn=Level10, cn=RoleDefs, cn=RoleConfig, cn=AppConfig, cn=UserApplication, ou=ser
vices, o=acme</value>
<value>cn=cj-users, cn=Application
Access, cn=Level10, cn=RoleDefs, cn=RoleConfig, cn=AppConfig, cn=UserApplication, ou=ser
vices, o=acme</value>
<value>cn=JTS Eligible Agency, cn=Application
Access, cn=Level10, cn=RoleDefs, cn=RoleConfig, cn=AppConfig, cn=UserApplication, ou=ser
vices, o=acme</value>
<value>cn=Sentencing Guidelines Eligible Agency, cn=Application
Access, cn=Level10, cn=RoleDefs, cn=RoleConfig, cn=AppConfig, cn=UserApplication, ou=ser
vices, o=acme</value>
</attr>
<attr name="correlationId">
<value>UserApp#UserStartWorkflow#082080ec-5b10-498f-b87d-28825ab63278</value>
</attr>
<attr name="nrfRequest">
<value>cn=20160308084310-15da49b28ddf4eelb7d71b4ce220c080-
0, cn=Requests, cn=RoleConfig, cn=AppConfig, cn=UserApplication, ou=services, o=acme</
value>
<value>cn=20160308084310-3c5b20b79cc046bb8267a41cad88a96a-
1, cn=Requests, cn=RoleConfig, cn=AppConfig, cn=UserApplication, ou=services, o=acme</
value>
<value>cn=20160308084310-fc24ac874aca4fc8bldb0e1d7662d9b3-
2, cn=Requests, cn=RoleConfig, cn=AppConfig, cn=UserApplication, ou=services, o=acme</
value>
<value>cn=20160308084310-411abb1e8f6f488182c37c8629275245-
3, cn=Requests, cn=RoleConfig, cn=AppConfig, cn=UserApplication, ou=services, o=acme</
value>
<value>cn=20160308084310-211a591e09b04fbbb195fb14d7f4df07-
4, cn=Requests, cn=RoleConfig, cn=AppConfig, cn=UserApplication, ou=services, o=acme</
value>
<value>cn=20160308084310-5339699630814a91ac44530a244a02ba-
5, cn=Requests, cn=RoleConfig, cn=AppConfig, cn=UserApplication, ou=services, o=acme</
value>
<value>cn=20160308084310-df2398cf36a042f0ac2241e693efb93c-
6, cn=Requests, cn=RoleConfig, cn=AppConfig, cn=UserApplication, ou=services, o=acme</
value>
</attr>
</wfRoleRequest>

```

