Oracle® Fusion Middleware

Securing Oracle Enterprise Data Quality

12*c* (12.1.3)

E51298-01

May 2014

Oracle Fusion Middleware Securing Oracle Enterprise Data Quality, 12*c* (12.1.3)

E51298-01

Contents

Preface

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

Audience

The intended audience of this guide are experienced Java developers, administrators, 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 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 Enterprise Data Quality 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

`http://http://docs.oracle.com`

Online Help

Online help is provided for all Oracle Enterprise Data Quality 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 Enterprise Data Quality Security Architecture

This chapter describes how security is incorporated in many aspects of the Oracle Enterprise Data Quality (EDQ) architecture.

This chapter includes the following sections:

1.1 Client-Server Communication

The security of communication between the web application server and the client applications is determined by the configuration of the Java Application Server hosting EDQ. The Java Application Server can be configured to use either HTTP or HTTPS.

1.2 Authentication

EDQ authenticates user passwords against values held in the database repository or in a Lightweight Directory Access Protocol (LDAP)-enabled user management server. The passwords are held in a hashed form that the application cannot reverse. This configuration is used by:

- the client user applications
- the EDQ web pages

The client user applications authenticate users using a proprietary protocol over HTTP or HTTPS. Passwords are encrypted before being sent to the server.

The web pages are secured using forms-based authentication that communicates with the server over HTTP or HTTPS.

Mandatory password strength enforcement can also be configured, encompassing the following criteria:

- Minimum length.
- Minimum number of non-alphabetic characters.
- Minimum number of numeric characters.
- Prevention of recent password re-use.
- Prevention of using the user name in the password.

Account security encompassing the following criteria can also be configured:

- Password expiry.
- Application behavior following failed login attempts.

1.3 Storing Security Information

EDQ stores security information in a number of places, depending on the nature of the information

Connection details for databases that EDQ connects to in order to perform snapshots and dynamic value lookups are stored in the configuration schema in the repository. This includes user names, passwords, hosts names, and port numbers for the database connections.

Passwords are stored in an encrypted form in the configuration schema or in a security store on WebLogic platforms, and then they are decrypted by EDQ when it needs to log into a database. A decrypted password is not shown to EDQ users or administrators.

The encryption/decryption key for passwords is generated randomly for each installation of EDQ. On WebLogic platforms, the key is retained in the security store. On Tomcat platforms, it is stored in a file in the EDQ configuration directory. This random key generation on a per-installation basis ensures that encrypted passwords cannot be copied meaningfully between systems.

In installations where EDQ uses Java Naming and Directory Interface (JNDI) connections to connect to the repository database, the authentication details to the database are stored in the application server. EDQ uses the credentials by referencing the JNDI names in a file (`director.properties`) in the EDQ base configuration directory (`oedq_home`).

1.4 Data Segmentation

When EDQ is being used across multiple business lines, or when several businesses are using the same system, it is important to be able to segment user access to data. EDQ supports this segmentation by allowing users and projects to be allocated to groups. Users can only access a project if they are members of the same group as the project. The Director user application presents accessible projects to a given user, and all other projects are invisible. The contents of any project (including reference data and web services) are unavailable to unauthorized users.

Note: Since administrative users must be able to manage all the projects in the system, any user with permission to create projects can see all projects in the system regardless of project settings.

2 Integrating with LDAP

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 Overview of LDAP Support

EDQ can be integrated with LDAP servers in two ways:

1. Using Oracle Platform Security Services (OPSS), configured on Oracle WebLogic Server
2. Directly, using connection settings configured in EDQ configuration files

Once EDQ is integrated with LDAP, external user management works in a consistent way. You map groups of users that exist on the LDAP system ("External Groups") to EDQ groups by using the Administration pages on the EDQ Launchpad. With these mappings, you grant users in the external groups permissions to the EDQ server.

Currently, EDQ is certified for integration with the following LDAP server technologies:

- 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 integrations with Active Directory, EDQ supports Single Sign-On (SSO). Authorized users of the AD domain can access EDQ without the need to log in to the EDQ client applications.

2.2 Integrating LDAP Using OPSS on a WebLogic Server

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

The integration is controlled by a property in the `login.properties` file. This file is installed in the `security` directory of the base configuration directory (`oedq_home/security`).

A setting in `login.properties` specifies the default mapping of the LDAP administrators group to the EDQ administrators group. The default mapping ensures that a WebLogic Server Administrator can access the Administration application on the EDQ Launchpad to map other external groups on LDAP to the appropriate internal groups.

Provided the WebLogic Server identity store or a configured LDAP server has a group with the name of `Administrators`, there is no need to adjust any of the settings in `login.properties`. If the LDAP server does not contain a group with the name of `Administrators`, you can do either of the following:

- Create a group named `Administrators` on the LDAP server, and then restart the server that manages EDQ in WebLogic Server.
- Modify the default administrators group mapping. See "To adjust the default Administrators group mapping" for instructions.

To adjust the default Administrators group mapping

This procedure creates a local `login.properties` file to override the base `login.properties` file, and then adjusts the default `Administrators` 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_home/security`) to `oedq_local_home/security`.
3. Look for the following default mapping:

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

   Note: If the name of the EDQ administrators group was changed to something other than `Administrators`, the entry will appear similar to the following:

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

   ---

4. Change the default mapping to map the name of the external administrators group to the name of the EDQ administrators group.

   ```
   opss.xgmap = name_of_external_group -> name_of_EDQ_admin_group
   ```

Note: The property accepts a delimited list of mappings. For example, the following syntax is valid for mapping the "DQ Admins" and "DQAdministrators" external groups to the EDQ "Administrators" group:

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

---

5. Save the file.
6. Restart the application server.

2.3 Integrating LDAP Directly on Apache Tomcat

On an Apache Tomcat server, EDQ provides direct integration with LDAP servers, but it is not enabled by default. To enable the integration, you use a template to create and configure a `login.properties` file in the local configuration directory. The settings in this file override those in the `login.properties` file in the base configuration directory.

