

Oracle® Fusion Middleware
Securing Oracle Enterprise Data Quality
12c (12.2.1.2.0)
E78155-02

November 2016

Oracle Fusion Middleware Securing Oracle Enterprise Data Quality, 12c (12.2.1.2.0)

E78155-02

Copyright © 2015, 2016 Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface	v
Audience	v
Documentation Accessibility	v
Related Documents	v
Conventions	vi
1 About EDQ Security	
1.1 Introducing EDQ Security	1-1
1.1.1 Authentication	1-1
1.1.2 Authorization	1-1
1.1.3 Encryption	1-2
1.1.4 Auditing	1-2
1.2 EDQ User Groups	1-2
1.3 EDQ Permissions	1-2
1.3.1 Application Permissions	1-2
1.3.2 Functional Permissions	1-2
1.3.3 Dynamic Permissions (in Case Management)	1-2
1.3.4 Project Permissions	1-2
1.4 Default EDQ Groups and Permissions	1-3
1.5 Default Administrator User Accounts	1-4
1.6 Mapping External Groups to EDQ Groups	1-4
1.7 Terms Used in this Guide	1-5
2 Applying Recommended Security Settings	
2.1 Configuring SSL with WebLogic	2-1
2.2 Configuring SSL with Tomcat	2-1
2.3 Processor Security	2-2
2.3.1 Default Permissions	2-3
2.3.2 Giving Scripts More Permissions	2-4
2.4 Encrypting LDAP Connections	2-5
2.5 Encrypting Database Connections	2-5
2.6 Limit Concurrent Logins	2-5
2.7 Disable FTP/SFTP Access	2-5
2.8 Exclude Configuration Area from FTP/SFTP	2-6
2.9 Account with Minimal Permissions for Service Integration	2-6

2.10	Protect JNDI Data Sources.....	2-6
3	User Authentication	
3.1	The EDQ login.properties File	3-1
3.1.1	Static Groups Mapping in login.properties	3-2
3.2	WebLogic Installations.....	3-2
3.3	Enabling the internal realm	3-3
3.4	Tomcat Installations.....	3-4
4	Integrating External User Management (LDAP) using WebLogic and OPSS	
4.1	Understanding Security Realms, Providers and Control Flags	4-1
4.2	Configuring WebLogic to use LDAP	4-2
4.2.1	Prerequisites	4-2
4.2.2	Integrating with Active Directory	4-3
4.2.3	WebLogic Configuration	4-4
4.2.4	EDQ Configuration.....	4-8
4.2.5	Filtering Groups.....	4-9
4.2.6	Using SSL to connect to LDAP	4-9
5	Configuring External User Management (LDAP) directly with EDQ	
5.1	Integrate EDQ with LDAP.....	5-1
5.1.1	Prerequisites	5-1
5.1.2	Integrating with Active Directory	5-2
A	Configuring EDQ to support Windows Integrated Authentication (Kerberos)	
A.1	EDQ running as Windows service using local system account.....	A-1
A.2	EDQ running on Unix	A-2
A.2.1	What is in the keytab?	A-2
A.2.2	Creating keytabs using existing tools	A-3
A.2.3	Creating keytabs using winktab.....	A-4
A.2.4	Check the Unix Kerberos configuration.....	A-5
A.2.5	Java Encryption.....	A-5
A.2.6	Changes to login.properties	A-5
B	Configuring Single Sign On with Oracle Access Manager (OAM)	
B.1	Prerequisites	B-1
B.2	OAM configuration	B-1
B.3	WebLogic plugin configuration.....	B-2
B.4	WebLogic Configuration	B-2

Preface

This guide explains the Oracle Enterprise Data Quality security features and administration.

Audience

The intended audience of this guide are experienced administrators, Java developers, deployers, and application managers who want to ensure that EDQ meets Oracle security standards.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Fusion Middleware documentation set.

EDQ Documentation Library

The following publications are provided to help you install and use EDQ:

- *Oracle Fusion Middleware Release Notes for Enterprise Data Quality*
- *Oracle Fusion Middleware Installing and Configuring Enterprise Data Quality*
- *Oracle Fusion Middleware Administering Enterprise Data Quality*
- *Oracle Fusion Middleware Understanding Enterprise Data Quality*
- *Oracle Fusion Middleware Integrating Enterprise Data Quality With External Systems*
- *Oracle Fusion Middleware Securing Oracle Enterprise Data Quality*
- *Oracle Enterprise Data Quality Address Verification Server Installation and Upgrade Guide*

- *Oracle Enterprise Data Quality Address Verification Server Release Notes*

Find the latest version of these guides and all of the Oracle product documentation at <https://docs.oracle.com>

Online Help

Online help is provided for all Oracle Fusion Middleware user applications. It is accessed in each application by pressing the **F1** key or by clicking the Help icons. The main nodes in the Director project browser have integrated links to help pages. To access them, either select a node and then press **F1**, or right-click on an object in the Project Browser and then select **Help**. The EDQ processors in the Director Tool Palette have integrated help topics, as well. To access them, right-click on a processor on the canvas and then select **Processor Help**, or left-click on a processor on the canvas or tool palette and then press **F1**.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

About EDQ Security

This chapter explains the key security concepts used in Enterprise Data Quality (EDQ). It includes the following sections:

- [Introducing EDQ Security](#)
- [EDQ User Groups](#)
- [EDQ Permissions](#)
- [Default EDQ Groups and Permissions](#)
- [Default Administrator User Accounts](#)
- [Mapping External Groups to EDQ Groups](#)
- [Terms Used in this Guide](#)

1.1 Introducing EDQ Security

Security within EDQ applies to accessing the application (ensuring that only authorized users can access it, and that data within the application is secured), and auditing of user actions to identify anomalies. This guide covers the following:

- Providing secure access control
- Securing data at rest and in transport.
- Providing appropriate security auditing capabilities to ensure user activity can be securely logged and traced.

For more information, see [About Oracle Enterprise Data Quality in Understanding Oracle Enterprise Data Quality](#).

1.1.1 Authentication

Details of users and groups in EDQ can be stored within its own internal directory or taken from an external LDAP server, such as Microsoft Active Directory. Using external authentication sources enables EDQ to share user credentials with other systems, reducing the number of passwords that users need to remember and maintain, while eliminating overhead in management of users and groups.

1.1.2 Authorization

Authorization controls what users can do once they have authenticated successfully. Authorization of users is based on a model of users, and permissions associated with groups. A user is a member of one or more groups (either directly or by mapping an

external group to an internal group), and is authorized according to the permissions that are associated with that group.

1.1.3 Encryption

Both the WebLogic and Tomcat servers support HTTPS and require that traffic between the client and EDQ is encrypted so that it cannot be read or modified in transit. For environments where HTTPS is not an option, EDQ encrypts passwords sent between the client and server.

1.1.4 Auditing

EDQ supports auditing of user actions using the Oracle Fusion Middleware Audit Framework. In addition, EDQ can be configured to write audit information to files.

1.2 EDQ User Groups

All rights to EDQ are granted using a set of permissions that are granted to a user group (or 'internal' group). Where users are managed externally, for example in WebLogic, or in Active Directory, or in an alternative LDAP provider, permissions are granted by mapping External Groups from that directory to these internal groups. Users in the external group then have the permissions from whichever group or groups the external group is mapped to. Where users are managed in EDQ (in the 'internal' realm), these users are made direct members of one or more of the internal groups, and therefore have the associated permissions.

1.3 EDQ Permissions

EDQ has four types of permission, as listed below. Permissions are granted to users by means of EDQ User Groups, which are in turn mapped to external user groups.

1.3.1 Application Permissions

These are simple permissions that grant (or deny) groups of users access to log in to EDQ user applications, such as Director, Server Console, and Case Management.

1.3.2 Functional Permissions

These are permissions to access certain functions of the system, for example, the ability to modify a Reference Data set in Director, or the ability to change the state of a case, or alert in Case Management.

1.3.3 Dynamic Permissions (in Case Management)

These permissions are dynamically created within a Case Management workflow as configured in Case Management Administration. Dynamic permissions are used to restrict access to specific cases or alerts or to case comments or attachments.

1.3.4 Project Permissions

By default, a project created in Director is accessible to all user groups. However, projects may be restricted to a smaller set of groups. Where this is the case, any user that is not in a group that has access to the project, will not be able to see or use the project in any way.

Note: Any user that has the ability to add projects in Director automatically has access to all projects.

1.4 Default EDQ Groups and Permissions

The default groups and a summary of their permissions are listed below. To see full details on the precise set of permissions granted to each group, select a group from the Administration - Groups option on the EDQ Launchpad (after logging in as an administrator), and click on the **Edit** button.

Table 1–1 *Permissions for various user groups*

Group	Summary	Functional Permissions	Application Permissions
Administrators	Power users with all functional and administrative privileges	All Dynamic Case Management permissions are not automatically granted to Administrators	All
Data Stewards	Users with review access to Director and Dashboard, with permission to review all results, resolve issues, and make manual match and merged output decisions, but without permission to create or change any processing logic	Read-only permissions to Director Full permissions to Match Review	Dashboard Director Match Review Issue Manager Case Management Server Console
Executives	Users with access to Dashboard results only	Dashboard: View Dashboard	Dashboard
Match Reviewers	Users with access to the Match Review application, the Case Management application and Dashboard only	Basic Case Management permissions (no bulk updates and cannot view cases assigned to others) Dashboard: View Dashboard	Match Review Issue Manager Case Management Dashboard
Review Managers	Users with access to the Case Management application, with permission to configure case management and perform bulk edits on cases and alerts.	Full Case Management permissions except deletion and editing of audit trail activities	Case Management Dashboard

Table 1–1 (Cont.) Permissions for various user groups

Group	Summary	Functional Permissions	Application Permissions
Data Analysts	Users with permission to create and modify processing logic in Director, but with no administration privileges	Full access to Director except the ability to modify System-level (cross-project) configuration and access the Users data store.	Director Server Console Match Review Case Management Issue Manager

1.5 Default Administrator User Accounts

On an installation of EDQ on WebLogic server, the weblogic administrator user account (normally 'weblogic', with a password selected when the domain was created), is a member of the 'Administrators' group in WebLogic, which is mapped to the 'Administrators' group in EDQ by means of a default *login.properties* file in the EDQ Home configuration directory. This therefore grants the weblogic user, and any other users in the WebLogic Administrators group, administrative access to EDQ.