```

2015-03-08 08:43:10,880 [pool-2-thread-5] INFO com.novell.soa.af.impl.LogEvent-
[RBPM] [Workflow_Forwarded] Initiated by System, Process ID:
95f25f5f31224efcab1611fe0fc2471f, Process Name:
cn=newuserinvitation, cn=RequestDefs, cn=AppConfig, cn=UserApplication, ou=services, o=
acme:205, Activity: Activity6, Recipient:
cn=Nancy.Wilmer, ou=Active, ou=People, o=acme

```

25.1.6 When a Role Is Listed in Role Catalog

When you issue a request to display a role in Role Catalog, the identity applications obtain the role object from the cache and then display the role information in Role Catalog.

2015-09-22 09:28:26,495 [DEBUG] RuntimeAuthorizationManagerService [RBPM]
Authorized result for:
cn=roleAdmin,cn=System,cn=Level20,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=User
Application Driver,cn=driverset1,o=system:role:read=true

2015-09-22 09:28:26,495 [DEBUG] CacheUtil [RBPM] Role object was found in cache:
cache-key-nrf-config

2015-09-22 09:28:26,495 [DEBUG] CacheUtil [RBPM] Role object RETRIEVED from cache:
cache-key-nrf-config

2015-09-22 09:28:26,509 [DEBUG] RuntimeAuthDelegatorBase [RBPM] Time to calculate
sel service access rights: 0milliseconds.

2015-09-22 09:28:26,509 [DEBUG] RuntimeAuthorizationManagerService [RBPM]
Authorized result for:
cn=secAdmin,cn=System,cn=Level20,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=User
Application Driver,cn=driverset1,o=system:role:read=true

2015-09-22 09:28:26,510 [DEBUG] CacheUtil [RBPM] Role object was found in cache:
cache-key-nrf-config

2015-09-22 09:28:26,510 [DEBUG] CacheUtil [RBPM] Role object RETRIEVED from cache:
cache-key-nrf-config

2015-09-22 09:28:26,522 [DEBUG] RuntimeAuthDelegatorBase [RBPM] Time to calculate
sel service access rights: 0milliseconds.

2015-09-22 09:28:26,522 [DEBUG] RuntimeAuthorizationManagerService [RBPM]
Authorized result for:
cn=resourceManager,cn=System,cn=Level20,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=
User Application Driver,cn=driverset1,o=system:role:read=true

2015-09-22 09:28:26,522 [DEBUG] CacheUtil [RBPM] Role object was found in cache:
cache-key-nrf-config

2015-09-22 09:28:26,522 [DEBUG] CacheUtil [RBPM] Role object RETRIEVED from cache:
cache-key-nrf-config

2015-09-22 09:28:26,533 [DEBUG] RuntimeAuthDelegatorBase [RBPM] Time to calculate
sel service access rights: 0milliseconds.

2015-09-22 09:28:26,533 [DEBUG] RuntimeAuthorizationManagerService [RBPM]
Authorized result for:
cn=roleManager,cn=System,cn=Level20,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=User
Application Driver,cn=driverset1,o=system:role:read=true

2015-09-22 09:28:26,534 [DEBUG] CacheUtil [RBPM] Role object was found in cache:
cache-key-nrf-config

2015-09-22 09:28:26,534 [DEBUG] CacheUtil [RBPM] Role object RETRIEVED from cache:
cache-key-nrf-config

2015-09-22 09:28:26,545 [DEBUG] RuntimeAuthDelegatorBase [RBPM] Time to calculate
sel service access rights: 0milliseconds.

2015-09-22 09:28:26,545 [DEBUG] RuntimeAuthorizationManagerService [RBPM]
Authorized result for:
cn=resourceAdmin,cn=System,cn=Level20,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=Us
er Application Driver,cn=driverset1,o=system:role:read=true

2015-09-22 09:28:26,545 [DEBUG] CacheUtil [RBPM] Role object was found in cache:
cache-key-nrf-config


```

2015-09-22 09:28:26,545 [DEBUG] CacheUtil [RBPM] Role object RETRIEVED from cache:
cache-key-nrf-config

2015-09-22 09:28:26,557 [DEBUG] RuntimeAuthDelegatorBase [RBPM] Time to calculate
sel service access rights: 0milliseconds.

2015-09-22 09:28:26,557 [DEBUG] RuntimeAuthorizationManagerService [RBPM]
Authorized result for:
cn=provAdmin,cn=System,cn=Level20,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=User
Application Driver,cn=driverset1,o=system:role:read=true

2015-09-22 09:28:26,557 [DEBUG] CacheUtil [RBPM] Role object was found in cache:
cache-key-nrf-config

2015-09-22 09:28:26,557 [DEBUG] CacheUtil [RBPM] Role object RETRIEVED from cache:
cache-key-nrf-config

2015-09-22 09:28:26,567 [DEBUG] RuntimeAuthDelegatorBase [RBPM] Time to calculate
sel service access rights: 0milliseconds.

2015-09-22 09:28:26,567 [DEBUG] RuntimeAuthorizationManagerService [RBPM]
Authorized result for:
cn=reportAdmin,cn=System,cn=Level20,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=User
Application Driver,cn=driverset1,o=system:role:read=true

2015-09-22 09:28:26,567 [DEBUG] CacheUtil [RBPM] Role object was found in cache:
cache-key-nrf-config

2015-09-22 09:28:26,567 [DEBUG] CacheUtil [RBPM] Role object RETRIEVED from cache:
cache-key-nrf-config

2015-09-22 09:28:26,582 [DEBUG] RuntimeAuthDelegatorBase [RBPM] Time to calculate
sel service access rights: 0milliseconds.