To configure direct integration with an LDAP server

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 different 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
4. Save the file as `login.properties` in the same directory.
5. Restart the application server.

Note: Properties of these built-in profiles can be overridden where required by using options specified in the following format: `ldap.`*`profile.propertyname`*.

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

2.4 Configuring Global LDAP Settings

EDQ supports integration with multiple realms. Each realm can use different LDAP server technologies. 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. This section describes the `login.properties` properties that are normally set globally (for all realms).

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 2.5, "Configuring Realm 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.
`keytab`	The path to a Kerberos keytab file. A single keytab must be defined at the global level. A single keytab can contain entries for several realms.	`/etc/krb5.keytab` If a specific path is not specified, the Operating System default path is used.	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.
`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`.
`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`*
`x509`	Enables the use of x509 certificates (client SSL certificates) for client authentication in EDQ. 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`.

Table 2–1 (Cont.) Global LDAP Settings

Property	Description	Example Value	Mandatory?
`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 Configuring Realm 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 `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 2–2 Realm-Level 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.	`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
`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 2.6, "Validating Credentials When Single Sign-On Is Not Used" for further information.	`bind`	No.

Table 2–2 (Cont.) Realm-Level LDAP Settings

Property	Description	Example Value	Mandatory?
`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 Kerberos/GSSAPI for SSO. 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`	Base of 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`.	`tls`	No `tls` should be used where possible.
`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=example3, 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`. No, otherwise
`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. For further information about this parameter, see Section 2.4, "Configuring Global LDAP Settings."	`true`	No. If not set, the default value is the one specified at the Global level.

Table 2–2 (Cont.) Realm-Level LDAP Settings

Property	Description	Example Value	Mandatory?
`ldap.prof.defaultusergroup`	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 Windows, which may be too large and cause display and memory issues.
`ldap.prof.groupsearchfilter`	Additional filter for groups; an LDAP search filter.	`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.

2.6 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, set the following additional properties:

- `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:` *`attr`*, which searches for users by a specific attribute in their user record.
- `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.

2.7 LDAP Security

By default, data transmitted over LDAP is unencrypted. Either Secure Sockets Layer (SSL) or Transport Layer Security (TLS) can be used to provide encryption.

LDAP over SSL (LDAPS) employs a properly formatted certificate. If used, the server certificate must define a valid host name and must be trusted by the Java Runtime Environment (JRE) that is running EDQ.

TLS uses the `startTLS` LDAP extension. In TLS implementations, "relaxed checks" are performed on the LDAP server certificate, meaning that the LDAP server does not need to be trusted by the JRE that is running EDQ.

2.8 Example LDAP Configurations

This section provides the following example configurations for the following LDAP technologies:

- Section 2.8.1, "Example of Oracle Internet Directory LDAP Configuration"
- Section 2.8.2, "Example of Microsoft Active Directory LDAP Configuration"
- Section 2.8.3, "Example of Open LDAP Configuration"
- Section 2.8.4, "Example of Novell eDirectory LDAP Configuration"

Note: The settings shown in these examples are included in the `login.properties.template` file with an additional `#` character at the beginning of each line. Remove this character to uncomment each property to make it active.

2.8.1 Example of Oracle Internet Directory LDAP Configuration

This section describes typical `login.properties` settings required to integrate EDQ with an Oracle Internet Directory LDAP server.

In this example, the LDAP user and password are transmitted in the clear (unencrypted) over the network. Oracle recommends the use of SSL (`example3.ldap.security = ssl`) or TLS (`example3.ldap.security.tls`) to encrypt LDAP traffic.

```
# Oracle Internet Directory Example
realms = example3
# Map the realm to a domain name
example3.realm                      = EXAMPLE3.COM
# Disable GSS
example3.gss                        = false
# Authorize user by using LDAP bind to server
example3.auth                       = ldap
 # Use distinguished name for authentication
example3.auth.binddn                = search: dn
# The LDAP server
example3.ldap.server                = server3
example3.ldap.auth                  = simple
# The OID user credentials to be used by EDQ (user.name@example3.com)
example3.ldap.user                  = cn=intuser,cn=users,dc=example3,dc=com
example3.ldap.pw                    = password
# The base distinguished name is example3.com
example3.ldap.basedn                = dc=example3,dc=com
# Use InetOrgPerson Style LDAP schema
example3.ldap.profile               = inetorgoidldap
# The name of the user group containing all EDQ users
example3.ldap.prof.defaultusergroup = group3
example3.ldap.security = ssl
```

2.8.2 Example of Microsoft Active Directory LDAP Configuration

This section describes typical `login.properties` settings required to integrate EDQ with a Microsoft Active Directory LDAP server.

In an AD environment, EDQ can be configured to permit SSO by users in the same domain. This example does not configure SSO, because it does not include the `clientcreds=true` property and setting.

```
# Active Directory Example
realms = example1
# Map the realm to a domain name
example1.realm                      = EXAMPLE1.COM
# Authorize user by using LDAP bind to server;
# for Active Directory it must be ldap
example1.auth                       = ldap
# The authentication to use, in this case digest-md5 using
# the plain account name (example1.auth.binddn).
example1.auth.bindmethod            = digest-md5
example1.auth.binddn                = search: sAMAccountName
# Use Transport Layer Security. Requires a X.509 certificate to be
# installed on the domain controller.
example1.ldap.security              = tls
# The LDAP Schema to use.  For Active Directory adsldap is the
# standard schema to use
example1.ldap.profile               = adsldap
# The name of the user group containing all EDQ users.
example1.ldap.prof.defaultusergroup = group1
```

2.8.3 Example of Open LDAP Configuration

This section describes typical `login.properties` settings required to integrate EDQ with an Open LDAP server.

The path to the keytab (specified by the `example2.keytab` property in the example) can either be an absolute path or simply the file name. If the file name is provided, the system assumes the file is found in the `security` directory in the base configuration directory (`oedq_home/security`).

