Oracle® Fusion Middleware

Securing Oracle Enterprise Data Quality

12*c* (12.2.1.1.0)

E69492-01

June 2016

Oracle Fusion Middleware Securing Oracle Enterprise Data Quality, 12*c* (12.2.1.1.0)

E69492-01

Contents

Preface

This manual 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.
italic	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.

1 Introducing EDQ Security

This chapter introduces Oracle Enterprise Data Quality Security concepts, and guides you through the content of this manual. It includes the following sections:

- Introducing EDQ Security
- Terms Used in this Guide

1.1 Introducing EDQ Security

Security within EDQ applies to access to the application (ensuring that only authorized users can access it, and that data within the application is secured), and to auditing of user actions to identify anomalies. Section 1.1.1, "Authentication" and Chapter 5, "Authorizing Users" relate to access control, Chapter 6, "Using Encryption" details encryption of data during transport, and Chapter 7, "Auditing" details auditing.

1.1.1 Authentication

Details of users and groups in EDQ can be stored within its own internal directory or taken from an external directory service. EDQ can be integrated with an external Lightweight Directory Access Protocol (LDAP) server (including Microsoft Active Directory) or Oracle Access Manager (OAM). 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, permissions and subsystems (such as *Core*, *Director*, *Server Console*, and others) associated with groups. Users inherit permissions on subsystems from the groups to which they are associated.

1.1.3 Encryption

Both the WebLogic and Tomcat servers support HTTPS and should be configured to require 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.

Where databases support encryption of traffic, connections to the database should be configured to use this feature.

1.1.4 Auditing

EDQ supports auditing of user actions using the Oracle Fusion Middleware Audit Framework. In addition, EDQ can be configured to produce audit information in disk files. See Chapter 7, "Auditing" for more information.

1.2 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.

2

Authenticating Using LDAP/AD Services

This chapter describes how EDQ can be integrated with external user management systems based on the LDAP standard, thus allowing Administrators to manage user accounts externally to EDQ.

This chapter includes the following sections:

2.1 Learning About LDAP Support

LDAP is a standard for accessing distributed directory services such as Active Directory, Novell eDirectory, OpenLDAP or Oracle Internet Directory. LDAP directories store entries (which may include people, groups, organizational units, and other items) in a hierarchical arrangement. EDQ is certified for integration with the following LDAP directories:

- Oracle Internet Directory (OID) 11*g*
- Microsoft Active Directory (AD) for Windows Server 2000, 2003 and 2008
- Open LDAP 2.4
- Novell eDirectory 8.8

In the case of integration with AD, EDQ can support Single Sign-On (SSO), so users who have signed-on to the AD domain can access EDQ without the need to log in separately to the EDQ client applications.

EDQ can be configured to authenticate users against an LDAP directory using either Oracle Platform Security Services (OPSS) configured on Oracle WebLogic Server (see Section 2.1.1, "Import Filtering of Users and Groups"), or connection settings specified within EDQ's own configuration files (for example when hosted by an Apache Tomcat server).

LDAP configuration consists of three parts:

- Global (security/login.properties)
- Realm(s)
- Profile for each realm

The security/login.properties file which contains global security configuration is located within the base configuration directory (that is, oedq_home/security/login.properties). Realms define the LDAP directory or directories to use for authenticating users, and can be defined within the login.properties or realm-specific files (see Section 2.5, "Using LDAP Server Profile"). Profiles are generally chosen from a set of predefined profile configurations (see Section 2.4, "Configuring Global LDAP Settings (login.properties)"), and contain technical details of the LDAP directory structure for the realms.

2.1.1 Import Filtering of Users and Groups

By default LDAP integration imports all users from the LDAP directory. This can be a very large number of records, and it is recommended that where feasible the configuration specifies a group of users to import. Without this, performance of EDQ can be significantly impacted and excessive load placed on the LDAP directory. To specify a group, use the ldap.prof.defaultusergroup option in the ldap.properties file, For example:

```
ldap.prof.defaultusergroup=edqusers
```

See Section 2.6, "Configuring Individual Realm LDAP Settings" for details of the ldap.prof.defaultusergroup option.

2.2 Integrating LDAP Using Oracle Platform Security Services

In a default installation of EDQ on WebLogic Server, EDQ is integrated with Oracle Platform Security Services (OPSS). EDQ users and groups are managed by an OPSS identity store that is configured in WebLogic Server.

Properties in the login.properties file define the default mapping of the LDAP administrators group to the EDQ administrators group. The default configuration looks for a group called “Administrators” and maps it to the EDQ group of the same name, enabling users within that group to access the Administration application on the EDQ Launchpad.

2.2.1 Configuring LDAP Group Mappings

Where the naming of groups in WebLogic Server identity store or the configured LDAP server does not match their corresponding groups in EDQ, mappings can be defined to override the defaults. This can be done in two ways: in the login.properties configuration file or through the Launchpad administration. If the Administrators group requires mapping, create a local login.properties file to override the base login.properties file included with EDQ, and adjust the group mapping in the new file:

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_local_home/security`) to `oedq_local_home/security`.
3. Look for the property *opss.xgmap*, for example:

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

Here *opss* refers to the Oracle Platform Security Services realm, *xgmap* is short for *external group map*.

4. Modify the property with the desired group name mappings. The property accepts a comma-delimited list of mappings from the external group name to the corresponding group in EDQ. For example to map two groups *DQAdministrators* and *DQ Admins* to *Administrators*, you would use:

   ```
   opss.xgmap = DQAdministrators -> Administrators, DQ Admins ->
   Administrators
   ```

5. Save the file.
6. Restart the application server to reload the configuration.

Once administrators can log in to EDQ, other group mappings should be configured through Launchpad administration.

2.2.2 Configuring Server Failover

To configure OPSS to use multiple LDAP servers (that is, for failover if one server is down), see *Configuring Failover for LDAP Authentication Providers*.

2.3 Integrating EDQ Directly with LDAP Servers

EDQ also supports direct integration with LDAP servers without using Oracle Platform Security Services. This would typically be used where EDQ is hosted within an Apache Tomcat server, although it can be used with any container. Direct integration is configured through the login.properties file in the local configuration directory. See Section 2.3, "Integrating EDQ Directly with LDAP Servers" for details on the login.properties file.

To set up direct integration:

1. Navigate to the `security` directory in the EDQ local configuration directory (`oedq_local_home/security`).
2. Open the `login.properties.template` file with a text editor. This template contains sample settings that correspond to the supported LDAP providers.
3. Uncomment and edit the parameters that correspond with the LDAP server in the EDQ installation environment. The profile associated with an LDAP configuration provides information about the schema in the LDAP server that represents users and groups. EDQ provides the following built-in profiles:
   - `adsldap`: Microsoft Active Directory
   - `inetorgoidldap`: Oracle Internet Directory (OID)
   - `inetorgopenldap`: OpenLDAP using `inetOrgPerson` style schemas
   - `rfc2307ldap`: RFC 2307 (that is, Solaris)
4. Save the file as `login.properties` in the same directory.
5. Restart the application server.

Other schemas can be supported by creating new profiles or extending existing profiles.

2.4 Configuring Global LDAP Settings (login.properties)

EDQ supports integration with multiple realms which can use different LDAP servers. For example, a single EDQ server may support external authentication from both a Microsoft Active Directory (AD) realm and an Oracle Internet Directory (OID) realm, if required.

These global settings can be specified in the `security/login.properties` configuration file. Properties are configured using the syntax *`property_name = value`*, for example:

```
realms = realm1, realm2
```

Where noted, you can override the global settings at the realm level. Realm-level settings are more specific and always override global settings. (See Section 2.6, "Configuring Individual Realm LDAP Settings".