2015-09-22 09:28:26,582 [DEBUG] RuntimeAuthorizationManagerService [RBPM]
Authorized result for:
cn=provManager,cn=System,cn=Level20,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,cn=User
Application Driver,cn=driverset1,o=system:role:read=true

2015-09-22 09:28:26,582 [DEBUG] JuiceHelper [RBPM] Kicked out of main loop with:
openR=-1 closeR=-1 i=0 idx=-1

2015-09-22 09:28:26,582 [DEBUG] JuiceHelper [RBPM] Setting RsMeta from cache:
METAListRoles_defaultDescription&*-Name&*-enStatus_cn=uaadmin,ou=sa,o=data

2015-09-22 09:28:26,583 [DEBUG] VirtualDataModel [RBPM]
VDM.getEntityDefinition(String, Locale):sys-nrf-role

2015-09-22 09:28:26,583 [DEBUG] CacheUtil [RBPM] DAL Definition RETRIEVED from
cache: VDM_ENTITY_DEFINITION_sys-nrf-role

2015-09-22 09:28:26,583 [DEBUG] CacheUtil [RBPM] DAL Definition RETRIEVED from
cache: VDM_DEFINITION_ATTRIBUTE_LIST_sys-nrf-role

2015-09-22 09:28:26,583 [DEBUG] CacheUtil [RBPM] Role object was found in cache:
cache-key-nrf-config

2015-09-22 09:28:26,583 [DEBUG] CacheUtil [RBPM] Role object RETRIEVED from cache:
cache-key-nrf-config

```

The first log entry is the request to find the role object from the cache. The second log entry is the response that is returned, and it indicates that the role object was found. The object is then read and displayed in Role Catalog.

25.1.7 Checking the Status of Database Schema Validation

The identity applications database schema is validated when the administrator starts the identity applications server. The trace prints messages similar to the following:

```
2015-09-22 09:39:41,363 [INFO] DatabaseSchemaUpdate [RBPM] Connecting to PostgreSQL
version 9.4.10.

2015-09-22 09:39:41,375 [INFO] DatabaseSchemaUpdate [RBPM] Checking for database
schema

2015-09-22 09:39:41,401 [INFO] DatabaseSchemaUpdate [RBPM] Checking schema for
table af_resource_request_status....found

2015-09-22 09:39:41,422 [INFO] DatabaseSchemaUpdate [RBPM] Checking schema for
table af_role_request_status....found

2015-09-22 09:39:41,481 [INFO] DatabaseSchemaUpdate [RBPM] Checking schema for
table aactivity....found

2015-09-22 09:39:41,501 [INFO] DatabaseSchemaUpdate [RBPM] Checking schema for
table aactivitytimertasks....found

2015-09-22 09:39:41,576 [INFO] DatabaseSchemaUpdate [RBPM] Checking schema for
table afbranch....found

2015-09-22 09:39:41,612 [INFO] DatabaseSchemaUpdate [RBPM] Checking schema for
table afcomment....found

2015-09-22 09:39:41,651 [INFO] DatabaseSchemaUpdate [RBPM] Checking schema for
table afdocument....missing
```

The first log entry indicates that a connection to the database is initiated. The second log entry specifies that the database schema is being validated. The subsequent entries specify that the presence of database tables is being checked. The last trace entry indicates that `afdocument` table is not found.

25.1.8 Determining if Liquibase Changeset Has Executed

The Liquibase framework validates the database schema against the `changelog.xml` file, which contains the database changes. Liquibase first verifies whether all the tables are present in the schema. If any of the changesets was not executed, it indicates incomplete schema update and also identifies the changesets. Any schema changes made in the User Application are updated in the database when the User Application server is started and `com.netiq.idm.create-db-on-startup` flag is set to `true` in the `ism-configuration` properties file located by default in the `/netiq/idm/apps/tomcat/conf` directory. The database compares the existing schema with target schema and then updates the database schema. If the flag is not set, it reports the following message:

One or more required tables are missing. Check log for messages indicating tables that were not found.

When Liquibase validates the database schema, it generates log entries similar to the following:

```
2015-09-22 09:39:47,693 [INFO] liquibase liquibase: Reading from
public.databasechangelog

2015-09-22 09:39:49,133 [INFO] liquibase liquibase: Reading from
public.databasechangelog

2015-09-22 09:39:49,427 [INFO] liquibase liquibase: Reading from
public.databasechangelog
```

```

2015-09-22 09:39:49,551 [INFO] liquibase liquibase: Reading from
public.databasechangelog

2015-09-22 09:39:50,095 [INFO] liquibase liquibase: Reading from
public.databasechangelog

2015-09-22 09:39:50,403 [INFO] liquibase liquibase: Reading from
public.databasechangelog