Note: Ensure this mapping is fixed and intended only to grant initial access so that the actual required permissions can be set by mapping external groups to internal groups, see [Section 1.6, "Mapping External Groups to EDQ Groups"](#)

On an installation of EDQ on Tomcat, a default internal administrator user account called '*dnadmin*' is created. The initial password for this user is also '*dnadmin*', but it must be changed to a more secure password after initial login. In such an installation, by default this is the only user account with administration rights, and it is important not to delete the user or forget the password, as otherwise it may not be possible to access the system.

1.6 Mapping External Groups to EDQ Groups

External groups are mapped to internal EDQ Groups from the Administration pages of the EDQ Launchpad. An external group is granted all permissions from all EDQ groups that it is mapped to.

To map external groups to internal EDQ groups, follow the steps below:

1. Log in to the EDQ Launchpad as an Administrator.
2. Click on the Administration dropdown link, and click on External Groups.
3. Expand the name of the external realm (this is 'ORACLE' on a default system) and click on a group to edit its mappings.
4. Select the EDQ Groups you want to grant to the external group, and click on Save to save the changes.

It is useful and desirable to create some external groups on the domain that can be mapped precisely to internal EDQ groups, to ensure optimal permission assignment.

Note: For installations with a large number of external user groups, an optional filter can be specified in a local *login.properties* file to narrow down the list of groups that is available in this screen. See [Section 4.2.5, "Filtering Groups"](#)

for more information.

1.7 Terms Used in this Guide

The following terms are used in this guide:

- AD – Active Directory
- Certificate – Generally refers to an X.509 certificate
- Kerberos – Network authentication protocol
- LDAP – Lightweight Directory Access Protocol
- OID – Oracle Internet Directory
- OPSS - Oracle Platform Security Services
- SSL – Security Sockets Layer, a protocol for encrypted connections over which application traffic can be transported. Replaced by TLS, although SSL is still used as a generic term.
- TLS – Transport Layer Security, a successor to SSL.
- WLS – WebLogic Server
- X.509 Certificate – A certificate issued by a trusted authority (certificate authority) to certify that a specified entity (individual, organization, server, or other entity) holds the matching private key for a public key.

Applying Recommended Security Settings

This chapter provides tips for securing the EDQ environment.

The chapter includes the following sections:

- [Configuring SSL with WebLogic](#)
- [Configuring SSL with Tomcat](#)
- [Processor Security](#)
- [Encrypting LDAP Connections](#)
- [Encrypting Database Connections](#)
- [Limit Concurrent Logins](#)
- [Disable FTP/SFTP Access](#)
- [Exclude Configuration Area from FTP/SFTP](#)
- [Account with Minimal Permissions for Service Integration](#)
- [Protect JNDI Data Sources](#)

2.1 Configuring SSL with WebLogic

For instructions on configuring SSL with WebLogic Server, see the WebLogic documentation:

[Overview of Configuring SSL in WebLogic Server](#)

2.2 Configuring SSL with Tomcat

To enable encrypted connections with Tomcat, the HTTPS connector must be configured using the following procedure:

1. Locate the `server.xml` file for the Tomcat installation (generally this would be `conf/server.xml` within the Tomcat directory). By default it contains a section such as the following:

```
<!-- Define a SSL/TLS HTTP/1.1 Connector on port 8443
This connector uses the NIO implementation that requires the JSSE
style configuration. When using the APR/native implementation, the
OpenSSL style configuration is required as described in the APR/native
documentation -->
<!--
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS" />
```

-->

2. Enable the Connector element by removing the XML comment characters around it.
3. Set the port value for HTTPS if needed. The default is 8443, so if a different value is used also change the `redirectPort` value in the HTTP connector to match.

Remember that if using a port below 1024, the server may require special permissions depending on the OS.

4. Generate the server key and certificate, and have the certificate signed by a recognized certificate authority. Self-signed certificates can be used, however they will need to be installed on the client machines in order for them to be recognized.

Note: The certificate is stored either in a Java keystore (JKS format) or as a PKCS#12 file. The latter may be preferred in certain instances, as there are many tools available for working with PKCS#12 files.

5. Update the connector element as follows, replacing `pathtokeystorefile`, `keystorepassword` and `keystoretype` with the referenced information:

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS"
keystoreFile="pathtokeystorefile"
keystorePass="keystorepassword"
keystoreType="keystoretype"
/>
```

6. Set the `keystoreType` value to JKS or PKCS12 as required. If the key store contains multiple certificates, use the `keyAlias` attribute to set the alias.
7. Some Tomcat distributions include the Apache Portable Runtime (APR) native library. If this is the case, the certificate must be configured using Apache HTTPD `mod_ssl` style attributes. For example:

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11AprProtocol"
SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
clientAuth="false"
SSLCertificateFile="pathtocrtfile"
SSLCertificateKeyFile="pathtokeyfile" />
```

For additional Tomcat information, see *Apache Tomcat Configuration Reference* at <http://tomcat.apache.org/tomcat-8.0-doc/config/http.html>

For additional `mod_ssl` information, see *Apache Module mod_ssl* at http://httpd.apache.org/docs/2.2/mod/mod_ssl.html

2.3 Processor Security

EDQ 12.2.1.2 now allows optional use of security controls which can be used to increase the security level of EDQ processors, for example to block insecure uses of the Script processor. For backward compatibility reasons processor security is off by default, but it can be enabled by adding the following line to [EDQ Local Home]/director.properties:

```
processor.security= off/low/medium/high
```

The processor security option acts in conjunction with the use of a Java Security Manager. The use of a Java Security Manager is controlled by a Java option specified on the server (-Djava.security.manager); this is enabled by default for new installations of EDQ on WebLogic server as it is specified in the setStartupEnv.sh script that sets the Java options for the EDQ server startup group. In other installations it must be manually specified.

The security level of each of the different processor.security settings is summarized below:

- **Off:**

No security restrictions are applied.

- **Low**

The following restrictions are applied:

- The use of the Script processor from the tool palette, for direct use in processes, is disabled if the system is not running with a Java Security Manager.
- If the system is running with a Java Security Manager, the Script processor is available but is only granted a very small set of default permissions, limited to data processing. Scripts will not be able to make network connections or issue commands outside of the application.

Note: Java processors and scripts that are packaged in jars will run without any restrictions; if a jar contains a permissions element the processor will be granted only those permissions.

- **Medium:**

The same restrictions as Low apply, except that Java processors and scripts that are packaged in jars will be granted a very limited set of permissions. Any additional permissions required by the processor must be listed in permissions elements.

To clarify the difference, in 'low' level, all permissions are granted and permissions elements restrict permissions; in 'medium' level, permissions are limited and permissions elements extend permissions. In both levels a processor with a permissions element will run with exactly the same set of permissions.

- **High:**

The same restrictions as 'medium' apply, except that only processors that are signed by the Oracle EDQ certificate, or by certificates granted to approved partners, will be allowed to include permissions elements that grant additional permissions to processors.

If a permissions element is found in an unsigned jar the processors in the jar will be rejected and will not appear in the processor palette.

2.3.1 Default Permissions

When running with processor security set to **medium** or **high**, the permissions granted to processors allow the reading of these system properties:

- java.version
- java.vendor
- java.vendor.url
- java.class.version
- os.name
- os.version
- os.arch
- file.separator
- path.separator
- line.separator
- java.specification.version
- java.specification.vendor
- java.specification.name
- java.vm.specification.version
- java.vm.specification.vendor
- java.vm.specification.name
- java.vm.version
- java.vm.vendor
- java.vm.name

No other permissions will be granted.

2.3.2 Giving Scripts More Permissions

An installation may wish to run with secure scripts but still allow users to perform some additional operations in ad-hoc scripts. A typical example will be to make web service calls. To allow this additional functionality, the system supports a configuration file listing additional permissions for ad-hoc scripts.

Additional permissions files are located in the security/permissions folder in the local configuration directory. For script processors, the file is named scriptprocessor.xml and for script match gadgets the file is named scriptgadget.xml. The file XML format is:

```
<permissions>
  <permission type="permissionclass" [name="targetname" [action="action"]]/>
  ...
</permissions>
```

The permission class is the full name of any Java permissions class. The name attribute is present if the permission has a target name - the class has a two- or three-argument constructor. The action is present if the permission has an action - the class has a three-argument constructor.

For example, to grant the permission to read the file /etc/hosts:

```
<permission type="java.io.FilePermsion" name="file:/etc/hosts" action="read"/>
```

System properties can be included in the name using \${...} substitution. Alongside the standard system properties, these additional properties are available:

- **rootdir**: The location of the web application root.
- **webinfdir**: The location of the web application WEB-INF directory.
- **config.path**: The configuration directory path. A permissions entry using \${config.path} in its name will be replaced by an entry for each location in the path.

2.4 Encrypting LDAP Connections

Connections from EDQ to an LDAP directory can be encrypted using either an SSL/TLS connection layer or by negotiating encryption after a connection has been established (StartTLS).

2.5 Encrypting Database Connections

JDBC URL syntax for connections over TLS is dependent on the database driver being used. Connections to an Oracle database can be encrypted by adding the following property to the datasource connection pool configuration.:

```
oracle.net.encryption_client = REQUIRED
```

For more information see

<http://docs.oracle.com/database/121/JJDBC/clntsec.htm#JJDBC28296>

2.6 Limit Concurrent Logins

A limit can be specified in *login.properties* for the number of concurrent logins by an individual user. The limit is concurrent logins per application. So if the sessionlimit is 1 a user can login to director, server console, case management at the same time but cannot login twice to any.