Table 2–1 Global LDAP Settings

Property	Description	Example Value	Mandatory?
`realms`	A comma-separated list of realm names, representing active realm configurations. The specified name of each realm must correspond with the realm-specific properties later in the file, in the format *`realm_name.property_name = value`*. A realm configuration may be retained but disabled by removing it from this list.	`realm1`, `realm2`	Yes.
`clientcreds`	If set to `true`, the server uses the credentials of the local machine to connect to the LDAP servers. This is used to enable SSO for AD integrations where the EDQ server is on the domain. If set to `false`, and if SSO is enabled, the server uses the configured keytab. May be overridden at the realm level.	`true`	No. If not set, the default value is `false`.
`x509`	Enables the use of x509 certificates for client authentication over SSL. There is a small performance cost associated with setting this to `true`. May be overridden at realm level.	`true`	No If not set, the default is `false`.
`ldap.prof.useprimarygroup`	Defines whether or not to use the primary group (for example the "Domain Users" group in AD). May be overridden at realm level	`false`	No. This should be set to `false` for performance purposes unless the membership of the primary group has any relevance for EDQ.

2.5 Using LDAP Server Profile

The LDAP server profile specifies how users are stored in the LDAP directory, and is used by EDQ to determine how to look up users and display them.

2.5.1 Default Profiles

EDQ ships with a number of default profiles, which are selected using the ldap.profile property:

- `adsldap`: Microsoft Active Directory
- `inetorgoidldap`: Oracle Internet Directory (OID)
- `inetorgopenldap`: OpenLDAP using `inetOrgPerson` style schemas
- `rfc2307ldap`: RFC 2307 (that is, Solaris)

2.5.2 Properties

LDAP profile properties can be overridden at global or realm level. In almost all cases these can be left at the values specified in the provided profile, with the notable exception of *defaultusergroup* and *groupsearchfilter*, however the properties are documented below for reference.

For properties which take LDAP search filters, the syntax is defined in RFC 4515 *Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters*. See `https://docs.oracle.com/cd/E20295_01/html/821-1222/fnyth.html` for an introduction to the filter syntax.

Table 2–2 LDAP Profile Properties

Property	Description	Example Value
`useprimarygroup`	Defines whether or not to use the primary group (for example the "Domain Users" group in AD) to filter for users. May be overridden at realm level	`false`
`primarygroupfilter`	Filter to apply to users based on their primary group. Only used if useprimarygroup is set to true	
`defaultusergroup`	Canonical name of the default group that contains all EDQ users, used to determine users to display in issue, alert, and case assignment lists.	`edqusers`
`groupsearchfilter`	Additional filter for groups; an LDAP search filter.	`(cn=edq*)` This will include all groups with a canonical name beginning with edq.
`idmatch`	User identity regular expression, which is used to derive a filter to search the directory for the user	`(?i)(.*)@${realm:.*}`
`userfilter`	Search filter to locate a specific user	`+(uid={1})`
`username`	Attribute which stores the login name of a user	`uid`
`primaryuserattr`	Name of the attribute of a user which refers to groups they belong to	

Table 2–2 (Cont.) LDAP Profile Properties

Property	Description	Example Value
`primarygroupattr`	Name of the attribute of a group which primaryuserattr refers to	
`memberattr`	Name of the attribute of a group which identifies its members	`uniquemember`
`membertarget`	Name of the attribute of a user which memberattr refers to	`dn`
`membersearch`	Search filter to locate members of a group	
`userdisplayname`		

2.6 Configuring Individual Realm LDAP Settings

This section provides details of the properties that are normally set at the realm level. Realm settings may be specified with either of the following methods:

- In the `login.properties` file by using the syntax *`realm_name.property_name = value`*. This format enables you to specify settings for different realms within a single file, each set of properties having a different *`realm_name`* prefix.
- In a file named *`realm_name`*`.properties` in a `realms` subdirectory of the `security` directory. This method requires a separate `realm_name.properties` file for each realm that you want to configure. The *`realm_name`* prefix is not needed for properties in the *`realm_name`*`.properties` file.

In a similar manner, profile settings and overrides can be specified in the `security/login.properties` file by using the syntax *`profile_name.property_name = value`* or, alternatively, in separate files named *`profile_name`*`.properties` in the `security/profiles` folder.

Table 2–3 LDAP Settings

Property	Description	Example Value	Mandatory?
`realm`	The LDAP (AD or Kerberos) domain name.	*`EXAMPLE.COM`*	Yes
`ldap.profile`	Specifies the LDAP profile name used to configure parameters using shipped built-in settings. See Section 2.4, "Configuring Global LDAP Settings (login.properties)" for details on LDAP profiles.	`adsldap`	Yes
`auth`	Specifies the user authentication method, if SSO is not used. Possible values are `ldap` or `jaas`. All current certified configurations use `ldap`.	`ldap`	Yes

Table 2–3 (Cont.) LDAP Settings

Property	Description	Example Value	Mandatory?
`auth.method`	If the `auth` property is set to `ldap`, the `auth.method` property specifies which method is used to validate user credentials. See Section 2.7, "Validating Credentials When Single Sign-On Is Not Used" for further information.	`bind`	No
`label`	Specifies an alternative user-friendly label for the realm to display in the dialog when logging into user applications.	`myrealm`	No If not used, the `configured` realm name from the realm property is used.
`gss`	Specifies whether or not the realm supports SSO using Kerberos/GSSAPI. Possible values are `true` or `false`.	`false`	No If not set, defaults to `true`.
`ldap.server`	A comma or space separated list of LDAP servers (either names or IP addresses). Each server listed can include a specific port using the syntax *`server:port`*.	`192.168.1.0:389, server2`	No If not specified, a DNS lookup is used to look for LDAP servers.
`ldap.basedn`	Distinguished name of the base entry of the LDAP hierarchy.	`dc=example, dc=com`	No In many servers (including AD), this can be found from the RootDSE (the Root Directory Service Entry).
`ldap.security`	Sets the security mode for LDAP connection. Possible values are `ssl` or `tls`. See Section 2.8.1, "Using LDAP Over SSL/TLS" for details.	`tls`	No
`ldap.auth`	Sets the authentication mode for LDAP connection. Possible values are `simple`, `digest-md5` or `gss`.	`digest-md5`	No If not specified, this defaults to `gss`.
`ldap.user`	The LDAP username used to authenticate EDQ with the LDAP server. This property must be set if `ldap.auth` is not set to `gss`.	`cn=user, ou=users, dc=example, dc=com`	Yes, if authentication mode is `simple`. No, if the mode is `digest-md5`.
`ldap.pw`	The password associated with the LDAP username.	`password`	Yes, if authentication mode is `simple` or `digest-md5`. No, otherwise

Table 2–3 (Cont.) LDAP Settings

Property	Description	Example Value	Mandatory?
`ldap.clientcreds`	Specifies how the EDQ server connects to the LDAP server. If set, this parameter overrides the `clientcreds` parameter in the `login.properties` file. See Section 2.4, "Configuring Global LDAP Settings (login.properties)" for more information.	`true`	No. If not set, the default value is the one specified at the Global level.
`ldap.prof.defaultusergroup`	Canonical name of the default group that contains all EDQ users, used for display of users in issue, alert, and case assignment lists.	`edqusers`	Recommended If not set, this defaults to `Domain Users` on AD, which may be too large and cause display and memory issues.
`ldap.prof.groupsearchfilter`	Additional filter for groups; an LDAP search filter. See Section 2.4, "Configuring Global LDAP Settings (login.properties)" for more information.	`(cn=edq*)` This will include all groups with a name beginning with `edq`.	Recommended If not set, no filter is used and all groups will be displayed on the External Groups configuration page.