If the `inetOrgPerson` schema is used, the `example2.ldap.profile` property in the example should be set to `inetorgopenldap`.

```
# OpenLDAP Example
realms = example 2
# Map the realm to a domain name
example2.realm                      = EXAMPLE2.COM
# Do not use local machine credentials to connect to the LDAP server
example2.clientcreds                = false
# Specify the Service Principal Name to use
example2.spn                        = host/host2.example2.com@EXAMPLE2.COM
# The keytab where the SPN can be found.
example2.keytab                     = kerberos.ktab
# Authorize user by using LDAP bind to server
example2.auth                       = ldap
# Use simple authentication, using the distinguished name
example2.auth.bindmethod            = simple
example2.auth.binddn                = search: dn
# Specify the LDAP server
example2.ldap.server                = ldapserver.example2.com
# Specify the base distinguished name.
example2.ldap.basedn                = dc=example2,dc=com
# Use the LDAP schema based on RFC2307, user are assumed to have
# the posixAccount object class
example2.ldap.profile               = rfc2307ldap
# Use Transport Layer Security
example2.ldap.security              = tls
# The name of the user group containing all EDQ users
example2.ldap.prof.defaultusergroup = group2
```

2.8.4 Example of Novell eDirectory LDAP Configuration

This section describes typical `login.properties` settings required to integrate EDQ with a Novell eDirectory LDAP server and what to do if a `novell.properties` file is required.

Section 2.8.4.1, "Example Settings for login.properties"

Section 2.8.4.2, "Creating a novell.properties File"

2.8.4.1 Example Settings for login.properties

The following are typical settings for integrating EDQ with Novell eDirectory.

```
# Novell eDirectory Example
# Map the realm to a domain name
example4.realm                      = EXAMPLE4.COM
# The base distinguished name is example4.com
example4.ldap.basedn                = o=example4
# Authorize user by using LDAP bind to server
example4.auth                       = ldap
```

```
# Use distinguished name for authentication
example4.auth.binddn                = search: dn
# The LDAP server
example4.ldap.server                = server4
example4.ldap.auth                  = simple
# Use Novell Style LDAP schema
example4.ldap.profile               = novell
# The name of the user group containing all EDQ users
example4.ldap.prof.defaultusergroup = group4
# Use Transport Layer Security
example4.ldap.security              = tls
# The eDirectory user credentials to be used by EDQ
example4.ldap.user                  = cn=intuser,ou=users,o=example4
example4.ldap.pw                    = password
```

2.8.4.2 Creating a novell.properties File

The EDQ installation does not come with a preconfigured profile for integrating with Novell eDirectory LDAP. If required, a `novell.properties` file must be created and saved in the `security/profiles` directory in the local configuration directory (`oedq_local_home`). The following is an example of this file.

```
# Simple LDAP profile for the Novell eDirectory server
# ------------------------------------------------------

idmatch          = (.*)@${realm:.*}

userattributes   = uid givenName sn mail telephoneNumber
usersearch       = (objectClass=inetOrgPerson)
userfilter       = +(uid={1})
userkey          = GUID
userfind         = +(GUID={0})
username         = uid

# group lookup

groupsearch      = (&(objectClass=groupOfNames)(cn=*))
groupkey         = GUID
groupfind        = +(GUID={0})
groupnamefind    = +(cn={0})

# secondary user/group relationships

memberattr       = member
membertarget     = dn

# certificate support

certuserfilter   = +(userCertificate={der})

# vcard

vcard.fn         = fullName cn
vcard.org        = o
vcard.tel.work   = telephoneNumber
vcard.email.pref = mail

# support attributes

binaryattrs      = GUID
```

2.9 Customizing Password Expiry Settings

You can customize the message that is presented to users if their password expires or is about to expire. By default, a standard dialog is displayed. This feature is available for users of Active Directory and any other LDAP server that supports the standard LDAP password policy response control.

To customize the mesage, you add variables to the `login.properties` file in the `security` directory of the local configuration directory (`oedq_local_home/security`).

2.9.1 Overview of the Variables

The following variables can be used in the password messages:

- {0} - The standard pre-configured Password Expired message.
- {1} - The name of the user.
- {2} - The number of days left before the password expires.

The realm name for each value is also displayed using the standard global realms property. The displayed realm name may be overridden using the *`realm_name`*`.label` property.

2.9.2 Customizing the Password Expired Message

Do either of the following to configure the Password Expired message.

- To use the standard Password Expired message, enter the following value in the `login.properties` file:

  ```
  realm_name.extra.pwexpired.message = {0}
  ```

- To enter a custom Password Expired message, with a link to a specific URL for changing the password, use the following code in the `login.properties` file. The message text, formatting and URL in this code are included as examples for you to edit as required. The HTML formatting is optional.

  ```
  realm name.extra.pwexpired.message = <html><font size="+1">Dear
  <em>{1}</em><p>Your password has expired. Click <a
  href="[URL]">here</a> to set a new password.</p></font></html>
  ```

2.9.3 Customizing the Password Expiring Message

Use the following property in the `login.properties` file to create a custom Password Expiring message. Substitute the correct realm name and message text. The HTML formatting is optional.

```
realm_name.extra.pwexpiring.message = <html><font size="+1">Dear
<em>{1}</em><p>Your password will expire {2,choice,0#today|1#tomorrow|1<in
{2} days}.<p>Click <a href="[URL]">here</a> to manage your password
settings.</p></font></html>
```

If you do not set a Password Expiring message, users will see the normal login screen if their password is about to expire.

2.9.4 Customizing the Expiry Time

By default, EDQ inherits the password expiry time from the LDAP server. By default, the number of days before the expiry time that users see the Password Expiring message is set to seven days. If required, you can set a custom expiry time and warning threshold in the `login.properties` file to override the settings in the LDAP server.

- To set a custom password expiry time, add the following value to the `login.properties` file:

  `realm_name``.ldap.prof.passwordhandler.passwordage =` *`Number_of_days/hours/seconds`*

- To set a custom warning threshold, add the following value:

  `realm_name``.ldap.prof.passwordhandler.passwordwarning =` *`Number_of_days/hours/seconds`*

  For each value, you can specify a number of days, hours or seconds. For example:

  - For 30 days set the value to `30` or `30d`.
  - For 240 hours enter `240h`.
  - For 3000 seconds enter `3000s`.