This can be configured either globally or on a per realm basis. To set this globally for all realms, use the following line:

```
sessionlimit = 1
```

To use different settings for different realms, specify the realm name before the parameter - for example:

```
internal.sessionlimit = 1
```

Note: Using the line given above, you can also limit the concurrent logins in an `internal' realm, meaning the users set up and administered in EDQ itself.

```
<external realm name>.sessionlimit = 1
```

2.7 Disable FTP/SFTP Access

Where it is not needed, FTP/SFTP access to EDQ should be disabled for optimum security. Standard FTP can be disabled by adding the following line in *director.properties* within the local configuration folder:

```
launch.ftpserver = 0
```

SFTP/SCP can be disabled using the following line within the same file:

```
launch.sshd = 0
```

2.8 Exclude Configuration Area from FTP/SFTP

If non-admin users are allowed access to FTP/SFTP, it is advisable to remove access to the configuration folder from the SFTP server, as follows:

1. Create the folders `extras/ftpserver/conf` and `extras/ssh/conf` within `oedq.local.home`, if they do not already exist.
2. Copy the files `extras/ftpserver/conf/ftpserver.xml` and `extras/ssh/conf/ssh.xml` from the `oedq.home` configuration directory to the corresponding subfolders of `oedq.local.home`.
3. In each of the two files from the previous stage, comment out the following lines:

```
<!-- Configuration area -->  
<ref bean="configspaces"/>  
<!-- Command areas -->  
<ref bean="commandspaces"/>
```

The first reference is to the configuration directories; the second is to the command areas used for external tasks.

4. Restart the application server. The only location visible to the FTP and SFTP servers is now the landing area.

2.9 Account with Minimal Permissions for Service Integration

If Siebel is integrated with EDQ for batch jobs, the user account also needs the permission to connect to the management (JMX) port in order to start EDQ jobs. This is controlled by the System/System Administration functional permission. Also the 'connect to messaging system' should be listed as 'System/Connect to Messaging System'. The section ought also to reference Oracle Data Integrator (ODI). The EDQ account used for ODI integration will require the same permissions (it actually does not currently need Connect to Messaging System but it might eventually and it is good practice to include it). So, we need something like:

An EDQ account used by a remote system such as Siebel or Oracle Data Integrator should have the minimum set of permissions on an EDQ system. Specifically, the account should be in a custom group with the following permissions only, and no access to log into any user applications or perform any other functions:

- System / Connect to Messaging System - so that it is authorized to communicate with EDQ web services and JMS.
- System/ System Administration - so that it is authorized to connect to the EDQ Management (JMX) Port and can initiate jobs.
- Permissions to any projects containing any service interfaces (e.g. web services or jobs) that it needs to be able to call.

2.10 Protect JNDI Data Sources

Unless specific steps are taken, a user in EDQ can set up a data store with a reference to the JNDI name of any existing data source and then access data in these schemas, which can contain very sensitive information. To protect JNDI data sources in EDQ, specify the names (or regular expressions matching the names) in `director.properties`:

```
protected.jndi.datasources = <space separated list of JNDI names>
```

For example:

```
protected.jndi.datasources = jdbc/edqconfig jdbc/edqresults
```

The property is a space-separated list of regexes so you could also use:

```
protected.jndi.datasources = jdbc/edq.*
```

Note: When setting this value in the local *director.properties*, always include the default setting from the base properties file. This setting prevents access to the WebLogic internal data sources as well as the EDQ data sources.

User Authentication

This chapter explains how users are authenticated in EDQ.

This chapter includes the following sections:

- [The EDQ login.properties File](#)
- [WebLogic Installations](#)
- [Enabling the internal realm](#)
- [Tomcat Installations](#)

3.1 The EDQ login.properties File

User authentication in EDQ is configured using a file named *security/login.properties* in the EDQ configuration area. The file may exist in either the 'home' configuration directory or the 'local' configuration directory, or both. If present in both, the settings are merged with the values in the 'local' directory taking precedence. If you need to make changes to the file, always edit a version in the 'local' directory.

The EDQ 'home' configuration area contains a file *security/login.properties.template* which contains example settings for several types of LDAP integration.

The *login.properties* file defines a number of 'realms'. Each realm is an independent store of users. The file will start with global settings, followed by settings for each realm. Standard global settings are:

```
realms = realm1, realm2, ...
gss    = false
```

The *realms* property defines the list of realms configured in this installation. The names are arbitrary, except that the special realm name 'internal' specifies the EDQ internal user store (user dnadmin etc).

The 'gss' setting turns off advanced Kerberos style authentication.

The global settings are followed by blocks of realm specific properties, each prefixed with the realm name from the global list. For example, a configuration which uses the internal realm and an LDAP realm could be:

```
realms          = internal, corpldap
gss             = false
corpldap.realm  = EXAMPLE.COM
corpldap.ldap.server = dc1.example.com
...
```

These settings are covered in more detail below.

If more than one realm is defined in *login.properties*, the EDQ login screens - web console and UIs - will contain a dropdown selector for the realm associated with the username and password.

3.1.1 Static Groups Mapping in *login.properties*

If you are using an external LDAP user store, and do not wish to use the EDQ internal user store, then you will face a bootstrapping problem when attempting to setup mappings from LDAP groups to EDQ groups. This is done using the EDQ web console and requires an EDQ administrator login. However a LDAP group mapping to the EDQ Administrators group is required before an LDAP user can login to EDQ.

To overcome this problem you can define static group mappings in *login.properties*. The syntax is:

```
realm.xgmap = exgroup1 -> edqgroup1, exgroup2 -> edqgroup2 ...
```

'realm' is the realm name as listed in the global realms list. Each 'exgroup' is the name of an external LDAP group; each 'edqgroup' is the name of an EDQ group.

Static mappings should be used only to set the initial Administration mapping; other mappings should be configured in the EDQ web console External Groups page.

For example, to map an LDAP group named 'EDQ-ADMINS' to the EDQ Administrators group, add:

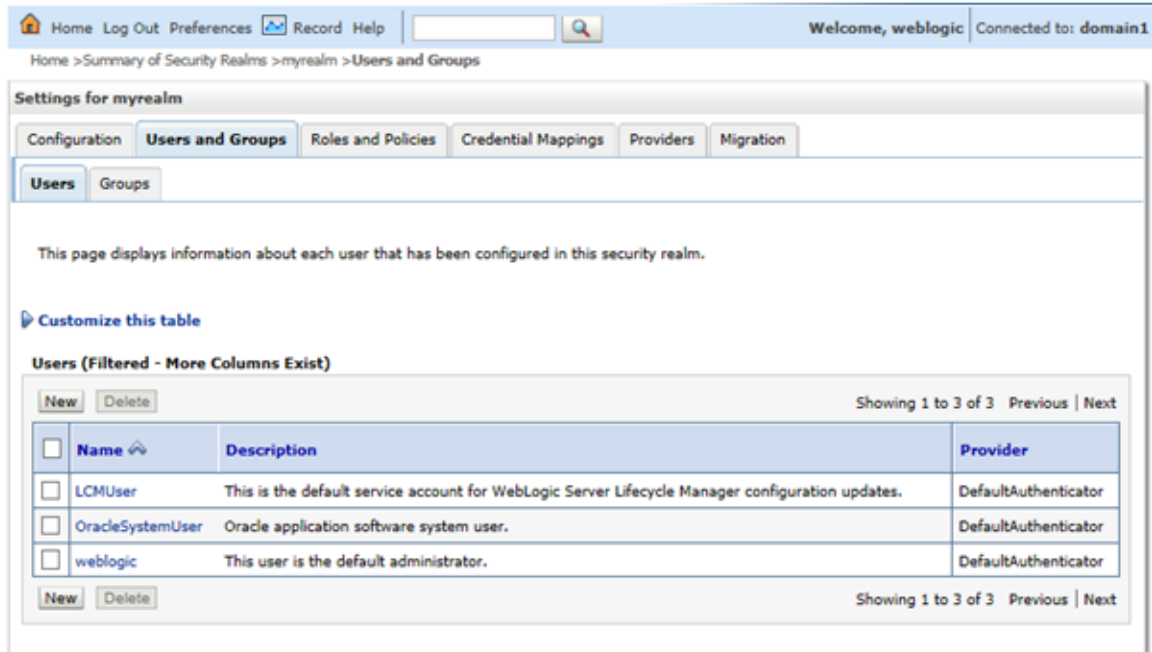
```
corpldap.xgmap = EDQ-ADMINS -> Administrators
```

3.2 WebLogic Installations

A standard installation of EDQ on WebLogic will use the domain's internal user store for authentication. This contains the domain administration user (usually 'weblogic') and a small set of other users.

The EDQ default security configuration maps any user in the WebLogic Administrators group to the EDQ Administrators group. No other group mappings are defined initially.

To add more users, login to the WebLogic administration console, and navigate to the Users and Groups tab in the default security realm.



As an alternative to managing users in WebLogic you can integrate with an external source of users, normally an LDAP server, such as Active Directory. Integration can be configured in the WebLogic administration console or directly from EDQ by editing the security configuration file. The former approach is covered in [Chapter 4, "Integrating External User Management \(LDAP\) using WebLogic and OPSS"](#) and the latter in [Chapter 5, "Configuring External User Management \(LDAP\) directly with EDQ"](#).

The default *login.properties* in the 'home' configuration directory contains:

```
realms      = opss
opss.realm = ORACLE
opss.label = Weblogic
opss.type  = opss
opss.xgmap = Administrators -> Administrators
```

This configuration indicates that the OPSS library supplied by WebLogic is used for authentication and that the WebLogic Administrators group is mapped to the EDQ Administrators group.