LDAP server profile settings can be overridden in the LDAP realm settings by specifying the property with the prefix *`ldap.prof`*, for example: *`ldap.prof.defaultusergroup= group=edqusers,ou=groups,dc=example,dc=com`*.

2.7 Validating Credentials When Single Sign-On Is Not Used

In installations where Single Sign-On (SSO) is not used and the `auth` realm property is set to `ldap`, it is necessary to set the `auth.method` realm property to specify how user credentials are validated. The possible values for this property are as follows. They are described in the following sections.

`auth.method = bind`

`auth.method = password`

`auth.method = compare`

auth.method = bind

This setting directs the EDQ server to connect to the LDAP server to verify the user credentials. This is the default setting. Where the bind method is used, the following additional properties must be set:

- `auth.binddn`: Specifies the actual user name that is used in the connection attempt. If omitted, a value in the form *`username@realmname`* is submitted. Otherwise, the value should be in the form `search:` *`dn`*, which uses the distinguished name (DN) of the user for the login.
- `auth.bindmethod`: Specifies the authentication method that is used to connect to the LDAP server. The possible values are `simple` or `digest-md5`. The `digest-md5` value encrypts the password on the network and is the recommended setting.

Note: If `auth.bindmethod` is set to `digest-md5` for an EDQ installation that is integrated with Active Directory, the `auth.binddn` property must be set to `search: sAMAccountName`.

auth.method = password

This setting directs the EDQ server to look up the user record on the LDAP server and then compare the submitted password to the stored password.

Note: This method cannot be used with Active Directory servers.

The LDAP attribute that stores the password must be specified with the following property:

```
auth.password = search: attr
```

where: *`attr`* is the LDAP attribute.

auth.method = compare

This setting uses the LDAP compare method to validate the password. This method is more secure than `auth.method = bind` because a session is not created in the LDAP server.

The LDAP attribute that stores the password is specified with the `auth.attribute` property, which has a default value of `userPassword`. This default is the correct value for Oracle Internet Directory LDAP integrations.

2.8 LDAP Security

The ldap.security option controls security of the connection to the LDAP server. It is a comma separated list of options, which can be:

- `ssl`: Use LDAP connections over SSL/TLS
- `tls`: Use Start TLS extension to upgrade LDAP connections to encryption
- `qop=none`
- `qop=auth`
- `qop=auth-int`
- `qop=auth-conf`

By default, traffic to and from the LDAP server is unencrypted. LDAP traffic can be encrypted either by sending it over a Secure Sockets Layer (SSL) connection (LDAPS), or by using the Start TLS extension.

2.8.1 Using LDAP Over SSL/TLS

LDAP over SSL/TLS (LDAPS) establishes a secure connection to the LDAP server, and then sends LDAP traffic over it. This requires that the remote server presents a valid X.509 certificate. Specifically, the certificate's canonical name must match the host name of the server, and must be trusted or signed by a certificate authority (CA) trusted by the Java Runtime Environment (JRE) that is running EDQ.

Where an LDAP server presents an X.509 certificate that is either self-signed or signed by a certificate authority which is not part of the trusted defaults, EDQ must be configured to recognise these certificates.

2.8.1.1 Extracting Certificate from Active Directory

To connect over SSL/TLS to an Active Directory server without a certificate signed by a known certificate authority (typically a self-signed certificate), the certificate must be extracted from Active Directory and then imported into the JRE trusted certificates key store. The certificate can be extracted using Certutil (see https://technet.microsoft.com/en-in/library/cc732443.aspx for details), for example:

```
certutil -ca.cert certificate.crt
```

The extracted certificate then needs to be imported into the cacerts keystore belonging to the JRE, by following the instructions in Section 2.8.1.2, "Importing Certificates into JRE (for Tomcat)".

2.8.1.2 Importing Certificates into JRE (for Tomcat)

The JRE includes a trusted certificate keystore, which can be found at the path `lib/security/cacerts` within the JRE installation folder. The keytool utility (provided with the JRE, see https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html) can import new certificates to this file, for example:

```
keytool -import -keystore lib\security\cacerts -file certificate.crt
```

At the prompts, provide the password, change it (note that you should not actually change this password), and confirm that you wish to trust the imported certificate.

2.8.1.3 Importing Certificates into OPSS (for WebLogic)

For instructions on managing the key/certificate stores used by WebLogic, see *Fusion Middleware Administrator's Guide*. For instructions on importing certificates into WebLogic Server, see https://docs.oracle.com/middleware/1221/opss/JISEC/cfgauthr.htm#JISEC2293.

2.8.2 Starting TLS

Alternatively the Start TLS LDAP extension can be used, where an unencrypted LDAP connection is established and encryption then negotiated. In this case, "relaxed checks" are performed on the LDAP server certificate, meaning that the LDAP server certificate does not need to be trusted. However, this means that EDQ cannot verify that it is connecting to the correct server, and requires that the network layer is trusted.

3 Authenticating Using Kerberos (GSSAPI)

This chapter describes how to authenticate EDQ using Kerberos.

EDQ supports integration with multiple authentication realms, which can use different LDAP servers. For example, a single EDQ server may support external authentication from both a Microsoft Active Directory realm and an Oracle Internet Directory realm, if required.

These global settings can be specified in the `security/login.properties` configuration file. Properties are configured using the syntax *`property_name = value`*, for example:

```
realms = realm1, realm2
```

Where noted, you can override the global settings at the realm level. Realm-level settings are more specific and always override global settings (see Section 2.6, "Configuring Individual Realm LDAP Settings").

Table 3–1 Kerberos Properties

Property	Description	Example Value	Mandatory?
`keytab`	The path to a Kerberos keytab file. If using SSO a single keytab must be defined at the global level. A single keytab can contain entries for several realms.	`/etc/krb5.keytab` If no path is specified, a default is chosen based on the operating.	No. Only necessary to enable SSO (where users do not need to log in to EDQ user applications) in environments where the EDQ server is not itself on the AD domain.
`spn`	Specifies the Kerberos Service Principal Name, used for SSO. May be overridden at realm level.	`HTTP/hostname@EXAMPLE.COM`	No. If not set, the default value is HOST/hostname.

4

Authenticating Using Oracle Access Manager

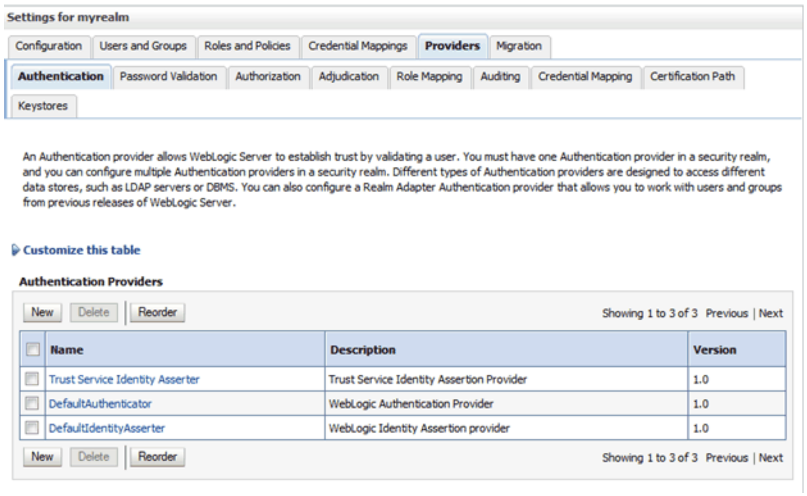
This chapter describes how to configure an Oracle WebLogic Server 12c server to authenticate users using Oracle Access Manager (OAM).