2.10 Configuring Parent and Child Active Directory Domains

Parent and child domains in Active Directory can be defined in the `login.properties` file by defining separate realms for each domain. However, if these domains have full trust relationships, it is possible to define only the parent domain as a realm, as shown in the following examples.

2.10.1 Example Settings for Parent and Child Domains

The following example assumes there is a parent domain `EXAMPLE.COM` and a child domain `CHILD.EXAMPLE.COM`. It provides example settings that illustrate how to configure `login.properties` for parent and child domains with only the parent domain configured as a realm. An explanation of this example follows the code segment.

```
# Global settings
clientcreds = true
realms = internal, parent
ldap.prof.useprimarygroup = false
# Realm settings
# Update match pattern to allow child domain components
child.ldap.prof.idmatch = (?i)(.*)@(?:.*\\.)?${realm:.*}
parent.realm = EXAMPLE.COM
parent.auth = ldap
parent.auth.bindmethod = simple
parent.auth.binddn = search: dn
parent.ldap.security = tls
parent.ldap.profile = adsldap
parent.ldap.prof.defaultusergroup = edqusers
parent.ldap.referral = follow
```

The settings for the parent domain are mostly the same as for a single AD domain. The significant differences are the following entries in the code.

parent.ldap.referral = follow
This property and its `follow` setting enable LDAP *referrals*. In a referral, when a search completes on the parent domain, it issues a referral reply that causes the search to

continue in the child domain(s). For example, a single search can return all the groups in the parent and child domains.

parent.ldap.prof.idmatch = (?i)(.*)@(?:.*\\.)?${realm:.*}
This setting uses the `idmatch` property to select the realm based on the identity of a user. After a Kerberos/SSO handshake, the server obtains the identity of the client from the handshake and then determines which realm is associated with the user. The property is a regular expression, where `${`*`realm_name`*`:.*}` is replaced with the *`realm_name`* from `login.properties`. The default value for single-domain AD is:

```
(?i)(.*)@${realm_name:.*}
```

In this case, the value expands to:

```
(?i)(.*)@EXAMPLE.COM
```

The value will match any user in the domain, such as `john.doe@EXAMPLE.COM`. The updated version adds the optional child domain component and would also match names like `jane.doe@CHILD.EXAMPLE.COM`.

parent.auth.bindmethod = simple
parent.auth.binddn = search: dn
A username and password are authenticated against AD by attempting a bind as that user. With parent and child domains, a user from the child domain can connect to the parent domain controller. However, the common `DIGEST-MD5` bind method does not work across domains, so set the bind method to `simple` and specify that the bind user name is the `Distinguished Name (DN)` of the user. For example:

```
CN=John Doe,OU=testusers,DC=parent,DC=com or CN=Jane
Doe,OU=localusers,DC=child,DC=parent,DC=com
```

parent.ldap.prof.defaultusergroup = edqusers
The default user group is used to find all the users who may need to use the EDQ applications. Users in this group appear in issue and case assignment lists, among other places in the EDQ user interface. The default group can be in either domain but must have Universal Scope, allowing it to contain members from both domains.

Groups that are used to assign EDQ permissions can be created as Universal and contain members from both domains, or they can be created as Global in each domain and contain users from the same domain.

The list shown on the `Administration > External Groups` configuration page contains groups from both domains. If no filter was set up, the page displays two of each of the standard groups (two Domain Users, two Backup Operators, and so forth). If you create EDQ-related groups in both domains, give them different names in each domain so that you can distinguish them in the list.

When only the parent domain is configured as a specific realm, EDQ treats both parent and child domains as a single realm. In EDQ the identity of a user is `user@REALM`. Given an identity of `user@EXAMPLE.COM`, for example, users from both domains will appear with `@EXAMPLE.COM` in assignment lists and in the user lists from the System Information Data Store.

The format of the display name of the user is configurable. You can set it to the `userPrincipalName` attribute of each user, for example, by using the following line:

```
parent.ldap.prof.userdisplayname = userPrincipalName
```

2.11 Kerberos Keytabs for Active Directory Accounts

When EDQ is installed on a UNIX (Solaris, Linux, AIX, or HP-UX) server, you can configure it to use AD for user authentication by making LDAP connections to the AD server and performing user lookups.

In a basic configuration, the connection to AD is made with a user name and password configured in `login.properties`. The connection can be protected using SSL or TLS if necessary. SSO, in which the user logs into Windows and then does not need to log again into EDQ, is not available in this configuration since the EDQ server is not on the AD domain.

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) as follows:

1. The client contacts the Domain Controller (DC) to request access to a service provided by the server application.
2. The response from the DC is encoded into a token sent to the server by the client.
3. The server validates this token and generates another token to send to the client.
4. 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 startup the server application sets up accept credentials, which it uses to initialize its half of the security context. When the server is running as the local system account on Windows, these credentials are obtained from the account's login context.

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

2.11.1 What is a Keytab?

A Kerberos key table, or keytab, 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 in the format *`service/hostname`*; for example:

`HOST/testserver.example.com`

Each entry in a keytab also includes a Key Version Number (KVNO). This 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 keytab is:

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

In the EDQ `login.properties` configuration file, the location of the keytab may be set by the following property:

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 only lists the principals; the second also includes the encryption algorithms, and the third also includes the key values in hexadecimal.