2015-09-22 09:39:50,514 [ERROR] EboPortalBootServlet [RBPM] Unexpected Runtime
Exception initializing servlet
java.lang.RuntimeException: com.netiq.common.i18n.LocalizedRuntimeException:
Schema is invalid. One or more required tables are missing. Check log for messages
indicating tables that were not found.
    at com.sssw.fw.servlet.EboBootServlet.init(EboBootServlet.java:115)
    at
com.sssw.portal.servlet.EboPortalBootServlet.init(EboPortalBootServlet.java:62)
    at javax.servlet.GenericServlet.init(GenericServlet.java:158)
    at
org.apache.catalina.core.StandardWrapper.initServlet(StandardWrapper.java:1183)
    at
org.apache.catalina.core.StandardWrapper.loadServlet(StandardWrapper.java:1099)
    at org.apache.catalina.core.StandardWrapper.load(StandardWrapper.java:989)
    at
org.apache.catalina.core.StandardContext.loadOnStartup(StandardContext.java:4913)
    at
org.apache.catalina.core.StandardContext.startInternal(StandardContext.java:5223)
    at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:150)
    at
org.apache.catalina.core.ContainerBase.addChildInternal(ContainerBase.java:752)
    at org.apache.catalina.core.ContainerBase.addChild(ContainerBase.java:728)
    at org.apache.catalina.core.StandardHost.addChild(StandardHost.java:734)
    at org.apache.catalina.startup.HostConfig.deployWAR(HostConfig.java:952)
    at
org.apache.catalina.startup.HostConfig$DeployWar.run(HostConfig.java:1823)
    at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
    at java.util.concurrent.FutureTask.run(FutureTask.java:266)
    at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
    at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
    at java.lang.Thread.run(Thread.java:745)
Caused by: com.netiq.common.i18n.LocalizedRuntimeException: Schema is invalid. One
or more required tables are missing. Check log for messages indicating tables that
were not found.
    at
com.novell.soa.persist.DatabaseSchemaUpdate.validateDatabaseSchema(DatabaseSchemaU
pdate.java:245)
    at com.sssw.fw.servlet.EboBootServlet.init(EboBootServlet.java:99) ... 18
more

```

```

Sep 22, 2015 9:39:50 AM org.apache.catalina.core.StandardContext loadOnStartup
SEVERE: Servlet [PortalAggregator] in web application [/IDMProv] threw load()
exception java.lang.NullPointerException at
com.sssw.portal.manager.EboPortalManager.<init>(EboPortalManager.java:179)

```

You must restart the application server to apply the changes. You cannot bring up the identity applications until the schema validation succeeds.

25.1.9 When Assigning a Resource to a User That Does Not Exist

If a user no longer exists in the system, and a request is issued to assign a resource to that user, the trace records messages similar to the following:

```
2015-09-22 11:50:53,605 [DEBUG] DataItemEvaluator [RBPM] result: Add Resource To
User - Laptop

2015-09-22 11:50:53,607 [ERROR] VirtualDataAccess [RBPM] Error occurred checking
the object type for: cn=rocio,ou=users,o=data
javax.naming.NameNotFoundException: [LDAP: error code 32 - NDS error: no such entry
(-601)]; remaining name 'cn=rocio,ou=users,o=data'
    at com.sun.jndi.ldap.LdapCtx.mapErrorCode(LdapCtx.java:3161)
    at com.sun.jndi.ldap.LdapCtx.processReturnCode(LdapCtx.java:3082)
    at com.sun.jndi.ldap.LdapCtx.processReturnCode(LdapCtx.java:2888)
    at com.sun.jndi.ldap.LdapCtx.c_getAttributes(LdapCtx.java:1329)
    at
com.sun.jndi.toolkit.ctx.ComponentDirContext.p_getAttributes(ComponentDirContext.j
ava:235)
    at
com.sun.jndi.toolkit.ctx.PartialCompositeDirContext.getAttributes(PartialComposite
DirContext.java:141)
    at
com.sun.jndi.toolkit.ctx.PartialCompositeDirContext.getAttributes(PartialComposite
DirContext.java:129)
    at sun.reflect.GeneratedMethodAccessor432.invoke(Unknown Source)
    at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:
43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at
com.sssw.fw.directory.realm.impl.jndildap.EboLdapContextProxyHandler.invokeMethod(
EboLdapContextProxyHandler.java:145)
    at
com.sssw.fw.directory.realm.impl.jndildap.EboLdapContextProxyHandler.invoke(EboLda
pContextProxyHandler.java:86)
    at com.sun.proxy.$Proxy155.getAttributes(Unknown Source)
    at
com.novell.srvprv.impl.vdata.model.VirtualDataAccess.getObjectType(VirtualDataAcce
ss.java:3943)
    at
com.novell.srvprv.impl.vdata.model.VirtualDataAccess.getObjectType(VirtualDataAcce
ss.java:4003)
    at
com.novell.srvprv.impl.vdata.model.VirtualDataModel.getObjectType(VirtualDataModel
.java:1405)
    at com.novell.soa.util.LdapUtil.isTypeOf(LdapUtil.java:109)
    at com.novell.soa.util.LdapUtil.isUser(LdapUtil.java:89)
```

The trace displays object unavailable errors only when errors occur while retrieving an object.

25.1.10 When Checking the Workflow Engine Heartbeat

The following are example entries that are logged when a user issues a request to check the state of the workflow engine.