This chapter includes the following sections:

4.1 Overview of Configuring WebLogic to use OAM Authentication

The following configuration is performed within the Security Realms/Providers section of the WebLogic Server dashboard. For a newly created WebLogic domain, the Providers tab has the following contents:

Figure 4–1 Providers Tab in Oracle WebLogic Server



An Authentication provider allows WebLogic Server to establish trust by validating a user. You must have one Authentication provider in a security realm, and you can configure multiple Authentication providers in a security realm. Different types of Authentication providers are designed to access different data stores, such as LDAP servers or DBMS. You can also configure a Realm Adapter Authentication provider that allows you to work with users and groups from previous releases of WebLogic Server.

Authentication Providers

	Name	Description	Version
☐	Trust Service Identity Asserter	Trust Service Identity Assertion Provider	1.0
☐	DefaultAuthenticator	WebLogic Authentication Provider	1.0
☐	DefaultIdentityAsserter	WebLogic Identity Assertion provider	1.0

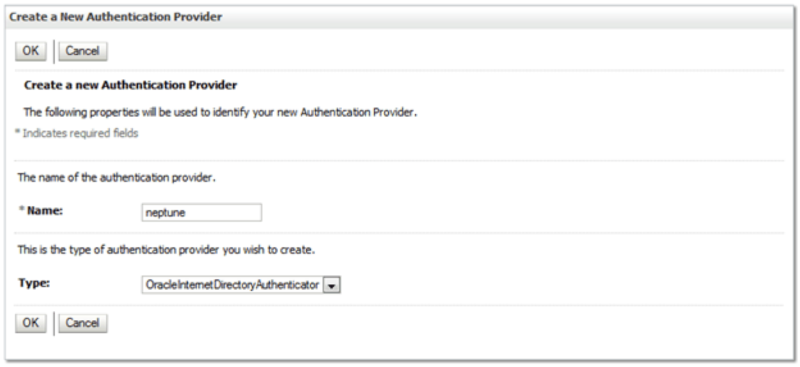
To configure Oracle Access Manager (OAM), you must set up two additional providers: LDAP and OAM.

4.2 Configuring an LDAP Provider

To configure an LDAP provider:

1. Click **New** to create a new authentication provider.
2. Enter a **Name** for the authentication provider.
3. Select `OracleInternetDirectoryAuthenticator` as the **Type**:

Figure 4–2 Creating a New Authentication Provider for LDAP



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:** neptune

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

Type: OracleInternetDirectoryAuthenticator

4. Click **OK**.
5. Select the name of the newly created provider from the list and set the **Control Flag** to `SUFFICIENT`.
6. Click **Save**.
7. Select the **Provider Specific** tab.
8. Set the following fields, leaving the remaining fields with default values:

Field	Value
Host	hostname of the OID server
Port	port number of the UID server
Principal	DN of an LDAP user with sufficient rights to search for users and groups
Credential	credential (that is, password) for the principal specified above
User Base DN	Base distinguished name for users
Group Base DN	Base distinguished name for groups

9. Set provider properties. For information, see "Setting Provider Priorities".
10. Ensure that the **Control Flags** are set to `SUFFICIENT` on the default and LDAP providers.
11. After configuring providers, adjust the order in which users are tested. This can be done after adding each provider or as the final step.
12. Restart the WebLogic admin server. Note that this must be done after all configuration changes.
13. In the WebLogic admin console, verify that you can see LDAP users and groups.

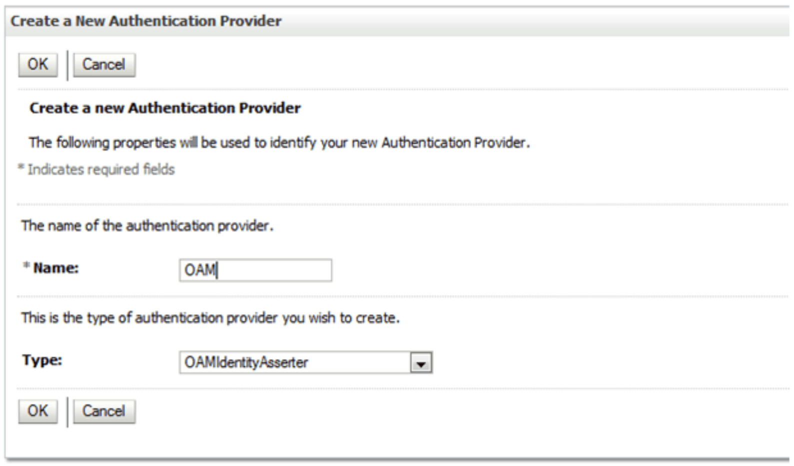
14. Ensure that there is a mapping to EDQ administrators group - either because your LDAP contains an *Administrators* group to which an EDQ user belongs, or by adding a new mapping to `login.properties`. See Section 2.2.1, "Configuring LDAP Group Mappings" for details.
15. Start EDQ server.
16. Verify you can login to EDQ using an LDAP user.
17. Configure any required additional external group mappings on the EDQ admin console.

4.3 Configuring an Oracle Access Manager Provider

To configure an Oracle Access Manager (OAM) provider:

1. On the providers list, click **New** and enter `OAM` as the name and `OAMIdentityAsserter` as the type:

Figure 4–3 Creating a New Authentication Provider for OAM



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.

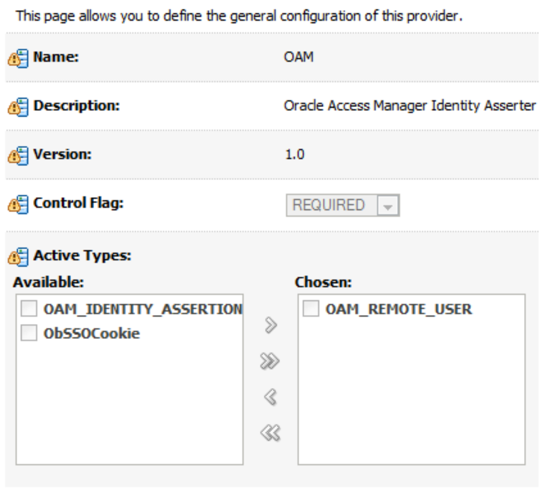
* **Name:** OAM

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

Type: OAMIdentityAsserter

2. Click **OK**.
3. Select OAM from the list and select the **Common** tab.
4. Set the control flag to `REQUIRED`:

Figure 4–4 Configuring the Provider



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: OAM_IDENTITY_ASSERTION, ObSSOCookie

Chosen: OAM_REMOTE_USER

5. Click **Save**.
6. Select the **Provider Specific** tab.
7. Set the following fields, leaving the remaining fields with default values:

Field	Value
Access Gate Name	The host name that you configured when you created the authentication provider. Use the plain host name without domain.
Primary Access Server	The primary Access Server, configured as `host:port`.

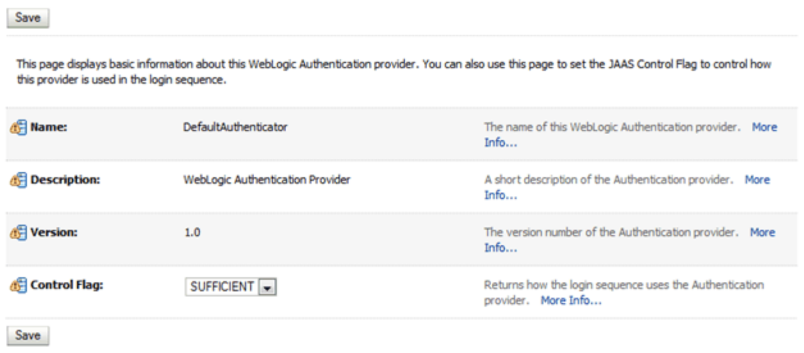
8. Move OAM to the top of the list of providers, just above LDAP providers.
9. Click **Save** to complete the provider definition.