3.3 Enabling the internal realm

To manage users internally, the internal realm needs to be enabled. Follow the procedure below to enable the internal realm for users:

1. Create a subdirectory called security in the local configuration directory (*oedq_local_home/security*).
2. Copy the *login.properties* file from the security directory of the base configuration directory (*oedq_home/security*) to *oedq_local_home/security*.
3. Add 'internal' to the comma-separated list of realms.

Note that 'opss' enables the realm of WebLogic users. To only manage users internally, you can remove opss from the list of realms.

4. Restart the server.

This enables the internal realm of users, and allow internal user accounts to be managed in the Administration pages on the Launchpad. A single default Administrator user '**dnadmin**' will exist, with an initial password of '**dnadmin**' that will need to be changed on first login.

If more than one realm is defined, users will need to select which realm they want to use when they log in.

3.4 Tomcat Installations

An installation of EDQ on Tomcat will use the EDQ internal authentication mechanism. This contains the single user 'dnadmin' with full administration rights. Additional users can be added through the EDQ administration web pages.

You can integrate with an external LDAP server such as Active Directory by editing the EDQ security configuration file. This is covered in [Chapter 5, "Configuring External User Management \(LDAP\) directly with EDQ"](#).

No *login.properties* file is used by default in Tomcat installations because the default setting is to use internal authentication. If additional setting are required, or LDAP integration is required, create a new *security/login.properties* file in the 'local' configuration directory and make changes there.

Integrating External User Management (LDAP) using WebLogic and OPSS

This chapter provides information on integration to LDAP using Weblogic and OPSS. It includes the following sections:

- [Understanding Security Realms, Providers and Control Flags](#)
- [Configuring WebLogic to use LDAP](#)

4.1 Understanding Security Realms, Providers and Control Flags

A WebLogic domain can contain one or more Security Realms. Each realm defines a security configuration, users and groups. Only one realm can be active at any one time, and generally there is no benefit in creating additional realms. The default realm in a WebLogic domain is named 'myrealm':

Summary of Security Realms

A security realm is a container for the mechanisms—including users, groups, security roles, security policies, and security providers—that are used to protect WebLogic resources. You can have multiple active security realms in a WebLogic Server domain, but only one can be set as the default security realm, which is reserved for domain administrative purposes.

This Security Realms page lists each security realm that has been configured in this WebLogic Server domain. Click the name of the realm to explore and configure that realm.

[Customize this table](#)

Realms (Filtered - More Columns Exist)

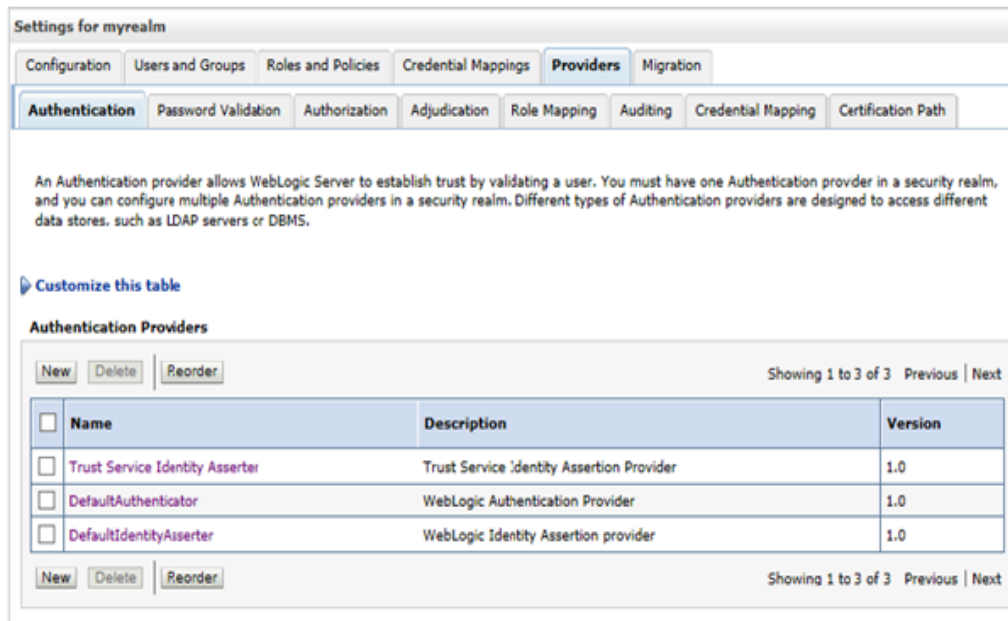
<input type="checkbox"/>	Name ↕	Default Realm
<input type="checkbox"/>	myrealm	true

Showing 1 to 1 of 1 Previous | Next

Each realm contains a number of security 'providers'. Providers are 'authenticators', which handle user authentication and user stores, or 'identity asserters', which can determine user identity from data embedded in a request, such as X.509 certificates or SAML tokens. The Oracle Access Manager Identity Asserter, used for OAM

integration, is explained in [Section B.4, "WebLogic Configuration"](#).

The default security realm has one authenticator and two identity asserters:



The **DefaultAuthenticator** handles users in the internal WebLogic store.

Each Authenticator provider has a control flag setting. The flag determines the part that the provider plays in authenticating a user. There are four possible flag values, but the two of importance here are:

REQUIRED: authentication in this provider must succeed for the overall process to complete.

SUFFICIENT: if authentication in this provider succeeds, the overall process is successful, as long as no other authenticator has the **REQUIRED** control flag.

The **DefaultAuthenticator** is pre-configured with the control flag set to **REQUIRED** so that it is required to succeed even if an LDAP authentication provider is configured. Part of the process of configuring LDAP support is to change this value to **SUFFICIENT**.

The order of providers is also important. When more than one authenticator is enabled and configured with the correct control flags, a user supported by any of the authenticators is able to login to the WebLogic Administration Console. However EDQ has access to the first authenticator in the list only; users from authenticators lower in the list will not be recognized. Therefore it is important that when an LDAP authenticator is added, it is moved to the top of the list. This is also covered below.

4.2 Configuring WebLogic to use LDAP

This section consists of pre-requisites and procedure to configure WebLogic.

4.2.1 Prerequisites

Before beginning the configuration process, the following information must be gathered:

- The type of LDAP server in use. WebLogic provides authenticators for several types of LDAP server, including Microsoft Active Directory, OpenLDAP and Oracle Internet Directory. There is also a generic LDAP authenticator which can be used with servers not supported directly (such as AD-LDS).
- The host name and port of the LDAP server. If your environment contains multiple servers for high availability, you can use more than one host in the configuration. The default LDAP port is 389.
- The identity and password of an LDAP user which can connect and perform searches. The user identity is normally a full Distinguished Name (DN) but Active Directory also allows shorter forms.
- The locations in the LDAP tree (base DNs) where users and groups can be found.
- The LDAP attribute on a user record which identifies the user on login. Whilst most LDAP user schemas have standard user name attributes, organizations can choose to use others.
- The LDAP attribute on a group record that identifies the group. In most installations, this can be the group 'Common Name' (cn).
- The name of an LDAP group containing all the users that will be working with EDQ. The group is used to filter the list of users presented for EDQ issue assignment, etc. Without this filter, every user in the LDAP server would be presented, and this is generally not recommended.

4.2.2 Integrating with Active Directory

The rest of this section will cover the detailed steps for an integration with Active Directory. The steps are broadly similar for other types of LDAP server. Initial configuration will not use SSL.

In the walkthrough, these settings are used:

- **Active Directory Domain:** EXAMPLE.COM
- **Domain Controller (LDAP server):** dc1.example.com. The default, non-SSL, port 389 will be used.
- **LDAP user:** *cn=netuser,cn=users,dc=example,dc=com*. Assuming that the AD username for this user is 'netuser' then you can also use netuser@example.com or example\netuser.
- **User base DN:** *dc=example,dc=com*. Here the base is the root of the full LDAP tree. If all the users are contained in a subtree, you could use something like: *cn=users,dc=example,dc=com*.
- **Group base DN:** *dc=example,dc=com*. Again a subtree could be used if suitable.
- **User name attribute:** *sAMAccountName*. This is the standard AD attribute which stores the short login name. In the Active Directory Users and Computers application this is displayed as the pre-Windows 2000 login name:

Member Of	Dial-in	Environment	Sessions
Remote control	Remote Desktop Services Profile		COM+
General	Address	Account	Profile
		Telephones	Organization

User logon name:
 @EXAMPLE.COM

User logon name (pre-Windows 2000):

- **Group name attribute:** *cn*
- **All EDQ users group:** *edqusers*

4.2.3 WebLogic Configuration

Note: Ensure that the WebLogic Administration server is running. It is recommended that the EDQ managed server(s) be stopped before beginning the configuration.

To configure the WebLogic, follow the steps below:

1. Login to the Administration Console and navigate to Security Realms/myrealm and click the providers tab. Click New to add a new provider. Enter a name, for example, AD and select the **ActiveDirectoryAuthenticator**:

Create a New Authentication Provider

OK | Cancel

Create a new Authentication Provider

The following properties will be used to identify your new Authentication Provider.

* Indicates required fields

The name of the authentication provider.

* **Name:**

This is the type of authentication provider you wish to create.

Type:

OK | Cancel

2. Click **OK**. The new, unconfigured provider is added to the bottom of the list. An alert will indicate that the Administration Server must be restarted for the changes to take effect. This is not necessary at this stage; restart the server when the configuration is complete.

Click on the new provider and on the Configuration screen change the control flag to **SUFFICIENT**:



Settings for AD

Configuration Performance

Common Provider Specific

Save

This page displays basic information about this Active Directory Authentication provider. You can also use this page to set the JAAS Control Flag to control how this provider is used in the login sequence.

 Name:	AD	The name of this Active Directory Authentication provider. More Info...
 Description:	Provider that performs LDAP authentication	A short description of this Active Directory Authentication provider. More Info...
 Version:	1.0	The version number of this Active Directory Authentication provider. More Info...
 Control Flag:	SUFFICIENT ▼	Specifies how this Active Directory Authentication provider fits into the login sequence. More Info...