Following is sample output using `klist`:

```
Keytab name: WRFILE:/etc/krb5.keytab
KVNO Principal
 --------------------------------------------------------------------------
 2   host/testserver.example.com@EXAMPLE.COM (DES cbc mode with CRC-32)
 2   host/testserver.example.com@EXAMPLE.COM (DES cbc mode with RSA-MD5)
 2   host/testserver.example.com@EXAMPLE.COM (ArcFour with HMAC/md5)
 2   host/testserver@EXAMPLE.COM (DES cbc mode with CRC-32)
 2   host/testserver@EXAMPLE.COM (DES cbc mode with RSA-MD5)
 2   host/testserver@EXAMPLE.COM (ArcFour with HMAC/md5)
 2   TESTSERVER$@EXAMPLE.COM (DES cbc mode with CRC-32)
 2   TESTSERVER$@EXAMPLE.COM (DES cbc mode with RSA-MD5)
 2   TESTSERVER$@EXAMPLE.COM (ArcFour with HMAC/md5)
 2   HTTP/testserver.example.com@EXAMPLE.COM (DES cbc mode with CRC-32)
 2   HTTP/testserver.example.com@EXAMPLE.COM (DES cbc mode with RSA-MD5)
 2   HTTP/testserver.example.com@EXAMPLE.COM (ArcFour with HMAC/md5)
 2   HTTP/testserver@EXAMPLE.COM (DES cbc mode with CRC-32)
 2   HTTP/testserver@EXAMPLE.COM (DES cbc mode with RSA-MD5)
 2   HTTP/testserver@EXAMPLE.COM (ArcFour with HMAC/md5)
```

GSSAPI requires that an SPN has a service component (before the /), though there is no requirement that the rest is a valid host name or that the service is meaningful. An SPN in the following form is equally valid:

hello/alpha.beta

In a normal Kerberos system using a standard Kerberos Domain Controller (KDC) each SPN is a separate principal with a different password. In AD, 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/`*testserver.example.com testserver*$

`setspn -A` *hello/alpha.beta alpha.beta*

The first command adds an HTTP SPN to the machine account for `testserver`; the second adds an SPN to a normal user account.

The Apache Directory Studio LDAP browser can be used to check on the SPNs associated with an account. If a connection to AD can be made with administrator privileges, it can also be used to add `servicePrincipalName` values. For more information, see Section 2.11.4.2, "Apache Directory Studio."

2.11.2 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. Existing options include:

- 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.

2.11.3 UNIX Kerberos Configuration

The Kerberos configuration (as used by commands such as `kinit` and the JRE) is read from a global configuration file, normally stored in the `/etc/krb5.conf` directory. This contains references to the Domain Controllers and mappings between DNS and Kerberos domains.

This is a simple example of such a configuration file 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 = adsrvr01.example.com:88
kdc = adsrvr02.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.

The `krb5.conf` file must be checked and adjusted for the configuration of the target domain. If it is not possible to update a file in `/etc`, the file can be stored elsewhere and a system property can be used to inform the JRE where it is. To do this, edit (or create, if it does not yet exist) the `jvm.properties` file in the EDQ configuration directory and add the line:

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

2.11.4 Managing LDAP Accounts

There are two tools that can be used to examine and update LDAP accounts: Oracle Directory Services Manager (ODSM) and Apache Directory Studio.

2.11.4.1 Oracle Directory Services Manager

ODSM is bundled with OID. For further information on how to use this tool for managing LDAP accounts, refer to the OID server administrators.

2.11.4.2 Apache Directory Studio

The Apache Directory Studio LDAP browser is a useful tool for examining and updating LDAP accounts.

To create a new account, use the following procedure:

1. Launch the browser and close the Welcome window.
2. Select **New Connection...** in the LDAP menu to create a new connection.
3. Enter a name for the connection and the host name of an AD server.
4. If the server supports TLS, select **Use StartTLS Extension** under Encryption Method.
5. In the Authentication window, enter an AD user name (in the format *`user@DOMAIN`* or *`SHORTDOMAIN\user`*) and password. The AD directory tree is then visible in the LDAP browser area

If connected as an account with Administrator-level privileges, other AD accounts can be updated. For example, to add an SPN to a normal user account, follow these steps:

1. Locate the user in the directory tree and click to see the user's LDAP attributes. Perform a search to find the user if there are a large number of objects.
2. Right-click the **Attributes** window and select **New Attribute...**.
3. Select `servicePrincipalName` as the Attribute type and click **Finish**.
4. The new attribute will appear in the list. Enter the required SPN as the value and press **Enter**.

Note: If this is attempted using an account that does not have Administrator privileges, it will fail at the final step when changes are committed to AD. Therefore, it is possible to practice this procedure using a normal user account without making changes.

2.11.5 Configuring SSO

For a EDQ server configured for SSO integration with AD, a typical basic configuration of `login.properties` is as follows:

```
clientcreds = false
keytab = /etc/krb5.keytab
realms = internal, ad
ldap.prof.useprimarygroup = false
# Realm details
ad.realm = DOMAIN
ad.auth = ldap
ad.auth.bindmethod = digest-md5
ad.auth.binddn = search: sAMAccountName
ad.ldap.security = tls
ad.ldap.profile = adsldap
ad.ldap.auth = simple
ad.ldap.user = user@DOMAIN
ad.ldap.pw = password
ad.ldap.prof.defaultusergroup = groupname
```

If the SPN used for SSO between client and server is not the default for the machine (`HOST/machinename.domain@DOMAIN`) then add a line like:

```
spn = hello/john.smith@EXAMPLE.COM
```

The SPN must be listed in the keytab.

If the keytab contains an entry for the internal machine or user account name (for example, without a service or prefix), then it is possible to use SASL and GSSAPI authentication between EDQ and the AD server. You would amend the realm details section in `login.properties` to:

```
# Realm details
ad.realm = DOMAIN
ad.auth = ldapad.auth.bindmethod = digest-md5
ad.auth.binddn = search: sAMAccountName
ad.ldap.security = tls
ad.ldap.profile = adsldap
ad.ldap.spn = "accountname"
ad.ldap.prof.defaultusergroup = groupname
```

The account name is *`machinename$`* for a machine account or the login name for a user account.

If the keytab contains HTTP service entries for the machine, then it is also possible to use SSO for browser-based logins (administration application, dashboard, etc). To enable this, add the line:

```
http.gss = true
```

Typically, you this should set this only if all the client machines will be part of the domain. If SSO is not possible, the behavior of browsers varies. For example, Internet Explorer will show a login dialog in a pop-up window while Firefox will revert to the normal EDQ login pages.