```
2015-09-22 11:53:33,646 [TRACE] EngineStateDAO [RBPM] Updating heartbeat of Engine
Engine State: engineId = ENGINE, heartBeat= 2017-09-22 11:52:33.637, startTime=
2015-09-22 09:55:31.532, shutdownTime= 2017-09-22 09:52:58.773, engineState=
Running
```

```
2015-09-22 11:53:33,647 [DEBUG] HibernateUtil [RBPM] Beginning new transaction for
caller com.novell.soa.af.impl.persist.EngineStateDAO:141
```

```
2015-09-22 11:53:33,652 [TRACE] EngineStateDAO [RBPM] Engine heartbeat updated
successfully: Engine State: engineId = ENGINE, heartBeat= 2017-09-22 11:53:33.651,
startTime= 2017-09-22 09:55:31.532, shutdownTime= 2017-09-22 09:52:58.773,
engineState= Running
```

```
2015-09-22 11:53:33,660 [DEBUG] HibernateUtil [RBPM] Committed transaction for
caller com.novell.soa.af.impl.persist.EngineStateDAO:162
```

```
2015-09-22 11:53:33,660 [DEBUG] EngineImpl [RBPM] Heartbeat updated for engine:
ENGINE, time: 2017-09-22 11:53:33.651
```

```
2015-09-22 11:54:03,017 [DEBUG] HibernateUtil [RBPM] Beginning new transaction for
caller com.novell.soa.af.impl.timers.PendingActivityTimerTask:94
```

```
2015-09-22 11:54:03,018 [DEBUG] HibernateUtil [RBPM] Committed transaction for
caller com.novell.soa.af.impl.timers.PendingActivityTimerTask:94
```

If the workflow engine is not properly shutdown due to some reason, the identity applications assume that the engine is still running when you start it the next time. The following example traces are logged to indicate this situation.

```
2015-09-22 12:01:43,384 [WARN] EngineImpl [RBPM] Duplicate engine id detected. This
engine may not have been shutdown cleanly or another engine is running with engine-
id: ENGINE. Waiting 60000 ms for heartbeat to timeout.
```

```
2015-09-22 12:01:43,480 [INFO] EngineImpl [RBPM] Workflow Engine setState:
[RUNNING]
```

25.2 Troubleshooting Open XDAS Logging

If you enable OpenXDAS logging, but the OpenXDAS daemon is not running, the User Application fails to start. View the `xdasd.log` to verify that XDAS started successfully or to determine the reason for an unsuccessful start.

If you stop OpenXDAS and restart it, you must also stop and restart the User Application. If you do not restart the User Application, the connection between OpenXDAS and the User Application is lost so no User Application events are logged.

Table 25-1 *Commands for Starting/Stopping the XDASD daemon/service*

Action	Command
To start XDAS	<code>xdasd -s</code>
To stop XDAS	<code>xdasd -x</code>

25.3 Troubleshooting Self Service Password Reset Issues

25.3.1 Unable to Unlock Account through SSPR

Issue: This issue occurs if NMAS or eDirectory Challenge Responses are stored only in eDirectory and not in SSPR.

Workaround: Force the users to setup the challenge questions in SSPR as follows:

- 1 Go to SSPR **Configuration Editor**.
- 2 Click **Modules > Enabled > Authenticated > Setup Security Questions > Force Response**.

25.3.2 SSPR Reports Error 5027 When Attempting to Access Configuration Manager through Internet Explorer

Issue: In a new installation or upgraded setup of Self Service Password Reset 4.x, when you successfully log in to the SSPR portal, accessing the Configuration Manager or Configuration Editor displays the following error:

```
2016-11-01T15:54:00Z, ERROR, http.PwmResponse, {21,uaadmin} 5027
ERROR_UNAUTHORIZED (Internet Explorer version is not supported for this function.
Please use Internet Explorer 11 or higher or another web browser.)
[151.155.213.181]
```

Workaround: Disable Compatibility View in Internet Explorer for the domain that hosts your SSPR web application.

25.3.3 SSPR Reports Out of Order Page Request Error

Issue: This issue occurs when you click the **Back** button while in an SSPR page. SSPR displays an incorrect sequence message in the SSPR error log similar to the following:

```
ERROR, password.pwm.servlet.TopServlet, 5035 ERROR_INCORRECT_REQUEST_SEQUENCE
(expectedPageID=3, submittedPageID=4, url=<some sspr url>
```

Workaround: Disable the **Back** button detection from SSPR **Configuration Manager > Settings > Security > Web Security**.

NOTE: Changing this setting has no effect on end users.

25.3.4 Pressing Enter Button in SSPR's People Search Displays Locale Screen on Microsoft Internet Explorer 11 Browser

Issue: If you press Enter in People Search on Internet Explorer 11 browser, the Locale screen appears.

This issue is not reported on other browsers such as Microsoft Edge, Mozilla Firefox, and Google Chrome.

Workaround: Perform one of the following actions:

- ♦ The search bar searches as you type. Therefore you do not require to press Enter to search.
- ♦ Use a different browser.
- ♦ Close the locale prompt.

25.4 Entitlement Configuration Error During Codemap Refresh

Issue: When a new resource is created in a driver, the resource is not added to the User Application after running the code map refresh for the driver. One of the reasons that can cause this issue is missing value of some of the parameters in the entitlement configuration of the driver. For example, `<entitlement data-collection="false" dn="CN=ExchangeMailbox,CN=AD Driver for Groups,CN=DriverSet,O=system" parameter-format="" resource-mapping="" role-mapping="">`.

User Application reports the following error in the `catalina.out` file:

```
2017-11-03 15:55:21,373 [http-bio-8443-exec-340] ERROR
com.novell.idm.nrf.persist.DirXMLDriverDAO- [RBPM] Error occurred parsing the
entitlement configuration XML: cn=EntitlementConfiguration,cn=AD Driver for
Groups,cn=DriverSet,o=system
```

```
java.lang.StringIndexOutOfBoundsException: String index out of range: 0
```

Workaround: Add the missing values in the entitlement configuration for the driver. For example, `<entitlement data-collection="false" dn="CN=ExchangeMailbox,CN=AD Driver for Groups,CN=DriverSet,O=system" parameter-format=`

25.5 Mismatch of Certificates Used by Identity Manager Engine and User Application Causes Code (-9205) Error in `vnd.nds.stream`

Issue: The Identity Manager drivers use Identity Manager engine's keystore instead of User Application's keystore to access the User Application. If these components use different certificates, drivers report an error message similar to the following when set at Trace level 5:

DirXML Log Event

```
Message: Code(-9205) Error in vnd.nds.stream://VAULT/TEST/DRIVERSET1/DRIVER1/
Publisher/POLICY#XmlData:133 :
Couldn't request assignment of role: '<Role DN>' to identity: '<User DN>':
com.novell.nds.dirxml.soap.UserAppClientException: java.lang.RuntimeException:
javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException:
PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid
certification path to requested target
```

Workaround: Verify that the JRE used by the Identity Manager engine has the required certificate to connect to the User Application

- 1 Locate `cacerts` in the Identity Manager engine directory.

For example, `/opt/novell/eDirectory/lib64/nds-modules/jre/lib/security/cacerts` on Linux.

2 Determine the certificate used by the User Application.

2a Navigate to the User Application keystore.

For example, `/opt/netiq/idm/apps/jre/lib/security/cacerts`.

2b List the certificates by running the following command from the command line:

```
keytool -list -v -keystore cacerts
```

3 (Conditional) If you have access to the certificate, import the certificate into Identity Manager engine's `cacerts` directory by running the following command:

```
keytool -import -alias <newalias> -keystore cacerts -file certificate.der
```

4 (Conditional) If you do not have access to the certificate, export the certificate from the User Application's `cacerts` directory, and then import the certificate into Identity Manager engine's `cacerts` directory.

5 Restart the Identity Vault.