Save

- Click **Save** and then select the **Provider Specific** tab. This is where the LDAP server information is entered. In the Connection area, enter the LDAP server, port, user (Principal) and password (Credential). Leave the **SSLEnabled** check box unset.

Connection

Host:	<input type="text" value="dc1.example.com"/>	The host name or IP address of the LDAP server. More Info...
Port:	<input type="text" value="389"/>	The port number on which the LDAP server is listening. More Info...
Principal:	<input "="" type="text" value="cn=netuser,cn=users,dc="/>	The Distinguished Name (DN) of the LDAP user that WebLogic Server should use to connect to the LDAP server. More Info...
Credential:	<input type="password" value="••••••"/>	The credential (usually a password) used to connect to the LDAP server. More Info...
Confirm Credential:	<input type="password" value="••••••"/>	
<input type="checkbox"/> SSLEnabled		Specifies whether the SSL protocol should be used when connecting to the LDAP server. More Info...

- In the Users area, enter the **User Base DN** and replace 'cn' with the user name attribute in the **User From Name Filter** and **User Name Attribute** fields:

Users		
User Base DN:	<input type="text" value="dc=example,dc=com"/>	The base distinguished name (DN) of the tree in the LDAP directory that contains users. More Info...
All Users Filter:	<input type="text"/>	If the attribute (user object class) is not specified (that is, if the attribute is null or empty), a default search filter is created based on the user schema. More Info...
User From Name Filter:	<input type="text" value="(&(sAMAccountName=%"/>	If the attribute (user name attribute and user object class) is not specified (that is, if the attribute is null or empty), a default search filter is created based on the user schema. More Info...
User Search Scope:	<input type="text" value="subtree"/> <input type="button" value="v"/>	Specifies how deep in the LDAP directory tree the LDAP Authentication provider should search for users. More Info...
User Name Attribute:	<input type="text" value="sAMAccountName"/>	The attribute of an LDAP user object that specifies the name of the user. More Info...
User Object Class:	<input type="text" value="user"/>	The LDAP object class that stores users. More Info...
<input type="checkbox"/> Use Retrieved User Name as Principal		Specifies whether or not the user name retrieved from the LDAP server should be used as the Principal in the Subject. More Info...

- In the Groups area, enter the **Group Base DN**, and replace 'cn' in the **Group From Name Filter** if you are not using 'cn' as the group name attribute:

Groups		
Group Base DN:	<input type="text" value="dc=example,dc=com"/>	The base distinguished name (DN) of the tree in the LDAP directory that contains groups. More Info...
All Groups Filter:	<input type="text"/>	An LDAP search filter for finding all groups beneath the base group distinguished name (DN). If the attribute is not specified (that is, if the attribute is null or empty), a default search filter is created based on the Group schema. More Info...
Group From Name Filter:	<input type="text" value="(&(cn=%g)(objectclass=g"/>	An LDAP search filter for finding a group given the name of the group. If the attribute is not specified (that is, if the attribute is null or empty), a default search filter is created based on the group schema. More Info...
Group Search Scope:	<input type="text" value="subtree"/> <input type="button" value="v"/>	Specifies how deep in the LDAP directory tree to search for groups. Valid values are subtree and onelevel. More Info...
Group Membership Searching:	<input type="text" value="unlimited"/> <input type="button" value="v"/>	Specifies whether group searches into nested groups are unlimited, limited or off. Valid values are unlimited, limited and off. More Info...
Max Group Membership Search Level:	<input type="text" value="0"/>	Specifies how many levels of group membership can be searched. This setting is valid only if GroupMembershipSearching is set to limited. Valid values are 0 and positive integers. For example, 0 indicates only direct group memberships will be found, and a positive number indicates the number of levels to search. More Info...
<input type="checkbox"/> Ignore Duplicate Membership		Determines whether duplicate members are ignored when adding groups. The attribute cycles in the Group membership. More Info...
<input type="checkbox"/> Use Token Groups For Group Membership Lookup		Indicates whether to use the Active Directory TokenGroups attribute lookup algorithm instead of the standard recursive group membership lookup algorithm. More Info...

- No changes are needed in the remaining sections. Click **Save** to save the configuration.

Return to the list of providers (Security Realms/myrealm, Providers tab) and click on the **DefaultAuthenticator**. Change the Control Flag to **SUFFICIENT** and click **Save**.

Settings for DefaultAuthenticator

Configuration Performance Migration

Common Provider Specific

Save

This page displays basic information about this WebLogic Authentication provider. You can also use this page to set the JAAS Control Flag to control how this provider is used in the login sequence.

Name:	DefaultAuthenticator	The name of this WebLogic Authentication provider. More Info...
Description:	WebLogic Authentication Provider	A short description of the Authentication provider. More Info...
Version:	1.0	The version number of the Authentication provider. More Info...
Control Flag:	SUFFICIENT	Returns how the login sequence uses the Authentication provider. More Info...

Save

- Return to the providers list and click **Reorder**. Select AD and use the arrow to move it to the top of the list:

Reorder Authentication Providers

OK Cancel

Reorder Authentication Providers

You can reorder your Authentication Providers using the list below. By reordering Authentication Providers, you can alter the authentication sequence.

Select authenticator(s) in the list and use arrows to move them up and down in the list.

Authentication Providers:

Available:

- AD
- Trust Service Identity Asser
- DefaultAuthenticator
- DefaultIdentityAsserter

OK Cancel

- Click **OK** to confirm the reordering.

The providers list will now show the AD provider first. This will enable EDQ to authenticate against Active Directory.

Restart the Administration Server and login again. Navigate to Security Realms/myrealm and click on the Users and Groups tab. You should see users and

groups loaded from Active Directory. If these are not present, check the Administration Server logs for errors and recheck the configuration.

If there is an excessive delay displaying the Users and Group tab and a size limit error is present in the logs:

```
<Administration Console encountered the following error:
java.lang.RuntimeException: netscape.ldap.LDAPException: error result (4);
Sizelimit exceeded
```

Then there are too many users in Active Directory to be displayed in the Administration Console. This will not cause problems with EDQ but you can reduce the number by entering an LDAP search filter in the All Users Filter field on the AD Provider Specific tab.

4.2.4 EDQ Configuration

This section provides the procedure to configure EDQ.

4.2.4.1 User Group

To define the group containing EDQ users, follow the steps below:

1. Logon to the server hosting the EDQ domain and navigate to the EDQ 'local' configuration directory.

This is located at: *DOMAIN_HOME/config/fmwconfig/edq/oedq.local.home*.

2. Create a local security directory and copy the default login configuration to this directory:

```
$ mkdir security
$ cp ../oedq.home/security/login.properties security/
```

3. Edit the new local configuration at *security/login.properties* and add the line:

```
opss.prof.defaultusergroup = edqusers
```

4.2.4.2 Permissions

The pre-defined group mapping in *login.properties* maps the external 'Administrators' group to the EDQ 'Administrators' group. This is a valid mapping when using the internal WebLogic user store because the internal Administrators group contains the standard 'weblogic' administration console user.

However the Administrators group in Active Directory contains domain level administrators and it is unlikely that users in this group will use or administer EDQ. To add additional group mappings from external EDQ relevant groups using the EDQ web console, it is necessary to log in to EDQ as an administrator. This will be possible with the default pre-defined group mapping only an Active Directory administrator is available. To allow other users to become EDQ administrators, there are two approaches:

- Edit the pre-defined mapping:

Edit the local *login.properties* and change the *xgmap* line to map a different LDAP group to the EDQ Administrators group. For example if a group named 'edqadmins' has been created in Active Directory to contain EDQ administration users, edit the line to read:

```
opss.xgmap = edqadmins -> Administrators
```

- Enable the 'internal' realm:

Edit the local *login.properties* and change the *realms* line to read:


```
realms = internal, opss
```

This will enable the EDQ internal user store and you can log in as user 'dnadmin' and setup the necessary external group mappings using the web console. Once the mappings are complete you can disable the internal realm again.

Once these changes are complete you can start the EDQ managed server and login in as an Active Directory user. Note that external group mappings must be complete to grant the necessary permissions before application logins can succeed.

4.2.5 Filtering Groups

LDAP stores in large organizations may include thousands of distinct groups and with no additional filtering, all of these will appear in the external group mapping list on the EDQ web console. To filter the group list to a manageable size, you can add an LDAP group filter to *login.properties*.

Ensure to make a copy of this in the local configuration area before adding the filter. To add the filter, edit and add the line:

```
opss.prof.groupfilter = LDAPGROUPFILTER
```

where *LDAPGROUPFILTER* is an LDAP-style filter to select groups. The filter uses generic OPSS attribute names instead of LDAP-specific attributes. Use 'name' to refer to the name of the group. The filter must return any groups used in pre-defined mappings.

In the EXAMPLE.COM setup all the groups used with EDQ start with the prefix 'edq' so the filter would be:

```
opss.prof.groupfilter = (name=edq*)
```

4.2.6 Using SSL to connect to LDAP

If you wish to secure connections to the LDAP server by using SSL, tick the SSL Enabled check box on the Provider Specific tab for the LDAP provider, and enter the SSL port (normally 636).

Note: In current versions of WebLogic, if you make changes to the Provider Specific page after initial configuration, you will need to enter the LDAP password again. The original password is obfuscated but unfortunately this version is then saved.