4.4 Setting Provider Priorities

To set the provider priorities:

1. On the Providers list, select `DefaultAuthenticator` and change the **Control Flag** to `SUFFICIENT`:

Figure 4–5 Setting Provider Priorities



Save

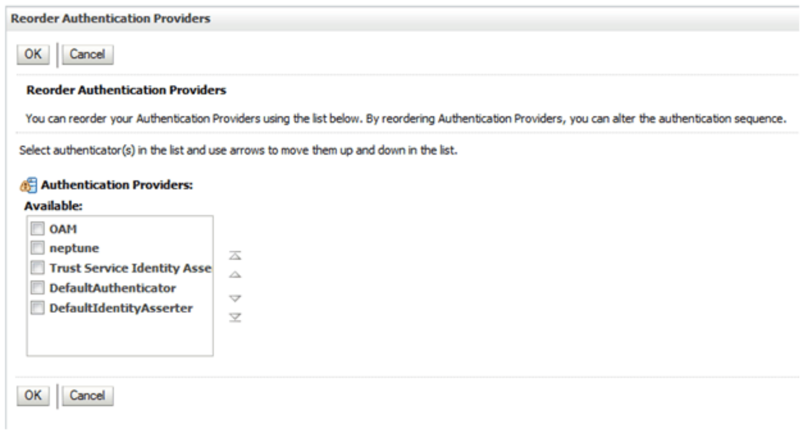
This page displays basic information about this WebLogic Authentication provider. You can also use this page to set the JAAS Control Flag to control how this provider is used in the login sequence.

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

2. On the Providers list, click **Reorder** and move OAM to the top with the `<provider_name>` second:

Figure 4–6 Reordering Authentication Providers



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:

- OAM
- neptune
- Trust Service Identity Asse
- DefaultAuthenticator
- DefaultIdentityAsserter

OK | Cancel

3. Restart the WebLogic server. Once the server is restarted, WebLogic is ready for OAM use. EDQ now gets all information from the LDAP provider, and the original user `weblogic` no longer works in EDQ. Instead, log in as user `edqadmin` with password `welcome1`.

4.5 OAM Configuration

Install Oracle HTTP Server (OHS) 11 or 12 and the WebGate extension. WebGate software is shipped with OHS 12. WebGate intercepts HTTP requests from users for web resources and forwards them to the Access Server for authentication and authorization.

If you use OHS 12, the WebGate software is bundled and you do not need a separate download. For more information, see Installing *WebGate in Oracle Access Manager Installation Guide*.

Configure the WebLogic plugin to forward /edq to WebLogic:

```
<Location /edq>
  SetHandler weblogic-handler
  WebLogicHost managedserverhost
  WebLogicPort managedserverport
```

```
</Location>
```

Finally install the WebGate artifacts, and restart OHS to complete the installation.

```
/edq/faces/** Protected Resource Policy
/edq/blueprints/*/jnlp Protected Resource Policy
/edq/** Public Resource Policy (or excluded)
```

5 Authorizing Users

Authorization of users is based on a model of users and permissions associated through groups. Users inherit permissions based on the groups to which they are associated. Permissions are grouped by subsystems in order to make the list easier to manage.

Groups are configured within EDQ, but may also be mapped from external groups (that is, in LDAP). Users and permissions are associated with groups, and the permissions a user has are based on the set of permissions assigned to any group to which they belong.

This chapter describes a sample EDQ plug-in script that performs custom, run-time filtering of user authorization groups based on the user's location as determined by IP address.

This chapter includes the following sections:

- Configuring Permissions
- Filtering User Authorization Groups
- Installing the Authorizations Plug-In
- Configuring the Authorizations Plug-In

5.1 Configuring Permissions

Permissions are granted to groups using the EDQ administration web pages. To administer Groups, use **Launchpad > Administration > Groups**. From this page you can Add, Edit, and Delete groups (Administrators, Data Analysts, Data Stewards, Executives, Match Reviewers, and Review Managers).

For more information, see the EDQ Administration web pages, and `http://www.oracle.com/webfolder/technetwork/data-quality/edqhelp/Content/administration/user_groups.htm`

5.2 Filtering User Authorization Groups

The sample *Authorizations Filter* plug-in is designed to address legislative requirements that some data must not be taken out of a particular country. In particular, a user logging on from a different country must not be able to view or access the restricted data, even if they would normally have sufficient privileges to do so in their home area.

Project access in EDQ is normally controlled by assigning users and projects to groups. Both users and projects may have multiple groups. A user has access to a project if they are a member of at least one of the project's groups as assigned to the project in Director. The plug-in described here operates at runtime to filter the groups assigned to a user, based on the user's IP address at the time they log on.

For this plug-in to work as intended, projects must be assigned to groups based upon their data access restrictions. That is, all projects that may only be accessed from within country A should be assigned to one group, projects that may only be accessed from country B should be assigned to a second group. Any projects with no country-based access restrictions can be made available to all groups. Users can be granted access to each group as usual, and the plug-in script will provide per-session, IP address-based filtering of the groups available to a user.

This script can be configured to filter user authorizations based on either IPv4 or IPv6 addresses. It is not possible to filter on both IPv4 and IPv6 addresses at the same time.

Note: Users with the *Add Project* permission (usually administrators and power users) always have access to all projects. This bypasses the group system and is unaffected by this plug-in. By default, the Administrators and Project Owners user groups have this permission. It is possible to circumvent this issue by removing the Add Project permission from all users once the system has been fully configured. If it is necessary to create further projects in the future, a user can be granted the Add Project permission as a temporary measure.

The Authorizations plug-in is a JavaScript plug-in script and configuration data that can be provided either as an Extensible Markup Language (XML) file or as a comma-separated list. You can activate the plug-in and select which type of configuration file is to be used by editing the `security.properties` file in the EDQ server configuration directory.

5.3 Installing the Authorizations Plug-In

The plug-in is installed by adding files to, and editing files in, the `oedq.local.home` configuration directory. The location of this directory is determined during installation. On a typical WebLogic platform, the path is usually as follows:

```
/middleware/Oracle_Home/user_
projects/domains/<yourdomain>/config/fmwconfig/edq/oedq.local.home
```

To install the plug-in:

1. Create a new script file, `userfilter.js`, in the `oedq.local.home` configuration directory.
2. Place the JavaScript code from Section 5.3.1, "Filter Script" into `userfilter.js`.
3. Create a new file to hold the IP address filtering rules in your configuration directory. Filter rules can be expressed either in comma-separated values (CSV) format, in which case your rules file should be named `ipranges.csv`, or in XML format, in which case your rules file should be named `ipranges.xml`. You will add data to these files in the configuration step, described in Section 5.4, "Configuring the Authorizations Plug-In."
4. Edit the `security.properties` file in the `oedq.local.home` configuration directory, or create it if it does not already exist, and add the line:

   ```
   user.filter.scriptfile = userfilter.js
   ```

5. Add one of the two following lines to `security.properties`. If you want to use the XML version of the configuration file, add:

   ```
   user.filter.xfile = ipranges.xml
   ```

   If you want to use the CSV version of the configuration file, add:

   ```
   user.filter.cfile = ipranges.csv
   ```

6. Restart your application server so that the new settings from the properties files are retrieved and set.

5.3.1 Filter Script

The filter script is as follows:

```
// User filter test which reads CSV or XML
// ======================================
addLibrary("logging");
addLibrary("io");

// Main filtering function for the script
// =====================================
function filter(user, ip)
{
logger.log(Level.INFO,
"Filtering user {0} from IP {1}",user.getUserName(), ip);
var xprop = props["user.filter.xfile"];
var cprop = props["user.filter.cfile"];
var cfile = cprop == null ? null : findFile(cprop);
var xfile = xprop == null ? null : findFile(xprop);

if (cfile == null && xfile == null)
{
logger.log(Level.INFO, "No IP range files available");
return true;
}

// Process CSV file
// ================
if (cfile != null)
{
// CSV file has group name and IP start/end on each line; a group
// matches if the IP is in any of the ranges
var hash= new Object();
var rdr= IO.createCSVReader(cfile);
var arr;

while ((arr = rdr.read()) != null)
{
// Ignore lines with too few fields
if (arr.length >= 3)
{
// Store group in hash once only
var grp= trim(arr[0]);
var start = trim(arr[1]);
var end= trim(arr[2]);
if (hash[grp] == null)
{
hash[grp] = false;
}
```

```
if (ipInRange(ip, start, end))
{
hash[grp] = true;
}
}
}

// Now reject any groups which did not match
for (var g in hash)
{
if (!hash[g])
{
user.removeGroupByName(g);
}
}
}

// Process XML file
// XML schema is:
//
//<ipranges>
//<group name="name">
//<iprange start="x" end="y"/>
//...
//</group>
//...
// </ipranges>
//
// The purifyXML call here removes the <?xml ..?> header
// which E4X does not like
// ================
if (xfile != null) {
var xml= new
XML(XMLTransformer.purifyXML(IO.load(xfile)));
var grps= xml.group;
var ng= grps.length();
for (var i = 0; i < ng; i++)
{
var grp = grps[i];
var ranges = grp.iprange;
var ok= false;

for (var j = 0; j < ranges.length(); j++)
{
var range = ranges[j];
if (ipInRange(ip, range.@start, range.@end))
{
ok = true;
break;
}
}

if (!ok)
{
user.removeGroupByName(grp.@name);
}
}
}
return true;
}
```

```
// Trim function
// =============
function trim(a)
{
return a.replace(/ /g, '');
}
```

5.4 Configuring the Authorizations Plug-In

This plug-in can accept configuration data either in XML or CSV format. In either case, the configuration data maps a group to one or more IP ranges that correspond to permitted access locations. If a user logs on to the system from an IP address outside the permitted ranges, the associated group will be blocked from the user for the duration of the session.

The data in these files can be edited to modify the location-based filtering of project access. When editing the data (for either file format), you should consider the following points:

- If a group is not mentioned in the configuration file, no location-based filtering will be applied to it.
- The IP address ranges in the files are those that are allowed to access projects in the associated groups. To allow access to a group from more locations, add an IP range or widen the scope of an existing IP range. To disallow access to a group from some locations, remove the corresponding IP address range, or narrow the scope of the relevant range.

5.4.1 XML File Format

The XML configuration file has the following structure:

```
<ipranges>
<group name="name">
<iprange start="x" end="y"/>
...
</group>
...
</ipranges>
```

The configuration data consists of one or more `<group>` elements, where each group is identified by name. Each group specifies one or more IP address ranges that are permitted access to the projects in that group. An address range is specified as an `<iprange>` element, with a start and an end attribute defining the limits of the address range. In this way, multiple valid IP address ranges can be configured for each group.

All the group/IP range mapping data in the file must be contained within the `<ipranges>` tag.

For example, suppose an XML configuration file contains the following data:

```
<ipranges>
<group name="group1">
<iprange start="1.1.1.0" end="1.1.1.20" />
<iprange start="10.1.0.0" end="10.1.0.255" />
</group>
<group name="group2">
<iprange start="10.8.1.0" end="10.8.1.125" />
</group>
```

```
</ipranges>
```

This configuration data means that:

- Projects that belong to `group1` can only be accessed by logging on from an IP address that is either in the range 1.1.1.0 to 1.1.1.20 or in the range 10.1.0.0 to 10.1.0.225.
- Projects that belong to `group2` can only be accessed by logging on from an IP address in the range 10.8.1.0 to 10.8.1.125.

If a group is not specified in the configuration file, access to projects in that group will not be controlled by IP address. For example, user access to projects belonging to a third group named group3 will be unaffected by this script and configuration data.

5.4.2 CSV File Format

The CSV configuration file format specifies one group and address range mapping per line. Each line consists of three comma-separated fields, as follows:

```
group name, start of address range, end of address range
```

For example, suppose a CSV configuration file contains the following data:

```
group1, 1.1.1.0, 1.1.1.20

group2, 10.8.1.0, 10.8.1.125

group1, 10.1.0.0, 10.1.0.255
```

This configuration file is functionally identical to the sample XML file in Section 5.4.1, "XML File Format." That is:

- Projects that belong to `group1` can only be accessed by logging on from an IP address that is *either* in the range 1.1.1.0 to 1.1.1.20 *or* in the range 10.1.0.0 to 10.1.0.225.
- Projects that belong to `group2` can only be accessed by logging on from an IP address in the range 10.8.1.0 to 10.8.1.125.

It is not necessary for all the valid IP address ranges for a group to be specified on adjacent lines. Again, if a group is not present within the file, no IP address filtering will be performed for that group.

6 Using Encryption

This chapter provides encrypting to protect your data. It includes the following sections:

- Understanding Encryption
- Configuring SSL with Tomcat
- Configuring SSL with WebLogic
- Encrypting LDAP Connections
- Encrypting Database Connections

6.1 Understanding Encryption

It is important to encrypt traffic in order to ensure both that it is neither read nor modified while traversing the network. If read, unencrypted traffic could expose data records being processed, or the results of that processing. If modified, instructions from client to server could be changed, allowing requests to be executed using the permissions of the user to whom the connection belongs.

There are four key EDQ areas that must be encrypted:

- Connections between web browsers and EDQ user applications (such as Director), to the EDQ server
- Connections from EDQ to authentication servers (such as LDAP)
- Connections from EDQ to databases
- Connections from remote systems to the FTP/SFTP servers included with EDQ

6.2 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`

6.3 Configuring SSL with WebLogic

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

https://docs.oracle.com/middleware/1221/wls/SECMG/ssl_overview.htm#SECMG386

6.4 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). These options and their configuration are described in more detail in Section 2.6, "Configuring Individual Realm LDAP Settings".

6.5 Encrypting Database Connections

JDBC URL syntax for connections over TLS is dependent on the database driver being used. For Oracle Database, this is achieved by adding *`PROTOCOL=tcps`* to the connection specification, for example:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=servername
     )(PORT=2484))(CONNECT_DATA=(SERVICE_NAME=servicename)))
```

For details on JDBC URL syntax see https://docs.oracle.com/database/121/JJDBC/

7 Auditing

Auditing user activity provides accountability; it tracks what has been done, when, and by whom. Auditing is accomplished either through Oracle Fusion Middleware Framework, or through This chapter describes auditing in EDQ. It includes the following sections:

- Using Oracle Fusion Middleware Audit Framework
- Using Audit Logs on Disk

7.1 Using Oracle Fusion Middleware Audit Framework

Where EDQ is installed with an Oracle WebLogic Server domain, you can configure it to log audit events via the Oracle Fusion Middleware Audit Framework. For detailed information on the framework see "Introduction to Oracle Fusion Middleware Audit Service" in Oracle Fusion Middleware Securing Applications with Oracle Platform Security Services.

To enable audit event logging:

1. Open the Enterprise Manager 12c Fusion Middleware Control application.

   The path to this application is:

   ```
   http://[servername]:[weblogic server admin port, e.g. 7001]/em
   ```
2. Navigate to the EDQ domain in the Target Navigation Tree on the left of the window.
3. Right-click the domain and select **Security > Audit Policy**.
4. Select "EDQ" in the **Audit Component Name** field.
5. Select "Custom" in the **Audit Level** field.
6. Select the categories to log, and the events within those categories.
7. Click **Apply**, or **Revert** to abandon the changes.

7.1.1 Configuring the EDQ Events in Fusion Middleware Framework

Set the directory property in the `audit.properties` file to be any other directory (that exists), relative to your local *config home*.

For example, add the line:

```
directory = myAudits
```

to your new file, where *myAudits* is a folder that exists at the same level as your new `audit.properties` file.

The EDQ event categories and types are as follows:

Event Category	Event Types
Asset Transfer	Import Package
Case Management	Bulk Delete, Bulk Update, Bulk Assignment, Display Data edited, Export, Edit, Assignment updated, State changed, Comment added, Comment deleted, Comment edited, Attachment added, Attachment deleted
Case Management Admin	Case Source Added, Case Source Imported, Case Source Deleted, Permission Added, Permission Modified, Permission Deleted, Workflow Added, Workflow Imported, Workflow Deleted, Parameter Added, Parameter Modified, Parameter Deleted, Reception Action Added, Reception Action Modified, Reception Action Deleted, Reception Transition Added, Reception Transition Modified, Reception Transition Deleted, State Transition Added, State Transition Modified, State Transition Deleted, Workflow State Added, Workflow State Modified, Workflow State Deleted
Group Permission Management	Join group, Leave group, Leave all groups, Create group, Delete group, Change permissions.
Launchpad Management	Extension Add, Extension Delete, Front Page Update
Object Management	Create, Update, Delete.
User Management	Login, Logout, Password Change, Password Expire, User Blocked, User Blocked Temporarily, User Unblocked, User Created, User Updated, User Deleted, Security Configuration Updated.

The attributes that can be logged by events and the corresponding Custom Attribute Slot are listed in the following table. Please note that this is not a complete list.

Event Attribute	Description	Custom Attribute Slot
Affected user	The name of the user for the logged event.	IAU_STRING_001
Login application	The name of the application that has been logged into.	IAU_STRING_002
Project Name	The name of the project containing the affected object. This attribute is left blank for system-level objects.	IAU_STRING_003
Item Type	The type of object created, modified or deleted.	IAU_STRING_004
Item Name	The name of the object created, modified or deleted.	IAU_STRING_005
Affected user	The name of the user affected by changes made by an administrator.	IAU_STRING_006
Affected group	The name of the group affected by changes made by an administrator.	IAU_STRING_007
Added Permissions	List of permissions added to a group.	IAU_LONGSTRING_001
Removed Permissions	List of permissions removed from a group.	IAU_LONGSTRING_002

Custom attributes are stored in the `iau_custom` table. For more information, see "Audit Reporting with the Dynamic Metadata Model" in *Oracle Fusion Middleware Securing Applications with Oracle Platform Security Services*. The generic attributes for the event are stored in the `iau_common` table. Both of these are in the IAU schema (`[RCUPREFIX]_IAU`).

Once enabled, EDQ audits events by calling the central Oracle Fusion Middleware Audit Framework APIs. The audit events can then be stored either as files or in a database for compliance reporting purposes. For more information on how to store and report on the results of auditing, see *Oracle Fusion Middleware Securing Applications with Oracle Platform Security Services*.

7.2 Using Audit Logs on Disk

Where EDQ is installed in Apache Tomcat, or if you prefer not to use Oracle Fusion Middleware Audit Framework, audit logs can instead be written to files on disk.

To enable this, create a file named `audit.properties` in the local configuration directory and add the line:

```
enabled = true
```

You can then either create a directory named audit in your local configuration directory, or specify a path to an existing directory using the directory property in `audit.properties`. This path is specified relative to the local configuration directory.

7.2.1 Configuring the EDQ Events in Audit Logs on Disk

For more fine-grained control over the specific categories and events that are audited, you can turn certain categories off. To do so add lines of the following structure to `audit.properties`:

```
category.<category name>. enabled = false
```

You can then turn individual events for that category back on, or turn them off if the category has been left enabled, as follows:

```
category.<category name>.<event name>.enabled = <true/false>
```

Upon audit events being generated, they will be placed in per-category files within the configured audit directory. These files contain entries as comma-separated values with the first line containing column headers.

A Tips and Troubleshooting

This appendix contains usage tips and troubleshooting information about authentication and encryption challenges that may arise. It contains the following sections:

- Optimizing Authentication
- Optimizing Encryption
- Optimizing Auditing

A.1 Optimizing Authentication

This section contains solutions for a number of common problems encountered with authentication in EDQ. Many LDAP configuration problems are the result of typographical errors or unexpected characters in the configuration files; files should be carefully checked in case of problems.

A.1.1 Correcting Excessive LDAP Connections

EDQ automatically regularly updates its internal cache of users from the LDAP directory, which can lead to a large number of connections. If only a small subset of users require access to EDQ, this process can be streamlined by associating EDQ users with an LDAP group and setting the `ldap.prof.defaultusergroup` option for the realm. EDQ will then constrain the subset of users retrieved to those in that group. See Section 2.6, "Configuring Individual Realm LDAP Settings" for details of the `ldap.prof.defaultusergroup` option.

A.1.2 Configuring LDAP Server Failover

Configuring LDAP server failover with EDQ integrated LDAP support can be done by specifying multiple servers for a realm. For OPSS, see the WebLogic documentation for details of setting up server failover:

`http://docs.oracle.com/middleware/1221/wls/SECMG/ldap_atn.htm#SECMG180`

A.1.3 Reconciling Multiple User Display Names

In rare cases, the default display names of users may include duplicates. In this case the display name can be configured to be generated from different user properties. The *`userdisplayname`* property (in LDAP and OPSS profile configurations) is an expression which can build a value from attributes read from LDAP. Here are two example solutions, showing how users appear in the mail notification dialogue for jobs:

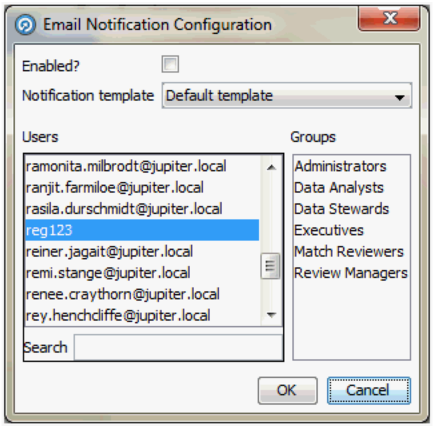
A.1.3.1 Use Email Address, or Login ID if not Set

Override the LDAP profile's `userdisplayname` option by putting the following in to `login.properties`:

```
opss.prof.userdisplayname = mail == '' ? loginid : mail
```

The example below shows the email notification configuration (of a job) with this setting applied:

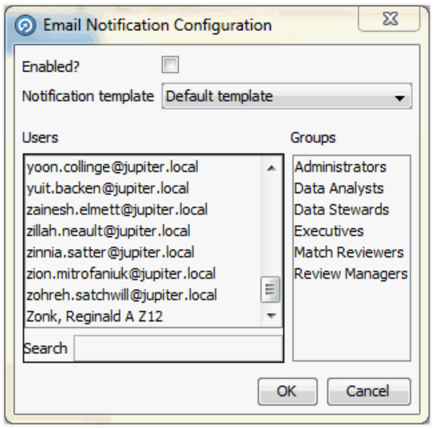
Figure A–1 Email Notification Configuration dialog


Email Notification Configuration
Enabled?
Notification template
Default template
Users
Groups
ramonita.milbrodt@jupiter.local
ranjit.farmiloe@jupiter.local
rasila.durschmidt@jupiter.local
reg123
reiner.jagait@jupiter.local
remi.stange@jupiter.local
renee.craythorn@jupiter.local
rey.henchcliffe@jupiter.local
Administrators
Data Analysts
Data Stewards
Executives
Match Reviewers
Review Managers
Search
OK
Cancel

Here *reg123* is the single user who does not have an email address, so their login ID is displayed instead.

Note that if integration is through OPSS, AD attributes cannot be used directly; instead you would use one of the names listed with the IGF UserProfile class. The loginid OPSS attributes maps to the *User Name Attribute* set in WLS; if you switch this from *sAMAccountName* (the Windows login ID) to *cn* the list changes to:

Figure A–2 Email Notification Configuration dialog used to reconcile display names


Email Notification Configuration
Enabled?
Notification template
Default template
Users
Groups
yoon.collinge@jupiter.local
yuit.backen@jupiter.local
zainesh.elmett@jupiter.local
zillah.neault@jupiter.local
zinnia.satter@jupiter.local
zion.mitrofaniuk@jupiter.local
zohreh.satchwill@jupiter.local
Zonk, Reginald A Z12
Administrators
Data Analysts
Data Stewards
Executives
Match Reviewers
Review Managers
Search
OK
Cancel

Here the final name is the cn value for the user with *sAMAccountName* = *reg123*.

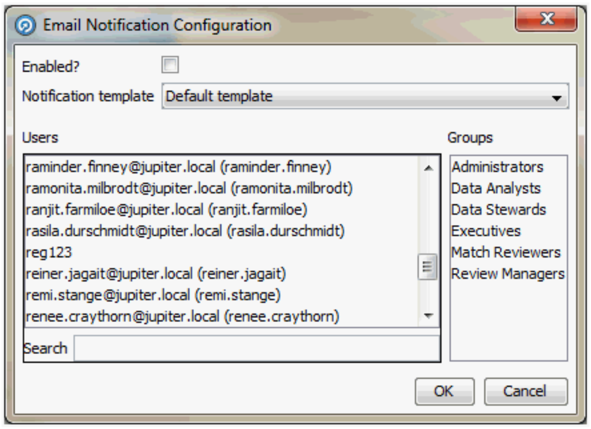
A.1.3.2 Use Email Address and Login ID

If there are any duplicate email addresses in AD, the exception will still occur, so in this case you can combine the email address with the login ID to ensure the string is unique.

```
opss.prof.userdisplayname = mail == '' ? loginid : mail || ' (' || loginid
|| ')'
```

The result is:

Figure A–3 Combining email address with login to create a unique string



Email Notification Configuration

Enabled? ☐

Notification template: Default template

Users
raminder.finney@jupiter.local (raminder.finney)
ramonita.milbrodt@jupiter.local (ramonita.milbrodt)
ranjit.farmiloe@jupiter.local (ranjit.farmiloe)
rasila.durschmidt@jupiter.local (rasila.durschmidt)
reg123
reiner.jagait@jupiter.local (reiner.jagait)
remi.stange@jupiter.local (remi.stange)
renee.craythorn@jupiter.local (renee.craythorn)

Groups
Administrators
Data Analysts
Data Stewards
Executives
Match Reviewers
Review Managers

A.1.4 Resolving Error Messages

This section contains common error messages and their solutions.

A.1.4.1 Error processing GSS server login

GSS errors when configuring EDQ to authenticate against an LDAP server are likely to indicate that the method for authenticating to the LDAP server (`ldap.auth`) is not set, and *gss* has been used as a default. Typically this would be set to *simple*. See Section 2.5, "Using LDAP Server Profile" for details on the `ldap.auth` setting.

A.1.4.2 Invalid argument (400) - Cannot find key of appropriate type to decrypt...

This error can occur when integrating EDQ with Kerberos, and indicates that the keytab file on the EDQ server does not support the required encryption type. To correct this, the keytab file must be updated to support the missing encryption type. See the documentation for the ktpass tool for further details:
`https://technet.microsoft.com/en-us/library/cc753771.aspx`

(for additional background information, see also:
`https://community.oracle.com/thread/1527533`

A.1.4.3 Cannot insert null in DN_EXTUSERMAP.OBJECT_KEY

This error can occur where an external domain is configured to use a field which can be null as the unique ID for a user. For example, with AD integration, normally the `objectguid` attribute in AD is used to provide a unique identifier for users, however if it is changed to a different attribute, this error could arise if the named attribute is missing or empty on a user record.

A.2 Optimizing Encryption

This section contains solutions for a number of common problems encountered with encryption in EDQ.

A.2.1 Untrusted Certificate Errors

Untrusted certificate errors indicate that the client does not trust the X.509 certificate presented by the server for a TLS connection. This can be for various reasons, although the most common is that the client cannot verify the chain of certificates used to verify the certificate that the server presents. Other reasons might be that the certificate is out of date, or is not yet valid (certificates have date ranges for which they are valid), or that the certificate was issued for a different server than the one being connected to.

Certificates are trusted based on either the client knowing the certificate already (that is, it's pre-installed, often shipped with the operating system), or it being signed by a certificate the client trusts. Servers present a certificate that identifies themselves, along with a signature from a certificate the client trusts (trusted root). Optionally server certificates may have intermediary certificates which are signed by the trusted root, and sign the server certificate, forming a chain of certificates to be validated by the client. These intermediary certificates and signatures are presented to the client along with the server certificate.

Common problems here include self-signed server certificates, server certificates which are not signed by a certificate authority the client trusts, and missing intermediate certificates or signatures.

A.2.1.1 Connecting to EDQ (HTTPS)

When connecting to EDQ the server certificate is managed by WebLogic Server or Apache Tomcat, and as such any problems with untrusted certificates must be resolved by following the instructions for the relevant server.

A.2.1.2 Connecting to LDAP (EDQ Integrated)

EDQ verifies server certificates when connecting to LDAP over TLS, and where servers use certificates that are not automatically recognized, certificates must either be installed into EDQ or verification-disabled. Certificate verification can be disabled by using Start TLS rather than TLS (this is an EDQ-specific behavior difference). See Section 2.7, "Validating Credentials When Single Sign-On Is Not Used" for details.

A better solution is to extract the certificate from the LDAP server and install it in the EDQ server so that it is recognized as being trusted.See Section 2.7, "Validating Credentials When Single Sign-On Is Not Used" for details.

A.2.1.3 Connecting to LDAP (WebLogic with OPSS)

For instructions on managing the key/certificate stores used by WebLogic, see the Fusion Middleware Administrator's Guide. For instructions on importing certificates into WebLogic Server, see
`https://docs.oracle.com/middleware/1221/opss/JISEC/cfgauthr.htm#JISEC2293`

A.3 Optimizing Auditing

This section contains information about auditing in EDQ.

A.3.1 “BAD PASSWORD” Events Not Captured for External Domain

Where authentication is handled by an external system such as OID or AD, EDQ’s audit trails do not include events such as bad passwords, as these are managed by the external authentication system. Auditing for such events must be configured in the external system.

All events will be captured as normal for any internal domain.