Then to enable successful connections from WebLogic to the LDAP server, so that the list of users and groups can be displayed, and you can login to WebLogic as an LDAP user, you will need to add the LDAP server certificate or root CA to the trust store of the JRE used to run WebLogic. Use the Java keytool command to update the JRE's cacerts file. Documentation for keytool can be found at:

<https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html>

Unfortunately the OPSS library used by EDQ for WebLogic authentication does not use the cacerts store. Instead a new keystore must be created. Stop the EDQ managed server but leave the Administration server running. Login to the server running WebLogic and set these environment variables:

```
JAVA_HOME=java install location
WL_HOME=WLSHOME/wlserver
ORACLE_HOME=WLSHOME
```

and then execute:

```
$ WLSHOME/oracle_common/bin/libovdconfig.sh -host HOST -port PORT -userName  
weblogic -domainPath DOMAIN_HOME -createKeystore
```

Here WLSHOME is the WebLogic installation directory; HOST is the name of the host running the Administration Server, PORT is the Administration port (typically 7001) and DOMAIN_HOME is the location of the WebLogic domain.

The command will prompt for the password for the weblogic user and then for a password for the new keystore. This can be anything.

On successful completion, a new keystore is created at:

```
DOMAIN_HOME/config/fmwconfig/ovd/default/keystores/adapters.jks
```

Use the Java keytool to add the server or CA certificate to the newly created keystore:

```
$ keytool -import -keystore DOMAIN_  
HOME/config/fmwconfig/ovd/default/keystores/adapters.jks -alias ALIAS -file  
CERTFILE
```

Replace **ALIAS** with the keystore identifying the certificate and replace **CERTFILE** with the location of the certificate file.

The command will prompt for the keystore password; enter the password you specified when the keystore was created.

Restart the EDQ managed server and LDAP users should be able to login. Traffic between WebLogic and LDAP will be protected by SSL.

Configuring External User Management (LDAP) directly with EDQ

This chapter provides information on integration to LDAP with EDQ. It includes the following sections:

- [Integrate EDQ with LDAP](#)

5.1 Integrate EDQ with LDAP

This section consists of pre-requisites and procedure to integrate EDQ with LDAP.

If you are not using WebLogic, or wish to use more than one LDAP store, you can integrate EDQ directly with LDAP using settings in *login.properties*. This may also be convenient if WebLogic does not provide a pre-defined Authenticator for your type of LDAP server. For example, AD-LDS, the lightweight LDAP server provided on Windows, is not compatible with Active Directory and you cannot use the WebLogic **ActiveDirectoryAuthenticator**.

This section covers the settings required in *login.properties* for simple LDAP integration. Kerberos support, used for Windows integrated authentication, is covered in [Appendix A, "Configuring EDQ to support Windows Integrated Authentication \(Kerberos\)"](#).

5.1.1 Prerequisites

Before beginning the configuration process, the following information must be gathered:

- The type of LDAP server in use. EDQ has built-in profiles for Active Directory, Oracle Internet Directory, OpenLDAP (using the **inetOrgPerson** user schema) and servers using the RFC 2307 (Posix) style user schema.
- The host name and port of the LDAP server. If your environment contains multiple servers for high availability, you can use more than one host in the configuration. The default LDAP port is 389.
- The identity and password of an LDAP user which can connect and perform searches. The user identity is normally a full Distinguished Name (DN) but Active Directory also allows shorter forms.
- The base DN of the LDAP tree. Where possible this is determined automatically from the LDAP RootDSE **defaultNamingContext** attribute, but if this is not available, it must be provided in *login.properties*.

- The name of an LDAP group containing all the users that will be working with EDQ. The group is used to filter the list of users presented for EDQ issue assignment, etc. Without this filter, every user in the LDAP server would be presented, and this is generally not recommended.

5.1.2 Integrating with Active Directory

The rest of this section will cover the detailed steps for an integration with Active Directory. The steps are broadly similar for other types of LDAP server. Initial configuration will not use SSL.

In the walkthrough, these settings are used:

- **Active Directory Domain:** EXAMPLE.COM
- **Domain Controller (LDAP server):** dc1.example.com. The default, non-SSL, port 389 will be used.
- **LDAP user:** *cn=netuser,cn=users,dc=example,dc=com*. Assuming that the AD username for this user is 'netuser' then you can also use netuser@example.com or example\netuser.
- **Base DN:** *dc=example,dc=com*. When using Active Directory this is determined automatically from the RootDSE.
- **All EDQ users group:** *edqusers*

Note: Ensure that you have a *login.properties* in the security directory in the EDQ 'local' configuration area. Tomcat installs do not have a default *login.properties* so just create a new empty file.

- **Global Settings:**

In this example the Active Directory realm will be named 'ad'.

```
realms                = internal, ad
gss                   = false
ldap.prof.useprimarygroup = false
```

Define the realm list. The EDQ internal realm is used so that the built-in administrator user 'dnadmin' can be used to set up external group mappings in the EDQ web console. The final setting indicates that the 'primary group' of the user should not be considered in determining group membership. For Active Directory this is always the 'Domain Users' group, containing every user and would not be used for EDQ.

- **Realm name:**

```
ad.realm = EXAMPLE.COM
```

Set the name of the Active Directory domain. If more than one realm is defined in *login.properties*, a dropdown selector for the realm is present on the EDQ login screens. If you wish this to contain more explanatory text, add a 'label' property:

```
ad.label = Example, Inc. Active directory
```

- **LDAP server settings:**

```
ad.ldap.server = dc1.example.com
ad.ldap.auth   = simple
ad.ldap.user   = cn=netuser,cn=users,dc=example,dc=com
ad.ldap.pw     = <password for the user>
```

Set the server name and user information. If the server is omitted EDQ attempts to determine it automatically by a DNS lookup for ldap service records. If the server

running EDQ is configured to use the Active Directory domain controllers for DNS, then these records should be setup automatically. Otherwise specify the server in *login.properties*. Multiple servers can be specified using a comma or space separated list:

```
ad.ldap.server = dc1.example.com, dc2.example.com:2342
```

To configure encryption for connections between EDQ and LDAP, add:

```
ad.ldap.security = ssl or tls
```

If you specify `ssl` as the value then a connection is made to the SSL port of the LDAP server (normally 636). In this case the certificate used by the LDAP server must be trusted by the JRE running EDQ. If the certificate was not created by a recognized provider, add the server or Root CA certificate to the JRE cacerts trust store using the `keytool` command. See:

<https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html>

If you specify `tls` as the property value then a connection is made to the normal non-SSL LDAP port and a StartTLS command is then issued to switch the connection to use TLS encryption. All sensitive information such as passwords is sent after the switch to TLS. In this case, EDQ applies relaxed checks to the server certificate - it need not be trusted by the JRE. If you are concerned about spoofing on the internal network, use SSL for extra checks; otherwise using TLS simplifies the configuration.

- **Configure User Authentication:**

This set of properties defines how a username and password presented to EDQ are verified in LDAP. This is a two-stage process. First a search is done to locate the user record identified by the username. For Active Directory this is done by searching for a user with the **sAMAccountName** matching the username or the **userPrincipalName** matching `username@DOMAIN`. These Active Directory attributes represent the 'short' and 'long' Windows login names.

If the user record is not found, authentication does not succeed. Then there are three different methods in which the password can be verified:

- **bind:** Attempt to connect to LDAP using the discovered user identity and the password. If this does not succeed, authentication fails.
- **password:** read the hashed password from the user record and compare it with the supplied password. Most LDAP servers do not allow user password values to be read and this method is rarely applicable. It is sometimes used with Posix style user schemas.
- **compare:** Use an LDAP compare operation to check the password. This method can be used with Oracle Internet Directory and can be preferable to 'bind' because it does not involve creating a new session on the LDAP server.

The default method is 'bind' and this is always used with Active Directory.

```
ad.auth          = ldap
ad.auth.bindmethod = simple
ad.auth.binddn   = search: dn
```

The 'auth' property sets LDAP-based authentication; other settings are possible if using advanced methods such as Kerberos. The 'bindmethod' property sets the connection method - 'simple' sends the user name and password in clear. The 'binddn' property sets the identity used in the connection attempt; here the DN (Distinguished Name) found from the initial user search is used.

An alternative bind method is `digest-md5` which sends the password in hashed form rather than clear. To use this method, replace the bind properties with:

```
ad.auth.bindmethod = digest-md5
ad.auth.binddn     = search: sAMAccountName
```

The Windows user name is used for the connection instead of the Distinguished Name - this is required for digest-md5 connections.

To use LDAP compare for password verification, with Oracle Internet Directory, use these settings:

```
oid.auth          = ldap
oid.auth.method   = compare
```

- **LDAP profile:**

These settings define the LDAP schema in use and customise user lookup:

```
ad.ldap.profile           = adsldap
ad.ldap.prof.defaultusergroup = edqusers
```

The 'adsldap' profile is used with Active Directory. Other pre-defined profiles are:

- inetorgoidldap: Oracle Internet Directory, with **inetOrgPerson** users
- inetorgopenldap: OpenLDAP, with **inetOrgPerson** users
- rfc2307ldap: RFC 2307 (Posix) style user records

The 'defaultusergroup' property defines the group containing all EDQ users.

LDAP stores in large organizations may include thousands of distinct groups and with no additional filtering, all of these will appear in the external group mapping list on the EDQ web console. To filter the group list to a manageable size, you can add an LDAP group filter:

```
ad.ldap.prof.groupsearchfilter = LDAPGROUPFILTER
```

where **LDAPGROUPFILTER** is an LDAP-style filter to select groups. The filter must return any groups used in pre-defined mappings.

In the EXAMPLE.COM setup all the groups used with EDQ start with the prefix 'edq' so the filter would be:

```
ad.ldap.prof.groupsearchfilter = (cn=edq*)
```

- **Putting it all together:**

This is the complete *login.properties* used for the Active Directory integration example:

```
# EXAMPLE.COM LDAP integration
# -----

realms                = internal, ad
gss                   = false
ldap.prof.useprimarygroup = false

ad.realm              = EXAMPLE.COM
ad.ldap.server        = dc1.example.com
ad.ldap.auth          = simple
ad.ldap.user          = cn=netuser,cn=users,dc=example,dc=com
ad.ldap.pw            = <password for the user>
ad.auth               = ldap
ad.auth.bindmethod    = simple
ad.auth.binddn        = search: dn

ad.ldap.profile       = adsldap
ad.ldap.prof.defaultusergroup = edqusers
```

- **Integrating with more than one LDAP store:**

To integrate with multiple LDAP stores, simply define multiple realms and include a block of settings for each:

```
realms = internal, ad1, ad2, oid
ad1.realm = ...
...
ad2.realm = ...
...
oid.realm = ...
...
```

If you are using an Active Directory Forest with multiple domains, each domain must be defined as a separate realm in *login.properties* - EDQ cannot make cross-domain references. For example:

```
realms = internal, aduk, asus
aduk.realm      = UK.EXAMPLE.COM
aduk.ldap.server = DC1.UK.EXAMPLE.COM
...
adus.realm      = US.EXAMPLE.COM
adus.ldap.server = DC1.US.EXAMPLE.COM
...
```

Configuring EDQ to support Windows Integrated Authentication (Kerberos)

The appendix contains information on configuring EDQ to work with Kerberos and Active Directory. It has the following sections:

- [EDQ running as Windows service using local system account](#)
- [EDQ running on Unix](#)
- [Kerberos Shared Libraries](#)

An integration of EDQ with Active Directory using *login.properties* (see [Chapter 5, "Configuring External User Management \(LDAP\) directly with EDQ"](#)) can be configured to support single sign on - SSO - in which the user logs into Windows and then does not need to log again into EDQ. This is supported on both WebLogic and Tomcat.

To enable SSO, the EDQ server must be set up to enable Kerberos authentication from the client PC. This authentication is achieved using the standard GSSAPI token exchange mechanism (RFC 4121). The client contacts the domain controller (DC) to request access to a service provided by the server application. The response from the DC is encoded into a token sent to the server by the client. The server validates this token and generates another token to send to the client. The token exchange can continue until client and server have established a secure context. In practice this exchange never requires more than one token in either direction.

At start up time the server application sets up 'accept' credentials which it uses to initialize its half of the security context.

A.1 EDQ running as Windows service using local system account

If the EDQ application server is running on a Windows server in the domain, using the local system account, then the configuration is very simple. EDQ will use the system account for the accept credentials and also to contact AD for user lookups.

The EXAMPLE.COM *login.properties* for this configuration would be:

```
# EXAMPLE.COM LDAP integration
# -----

realms                = internal, ad
ldap.prof.useprimarygroup = false
clientcreds           = true

ad.realm              = EXAMPLE.COM
ad.auth               = ldap
```

```
ad.auth.bindmethod      = simple
ad.auth.binddn          = search: dn
```

```
ad.ldap.profile         = adsldap
ad.ldap.prof.defaultusergroup = edqusers
ad.ldap.prof.groupsearchfilter = (cn=edq*)
```

The 'clientcreds' setting indicates that Kerberos credentials should be obtained from the current user's cache (in this case the local system account). These credentials are used to connect to Active Directory and to set up the 'accept' GSSAPI context.

A server which is a member of the Active Directory domain will generally use the domain controller for DNS lookups. If this is the case, EDQ can determine the LDAP server addresses automatically. If you wish to fix the address, perhaps because some of the domain controllers are at a remote location, use the `ldap.server` property:

```
ad.ldap.server          = dc1.example.com
```

A.2 EDQ running on Unix

If the server is running on Unix, it must use an account in AD to set up the accept credentials. It validates the request using the encrypted account password read from a Kerberos key table (keytab). Setting up a valid key table is an essential step in configuring SSO on Unix.

A.2.1 What is in the keytab?

A Kerberos key table contains encrypted passwords for one or more Kerberos principals. The DC normally supports a number of different encryption algorithms (DES3, AES, RC4 etc) and the entry for a principal will include keys for each of these algorithms. The client will pick the best algorithm available for communication with the DC.

The service requested using GSSAPI is identified by a service principal name (SPN). Normally this will be a reference to a particular service type at a machine hostname. Examples of service types are **HOST** (for general access such as ssh), **HTTP** (for SSO from browsers) and **LDAP** (for LDAP servers such as AD domain controllers). An SPN is usually displayed as:

```
service/hostname
```

For example:

```
HOST/testserver.example.com
```

Each entry in a keytab also includes a 'key version number' (KVNO). This is a version number which is incremented whenever the password for the principal is changed in the DC. The keytab must contain the correct KVNO for authentication to succeed.

On most Unix systems, the default location of the system Kerberos keytab is:

```
/etc/krb5.keytab
```

The Java Kerberos implementation does not have a default keytab location and this must be set in *login.properties*:

```
keytab = path to keytab
```

If the path is not absolute, it is relative to the security folder containing *login.properties*.

The *klist* command can be used to list the contents of a keytab:

```
klist -k [file]
klist -ke [file]
klist -keK [file]
```

A file name can be provided if the keytab is not in the default location. The first form just lists the principals; the second also includes the encryption algorithms and the third also includes the key values in hexadecimal.

Here's some example output from *klist -ke*:

```
Keytab name: FILE:/etc/krb5.keytab
KVNO Principal
```

```
-----
 5 ALTAIR$@EXAMPLE.COM (des-cbc-crc)
 5 ALTAIR$@EXAMPLE.COM (des-cbc-md5)
 5 ALTAIR$@EXAMPLE.COM (des3-cbc-sha1)
 5 ALTAIR$@EXAMPLE.COM (aes128-cts-hmac-sha1-96)
 5 ALTAIR$@EXAMPLE.COM (aes256-cts-hmac-sha1-96)
 5 ALTAIR$@EXAMPLE.COM (arcfour-hmac)
 5 HOST/altair@EXAMPLE.COM (des-cbc-crc)
 5 HOST/altair@EXAMPLE.COM (des-cbc-md5)
 5 HOST/altair@EXAMPLE.COM (des3-cbc-sha1)
 5 HOST/altair@EXAMPLE.COM (aes128-cts-hmac-sha1-96)
 5 HOST/altair@EXAMPLE.COM (aes256-cts-hmac-sha1-96)
 5 HOST/altair@EXAMPLE.COM (arcfour-hmac)
 5 HOST/altair.EXAMPLE.COM@EXAMPLE.COM (des-cbc-crc)
 5 HOST/altair.EXAMPLE.COM@EXAMPLE.COM (des-cbc-md5)
 5 HOST/altair.EXAMPLE.COM@EXAMPLE.COM (des3-cbc-sha1)
 5 HOST/altair.EXAMPLE.COM@EXAMPLE.COM (aes128-cts-hmac-sha1-96)
 5 HOST/altair.EXAMPLE.COM@EXAMPLE.COM (aes256-cts-hmac-sha1-96)
 5 HOST/altair.EXAMPLE.COM@EXAMPLE.COM (arcfour-hmac)
```

In a normal Kerberos system using a standard Kerberos Domain Controller (KDC) each SPN is a separate principal with a different password. In Active Directory, SPNs are essentially 'aliases' of a single account, stored as values of the AD **servicePrincipalName** LDAP attribute. When a computer account is created in AD, SPNs for the **HOST** service are created automatically. If additional services such as IIS or SQLserver are installed on the server, additional SPNs will be added to the account.

The Windows `setspn` command can be run on an AD server to manage the SPNs for an account. For example:

```
setspn -A HTTP/altair.example.com altair$
```

The first command adds an HTTP SPN to the machine account for altair. (The Windows account name for a computer always ends with \$).

A.2.2 Creating keytabs using existing tools

In a normal Kerberos system, keytab entries are created using the `ktadd` subcommand of the Kerberos administration tool *kadmin*. AD does not provide a Kerberos administration server so other approaches are required.

The keytab contains the encrypted password for the account so for each method either the password for the account must be known in advance, or it must be run with privileges to change the account password.

The method to use depends on the system configuration. Some existing options are:

- **Samba**: If the system has been registered with AD using the Samba suite, the `net ads keytab` command can be used to create and update the keytab. This works because Samba has set the password for the account and stored it in a secret location.
- **ktpass**: The Windows `ktpass` command can run by an AD administrator to generate keytab entries. Unless there is no other alternative, do not use this

command. It is complex and very difficult to use reliably. It will update the password of the account, thus rendering any previous keytab useless.

- **msktutil**: This is an open source application for Unix which can be used to manage keytabs.

A.2.3 Creating keytabs using winktab

winktab is a Java application which can be used to create a keytab for an account in AD. It contacts AD to determine the KVNO for the account and to determine the SPNs associated with the account. If run with an administrator account, it can reset the account password to a random value; alternatively the account password can be supplied to the command.

winktab can be run on any system which can connect to the AD server.

To create a keytab for a machine account and reset the password to a random value:

```
java -jar stuff.jar winktab -o ktfile -domain DOMAIN -server adserver
                        -user aduser -pw adpw -tls machinename
```

To create a keytab for a machine account using a known password:

```
java -jar stuff.jar winktab -o ktfile -accpw accpw -domain DOMAIN -server adserver
                        -user aduser -pw adpw [-tls] machinename
```

The parameters are:

- **ktfile**: the keytab is written to this file
- **accpw**: the password for the machine or user account; if a single dash (-) is used the command will prompt for the password without echo
- **DOMAIN**: the AD domain name
- **adserver**: the host name or IP address of an AD server for the domain
- **aduser, adpw**: the user name and password of an AD account which is used to connect to the server for queries. The user name must be:

user@DOMAIN

orSHORTDOMAIN\user

If a single dash (-) is given as the password the command will prompt for it without echo.

If the account password is set to a fixed (**-setpw** used) or random (**-setpw** and **-accpw** omitted), the user must have administrator privileges.

- **machinename**: the name of a computer account. The internal AD account name will be **machinename\$**. The internal AD account name is stored in the **sAMAccountName** LDAP attribute.

Use the **-tls** switch if the AD server requires authenticated connections. Windows requires an encrypted connection if an account password is being reset so **-tls** must be used if the password is being reset to a random or known value.

Example for the domain EXAMPLE.COM, split over several lines for clarity:

```
java -jar stuff.jar winktab -o altair.keytab -setpw altair \
                        -domain EXAMPLE.COM -server dc1.example.com \
                        -user "example\Admin" -tls altair
```

Create a keytab for a machine account, setting the password to a known value and prompting for the administrator password.

A.2.4 Check the Unix Kerberos configuration

The Kerberos configuration used by commands such as *kinit* and the Java runtime is read from a global configuration file, normally stored at */etc/krb5.conf*. This contains references to the domain controllers and mappings between DNS and Kerberos domains. Here's an example for the domain EXAMPLE.COM:

```
[logging]
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log

[libdefaults]
default_realm = EXAMPLE.COM
dns_lookup_realm = false
dns_lookup_kdc = false
ticket_lifetime = 24h
renew_lifetime = 7d
forwardable = yes

[realms]
EXAMPLE.COM = {
    kdc = dc1.example.com:88
}

[domain_realm]
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
```

The *[realms]* section lists the KDCs by host or IP for each domain; the *[domain_realm]* section maps DNS host names to Kerberos domains.

krb5.conf must be checked and adjusted for the configuration of the target domain. If it is not possible to update a file in */etc*, store the file elsewhere and inform the Java runtime of the location via a system property. To do this, edit or create the file *jvm.properties* in the EDQ local configuration directory and add the line:

```
java.security.krb5.conf = absolute path to modified krb5.conf
```

A.2.5 Java Encryption

Java runtime environments do not ship with support for high-strength ciphers by default. For example AES with 256-bit keys is not supported. Active Directory and Kerberos support a full set of ciphers and it may be that your key table contains entries not supported by Java. In this case you should install the JCE Unlimited Strength Jurisdiction Policy Files, if permitted in your location.

A.2.6 Changes to login.properties

In this example, the server running EDQ is assumed to be **altair.example.com**.

- **Global Settings:**

```
realms                = internal, ad
spn                   = HOST/altair.example.com@EXAMPLE.COM
keytab                = /etc/krb5.keytab
ldap.prof.useprimarygroup = false
```

The 'spn' property sets the Service Principal to use for the GSSAPI accept context. The value shown here is the default and can be omitted if the Unix server's DNS domain the same as the AD domain. Sometimes Unix servers are in a separate DNS domain and the 'spn' property is then required.

The 'keytab' property sets the location of the Kerberos key table. Java does not default this to the standard location and this property is always required.

- **LDAP server settings:**

```
ad.ldap.server = dc1.example.com
ad.ldap.spn    = "ALTAIR$@EXAMPLE.COM"
```

Set the server name and Kerberos principal used to connect to Active Directory. The principal should be the key table entry containing the actual machine account name. As before the server can be omitted if it can be determined from DNS.

- **Putting it all together:**

This is the complete *login.properties* used for the Active Directory + Kerberos integration example:

```
#EXAMPLE.COMLDAPintegrationwithKerberos

realms          = internal, ad
spn             = HOST/altair.example.com@EXAMPLE.COM
ldap.prof.useprimarygroup = false
keytab          = /etc/krb5.keytab

ad.realm        = EXAMPLE.COM
ad.label        = Example, Inc.Activedirectory
ad.ldap.server  = dc1.example.com
ad.ldap.spn     = "ALTAIR$@EXAMPLE.COM"
ad.ldap.security = tls
ad.auth         = ldap
ad.auth.bindmethod = simple
ad.auth.binddn  = search: dn

ad.ldap.profile = adslldap
ad.ldap.prof.defaultusergroup = edqusers

ad.ldap.prof.groupsearchfilter = (cn=edq*)
```

With this *login.properties* in place, a Windows user in the EXAMPLE.COM domain should be able to login to an EDQ application without supplying additional credentials. The user must of course have the permission in EDQ to use the application.

[7] Kerberos Shared Libraries

The shared libraries (*wingss.dll* and *libunixgss.so*) required for Kerberos integration are shipped inside the *edq.war* file. For most installations this is sufficient since EDQ can determine the location of the shared libraries and load the right version automatically.

However, this automatic loading does not work with all Java Runtime Environments (JREs), and notably it does not work with the IBM JRE. For these installations the libraries need to be extracted from the provided *kerberos-gss.zip* file and copied to a known location on disk. The location must then be added to the following environment variables such that the JRE can find it:

- LD_LIBRARY_PATH (Linux, Solaris)
- LIBPATH (AIX)
- PATH (Windows)

Examine the *native/Kerberos-gss.zip* archived provided with the EDQ install and verify it contains the following files:

- aix/ppc/libunixgss.a
- aix/ppc64/libunixgss.a
- linux/amd64/libunixgss.so
- linux/i386/libunixgss.so
- win32/amd64/wingss.dll
- win32/x86/wingss.dll

Extract the relevant library for the OS the EDQ server runs on, and copy it to a location on a disk accessible by the user EDQ runs as. This location needs to be added to the environment variables mentioned above, so the JRE can find the library.

B

Configuring Single Sign On with Oracle Access Manager (OAM)

When EDQ is integrated with Oracle Access Manager, a user can login on a common access page and have automatic access to EDQ applications and the web console without additional Logins (assuming of course that the user has the required EDQ permissions). If there are multiple EDQ installations using the same OAM configuration, the login will work for each.

This section covers the configuration steps to integrate EDQ with OAM. It does not cover installation and basic configuration of OAM or installation of the Web Tier front end (OHS). This appendix contains the following sections:

- [Prerequisites](#)
- [OAM configuration](#)
- [WebLogic plugin configuration](#)
- [WebLogic Configuration](#)

B.1 Prerequisites

The following are the prerequisites for installing OAM:

- OAM must be configured with an Authentication Scheme using an identity store supported by WebLogic (typically LDAP - Active Directory or Oracle Internet Directory).
- WebLogic must be configured to authenticate EDQ using the same identity store. See [Chapter 4, "Integrating External User Management \(LDAP\) using WebLogic and OPSS"](#). This should be configured and tested with EDQ before proceeding with the OAM integration steps.
- A web server front end (OHS or Apache) must be installed and configured with Webgate software and the WebLogic plugin (mod_wl_ohs). These are bundled with OHS 12 releases.

B.2 OAM configuration

To configure OAM, follow the steps below:

1. Create a Webgate in OAM using the authentication schema which refers to the identity store configured in WebLogic.

Create these HTTP resources in the Webgate:

Table B-1 Creating HTTPS resources in the Webgate

RESOURCE	POLICY
/edq/faces/**	Protected Resource Policy
/edq/blueprints*/jnlp	Protected Resource Policy
/edq/**	Public Resource Policy (or Excluded)

2. Copy the Webgate artefacts to your OHS installation and place in the *webgate/config* directory.

B.3 WebLogic plugin configuration

Ensure that the WebLogic plugin (`mod_wl_ohs`) is configured in the web server front end. Add this entry to the plugin configuration file (normally `mod_wl_ohs.conf`):

```
<Location /edq>
  SetHandler weblogic-handler
  WebLogicPort managed server port
  WebLogicHost hostname
</Location>
```

If you are using a WebLogic cluster, replace the host and port settings with a cluster definition:

```
WebLogicCluster host1:port1, host2:port2, ...
```

Ensure that the WebLogic Plug-In enabled option is set for the EDQ servers. This can be done at the domain, cluster, server template or server level. For the domain the option is present in the Configuration/Web Applications tab. For the other items the option is present in the Advanced area of the General Configuration tab.

B.4 WebLogic Configuration

To configure WebLogic, follow the steps below:

1. Login to the Administration Console and navigate to Security Realms/myrealm.
2. Click the **Providers** tab.
3. Click **New** to add a new provider.
4. Enter a name, for example, OAM and select the **OAMIdentityAsserter**.

Create a New Authentication Provider

OK | Cancel

Create a new Authentication Provider

The following properties will be used to identify your new Authentication Provider.

* Indicates required fields

The name of the authentication provider.

* **Name:**

This is the type of authentication provider you wish to create.

Type: ▼

OK | Cancel

5. Click **OK**.

The new, unconfigured provider is added to the bottom of the list.

Note: There is no need to restart the Administration Server at this point.

6. Click on the new provider. On the Configuration screen change the control flag to **REQUIRED**.

Settings for OAM

Configuration

Common Provider Specific

Save

This page allows you to define the general configuration of this provider.

Name: OAM

Description: Oracle Access Manager Identity Asserter

Version: 1.0

Control Flag: REQUIRED ▼

Active Types:













Available:		Chosen:
<input type="checkbox"/> OAM_IDENTITY_ASSERTION <input type="checkbox"/> ObSSOCookie <input type="checkbox"/> SM_USER <input type="checkbox"/> iv-user	> >> < <<	<input type="checkbox"/> OAM_REMOTE_USER

Base64 Decoding Required: false

Save

7. Click **Save** and then select the **Provider Specific** tab.

Enter configuration information for OAM. Also, where OAM is configured without Webgate security, just enter the **Access Gate Name** and **Primary Access Server**.

 Access Gate Name:	<input type="text" value="edq"/>
 Minimum Access Server Connections In Pool:	<input type="text" value="5"/>
 Secondary Access Server:	<input type="text"/>
 Transport Security:	<input type="text" value="open"/> 
 Trust Store:	<input type="text"/>
 Key Store:	<input type="text"/>
 Simple Mode Pass Phrase:	<input type="text"/>
 Confirm Credential:	<input type="text"/>
 Maximum Access Server Connections In Pool:	<input type="text" value="10"/>
 Internal Compatibility Mode:	<input type="text"/>
 Primary Access Server:	<input type="text" value="oam.example.com:5575"/>

This information for this tab will be provided by the OAM administrator.

8. Click Save and return to the Providers list.

Use the **Reorder** button to move the OAM provider to the top of the list.

9. Restart the Administration Server and EDQ managed servers. OAM integration is now complete.