3 Filtering User Authorization Groups

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:

- Section 3.1, "Installing the Authorizations Plug-In"
- Section 3.2, "Configuring the Authorizations Plug-In"

The sample Authorizations Filter plug-in is designed to address legislative requirements that state 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.

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 run-time 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 should 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 then be granted access to each group as usual, and the plug-in script will provide per-session, IP address-based filtering of the groups actually 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 all 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.

3.1 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/user_projects/domains/your domain/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 3.1.1, "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 3.2, "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.

3.1.1 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, '');
}
```

3.2 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.

3.2.1 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.

3.2.2 CSV File Format

The CSV configuration file format specifies one group and address range mapping per line. Each line is 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 the sample XML file in Section 3.2.1, "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.

4 Configuring SSL with Tomcat

This chapter provides instructions for setting up Secure Sockets Layer (SSL) on an Oracle Enterprise Data Quality (EDQ) running on Tomcat application server.

This chapter includes the following sections:

- Section 4.1, "Configuring SSL During Installation"
- Section 4.2, "Configuring SSL Client Authentication"
- Section 4.3, "Using SSL With JMX"

Note: For other application servers (such as Oracle WebLogic Server or IBM Websphere), consult the standard server documentation.

EDQ user applications (such as Director) always encrypt user passwords, so SSL is not required.

4.1 Configuring SSL During Installation

When installing Tomcat on Windows or any other platform, the HTTPS connector must be configured using the following procedure:

1. Locate the `server.xml` file for the Tomcat installation. Typically it contains the following:

```
<!-- Define a SSL HTTP/1.1 Connector on port 8443
This connector uses the JSSE configuration, when using APR, the
connector should be using the OpenSSL style configuration
described in the APR documentation -->
<!--
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS" />
-->
```

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

Note: The certificate is supplied in a Java keystore, either in the default 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:

   ```
   <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
   maxThreads="150" scheme="https" secure="true"
   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 `mod_ssl` style attributes. For example:

   ```
   <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
   maxThreads="150" scheme="https" secure="true"
   SSLCertificateFile="pathtocrtfile"
   SSLCertificateKeyFile="pathtokeyfile" />
   ```

For additional Tomcat information, see *Apache Tomcat Configuration Reference* at

http://tomcat.apache.org/tomcat-7.0-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

4.2 Configuring SSL Client Authentication

EDQ can support authentication using SSL client certificates.

There are three stages to configuring SSL client certificates:

1. Configure the server to request client certificates.
2. Assign a personal certificate and associated private key to each user.
3. Associate each certificate with an internal EDQ user or an entry in an external LDAP server.

4.2.1 Configuring Tomcat to Support Client Certificates

1. Locate the HTTPS connector and add the following settings:

   ```
   clientauth="true"
   truststoreFile="pathtotruststore"
   truststorePass="truststorepassword"
   truststoreType="truststoretype"
   ```

   a. Set the `clientauth` attribute to `true` (valid client certificate required for a connection to succeed) or want (use a certificate if available, but still connect if no certificate is available).

b. Add the location of the trust file containing the certificate issuers for trusted client certificates.

c. Set `truststoreType` to `JKS` or `PKCS12`.

2. If the Tomcat installation includes Apache Portable Runtime (APR), then the equivalent `mod_ssl` settings are used:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
SSLCertificateFile="pathtocrtfile"
SSLCertificateKeyFile="pathtokeyfile"
SSLCACertificateFile="pathtocabundlefile"
SSLVerifyClient="require" />
```

3. Locate the EDQ `login.properties` file in the `security` subdirectory of the `configuration` directory. If the file does not exist, create it by copying the `login.properties` file from oedg.home.

4. Add the following line to the `login.properties` file to enable authentication for all realms using X.509 certificates:

```
x509 = true
```

Note: To enable X.509 certificate for specific realms, add a line of code for each realm, including the realm name as a prefix. For example, for the dn realm, add the following line:

```
dn.x509 = true
```

5. If required, enable certificate authentication for web pages in all contexts by adding the following setting:

```
http.x509 = true
```

Alternatively, to enable selectively for different contexts, add the context as a suffix to the setting; for example:

```
http.x509.admin = true
http.x509.formws = true
http.x509.ws = true
http.x509.dashboard = true
```

4.2.2 Assigning Personal Certificates and Key Combinations

When SSL client authentication is enabled on a server, each user must have a certificate and associated private key available on the client. The certificate is not sensitive and can be distributed freely, but the private key must be stored and distributed securely.

Each certificate and key combination user can be stored in a number of ways:

- In the operating system certificate store. For example, Internet Options, Content, and Certificates on a Windows platform.
- A smart card.
- A USB dongle.

Certificate and key combinations can either be generated and distributed to users or created by a certificate authority website, allowing users to apply for one as required.

The latter approach is preferable, because the private key is generated on the system of the user and therefore is not transmitted.

On Windows platforms, Java Web Start uses the operating system certificate store in addition to the internal Java store. If a certificate has been created in Internet Explorer (or Google Chrome), it is stored in the system store and will be used by Web Start. Mozilla Firefox has an internal certificate store; certificates generated in Firefox must be added manually to the system or Java store before use with Java Web Start.

4.2.3 Associating Certificates With a User

Once a user has a certificate, it must be associated with an EDQ user account to enable automatic authentication. This association provides an alternative way of performing pass-through authentication, or single sign-on (SSO), whereby users do not need to log in to EDQ applications.

4.2.3.1 Internal Users

A Java Management Extensions (JMX) interface and associated script can be used to store the certificate in the user record:

```
java -jar jshell.jar setcert.groovy -server SERVER:JMXPORT
[ -username adminusername -pw password ]
[ -cert certfile ] -for username
```

- *SERVER* is the host name of the EDQ server and *JMXPORT* is the port used with jconsole.
- *adminusername* and *password* are for an EDQ administrator (they can be omitted if SSO is enabled and jshell is run from the tools directory).
- *certfile* is the file containing the certificate, in PEM or DER format. If `-cert` is omitted, any existing certificate is removed from the user record.
- *username* is the internal user being updated.

4.2.3.2 External Users

The certificate is stored in an attribute for the user in the external LDAP server. For example, in Active Directory the `userCertificate` attribute is used.

4.3 Using SSL With JMX

EDQ supports the use of SSL to connect to JMX. Because the connector for JMX is created within EDQ, the configuration of SSL is performed by editing the `director.properties` file and not by altering the configuration of the application server.

4.3.1 Enabling the SSL Settings for JMX

Use the following steps to enable SSL for use with JMX:

1. To enable SSL for JMX add the following line to the `director.properties` file:

   ```
   management.ssl.port = portnumber
   ```

   - Replace *portnumber* with the required port number.
   - The key and certificate for the connection can be specified in separate files or in a Java keystore.

2. To use separate `crt` and `key` files in Privacy-enhanced Electronic Mail (PEM) or Distinguished Encoding Rules (DER) format, add these settings:

   ```
   management.ssl.km.crt = crtfile
   management.ssl.km.key = keyfile
   ```

3. If the key is encrypted, add the following setting to set the key password:

   ```
   management.ssl.km.keypw = password
   ```

   However, if the certificate is in a Java keystore, use these settings:

   ```
   management.ssl.km.keystore = keystorefile
   management.ssl.km.storetype = storetype
   management.ssl.km.storepw = storepassword
   management.ssl.km.alias = alias
   management.ssl.km.keypw = keypassword
   ```

   - The alias property can be omitted if the key store contains a single entry; otherwise it is the alias of the certificate and key entry in the store.
   - The store type defaults to `JKS`.
   - The `keypw` can be omitted if the password for the key is the same as the password for the store (this is typically the case).

4. To enable SSL client authentication for JMX, add the following setting:

   ```
   management.ssl.clientauth = required
   ```

   Replace required with optional to use a certificate if available, but to successfully connect if it is not.

5. To configure the issuer certificates for valid client certificates, add the following setting to accept any client certificate:

   ```
   management.ssl.tm.any = true
   ```

   Alternatively, add the following setting to specify a certificate bundle file (containing a series of concatenated PEM certificates):

   ```
   management.ssl.tm.bundle = bundlefile
   ```

   Or add the following settings to specify a keystore:

   ```
   management.ssl.tm.keystore = keystorefile
   management.ssl.tm.storetype = keystoretype
   management.ssl.tm.storepw = keystorepassword
   ```

Note: The default keystore type is JKS, which does not require the `keystorepassword` setting.

4.3.2 Using SSL with JMX Clients

If SSL has been enabled for JMX and SSL client authentication is not enabled, no special configuration is required if the server certificate is issued by an authority trusted by the Java Runtime Environment (JRE).

If the issuer is not currently trusted, the Certification Authority (CA) certificate can be added to the JRE `cacerts` store using the `keytool` command supplied with the JRE.

Alternatively, a keystore containing the CA certificate can be supplied using the standard Java SSL properties.

For example:

```
jconsole -J-Djavax.net.ssl.trustStore=trustkeystorefile
```

If SSL client authentication has been enabled, key store properties are also required:

Notice the parallel with server configuration. The client trust store is used to trust the certificate in the server key store; the client key store contains the certificate that is trusted by the server trust store (or certificate bundle).

```
jconsole -J-Djavax.net.ssl.trustStore=trustkeystorefile
-J-Djavax.net.ssl.keyStore=keystorefile
-J-Djavax.net.ssl.keyStoreType=keystoretype
-J-Djavax.net.ssl.keyStorePassword=keystorepassword
```

See the JRE documentation for details of the `java.net.ssl` property set.

The JMX command line tools (and the `jshell` script interpreter) also support setting SSL configuration with environment variables and/or command line arguments. Not all JMX scripts support SSL options; if not available, use the environment variables.

If SSL client authentication is not enabled, use the `EDQ_SSL_TRUST` environment variable or `-ssltrust` command line option to specify a Java keystore containing the CA certificate for the server (this is analogous to the first `jconsole` example in this section).

If SSL client authentication has been enabled, use the `EDQ_SSL_PROPS` environment variable or `-sslprops` command line argument to specify a properties file containing key and trust store settings. The property format is identical to the server configuration in director.properties except that the `management.prefix` is not used.

For example, to specify trust for the server certificate using a Java keystore, and the client certificate and key as separate crt and key files the properties file would contain:

```
tm.keystore = trustkeystorefile
km.crt = crtfile
km.key = keyfile
```

Key and store password, and other properties can be added as necessary.

The property file contents can be specified directly in the environment or on the command line by enclosing the property settings in `{..}` and separating property values with commas. For example, the preceding property file would be specified as:

```
{tm.keystore=trustkeystorefile, km.crt=crtfile, km.key=keyfile}
```

4.3.2.1 Command Examples

This section contains examples of some of the commands described in the previous section:

```
java -jar jmxtools.jar runjob -job x -project z -sslprops c:\tmp\ssl.properties
localhost:9005
```

To specify SSL trust and key information in a properties file, using the command line option, you could use the following:

```
set EDQ_SSL_TRUST=c:\tmp\trust.jks
java -jar jshell.jar scripts\system\sysreport.groovy -user dnadmin -pw password
-server localhost:9005
```

To run a system report specifying a trust store using an environment variable. In UNIX, for example, the command might be:

```
EDQ_SSL_TRUST=/tmp/trust.jks
export EDQ_SSL_TRUST
java -jar jshell.jar scripts/system/sysreport.groovy -user dnadmin -pw password
-server localhost:9005
```

To use a client certificate with in line properties, the command might be:

```
EDQ_SSL_PROPS="{tm.keystore=/tmp/trust.jks,km.crt=/tmp/me.crt,km.key=/tmp/me.key}"
export EDQ_SSL_PROPS
java -jar jshell.jar scripts/system/sysreport.groovy -server localhost:9005
```

5 Using the Audit Framework with Enterprise Data Quality

This chapter explains how to enable and configure EDQ to log events with the Oracle Fusion Middleware Audit Framework.

This chapter includes the following sections:

- Section 5.1, "Enabling EDQ Audit Event Logging"
- Section 5.2, "Configuring the EDQ Events"

When you install EDQ to operate in an Oracle WebLogic Server domain, you integrate it to log events in the Oracle Fusion Middleware Audit Framework. This auditing provides a measure of accountability and answers the "who has done what and when" types of questions. For detailed information about this auditing service, see "Introduction to Oracle Fusion Middleware Audit Service" in *Oracle Fusion Middleware Securing Applications with Oracle Platform Security Services 12c (12.1.3)*.

5.1 Enabling EDQ Audit Event Logging

To enable audit event logging, use the following procedure:

1. Open the Enterprise Manager 11g Fusion Middleware Control application.
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.

5.2 Configuring the EDQ Events

The EDQ event categories and types are as follows:

Event Category	Event Types
User Management	Login, Logout, Password Change, Password Expire, User Blocked, User Blocked Temporarily, User Unblocked, User Created, User Updated, User Deleted, Security Configuration Updated.

Event Category	Event Types
Object Management	Create, Update, Delete.
Group Permission Management	Join group, Leave group, Leave all groups, Create group, Delete group, Change permissions.

Note: Object Management logs changes made to objects in the Project Browser of the Director application only, such as projects or processes.

It does **not** cover changes to objects made in other applications, such as Case Management.

The attributes that can be logged by event are listed in the following table. Note that not every attribute is available to each event type.

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

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 12c (12.1.3)*.

6

Integrating EDQ with a Fusion Middleware Credential Store

This chapter describes how to use an Oracle Fusion Middleware credential store with EDQ.

This chapter contains the following sections:

6.1 Overview of the Credential Store

EDQ supports the use of the Oracle Fusion Middleware credential store to hide user names and passwords that are used by EDQ to connect to protected resources, such as a JMS broker or LDAP server. These credentials otherwise would be exposed as clear-text in the EDQ properties files. When a credential store is used, a user name and password are replaced by a key name that serves as an alias for the credential whenever a login is required.

Using a credential store with EDQ comprises the following steps:

Configuring the Credential Store for EDQ

Specifying the EDQ Credential Key in Properties Files

6.2 Configuring the Credential Store for EDQ

To configure a credential store, use Oracle Enterprise Manager Fusion Middleware Control. For more information about using this browser-based console, see *Administering Oracle Fusion Middleware*.

In a credential store, a credential is identified by a *credential map*. The credential map consists of a *map* and one or more *keys*. In EDQ, the default map name is `edq`. The key name is specified by the person who is creating the credential map and serves as the "alias" for the credential in the properties files. The person who creates the credential map must be an Oracle Fusion Middleware administrator.

To Configure a Credential Store for EDQ

1. Log in to Oracle Enterprise Manager Fusion Middleware Control as an administrator.

2. Navigate to *Domain* > **Security** > **Credentials** to display the Credentials page.
3. Click **Create Map** to display the Create Map dialog. Once you create a map, you can create multiple keys for it at the same time, or you can add more keys at a later date.
4. Create a map named `edq`, and then click **OK**. The `edq` map name is displayed in the table.
5. Click **Create Key** to display the Create Key dialog.
6. Select the following in this dialog:
   - Select the `edq` map from the **Select Map** pull-down menu.
   - Enter a name for the key in the **Key** text box. This is the key name that will be entered in the properties files to replace the credential.
   - Select **Password** from the **Type** pull-down menu.
   - Enter the user name for the EDQ user in the **User Name** field and enter the password for that user in the **Password** field. Confirm the password in the **Confirm Password** field.
   - Optionally, you can add a description of this credential.
7. Click **OK** to return to the Credentials page. The new key is displayed under the `edq` map icon.

6.3 Specifying the EDQ Credential Key in Properties Files

Once you have configured an EDQ credential map in Fusion Middleware Control, use the `.cred.key` property to specify the key name in place of the credential in properties files.

The syntax is this:

```
prefix.cred.key = keyname
```

It replaces the standard, non-secured `username` and `password` entries:

```
prefix.username = username
prefix.password = password
```

The following shows an entry for a credential for user “myuser”, followed by an entry for the same credential as represented by its key name.

Non-secured Credential in director.properties

This example shows the regular way of using the `username` and `password` properties to specify the actual user name and password.

```
sccs.vcs.username = myuser
sccs.vcs.password = mypassword1234
```

Secured Credential in director.properties

This example uses the `cred.key` property to specify a key name from the credential store in place of the login credential.

```
sccs.vcs.cred.key = mykey1
```

Secured Password-Only Entry

In cases where only a password is required, for example if creating a keystore for JMX over SSL, append the `.cred.key` property to the property name. The following is an example:

```
management.ssl.km.storepw.cred.key = mykey1
```

6.4 Examples of Specifying a Key Name

These examples show additional ways to specify credentials by means of a key name.

Connection to a JMS Broker

This example shows a realtime bucket definition in which a credential is required to connect to a JMS broker.

The following is the unsecured way of specifying the credential:

```
<messengerconfig>
  …
  username = myuser
  password = mypassword1234
 …
</messengerconfig>
…
```

The following is the secure specification using the key name:

```
…
<messengerconfig>
  …
  cred.key = mykey1
 …
</messengerconfig>
…
```

Connection to a JNDI Store

This example uses a credential to connect to a JNDI store.

The following is the unsecured way of specifying the credential:

```
…
<messengerconfig>
  …
  java.naming.security.principal   = myuser
  java.naming.security.credentials = mypassword1234
 …
</messengerconfig>
…
```

The following is the secure specification using the key name. In this case, the `jndi` prefix is required, so the .cred.key is appended to it.

```
…
<messengerconfig>
  …
  jndi.cred.key = mykey1
 …
</messengerconfig>
…
```

Connecting to an LDAP Server

This example shows the correct syntax for specifying a connection to an LDAP server in the `login.properties` file.

Non-secured entry:

```
myrealm.ldap.user = myuser
myrealm.ldap.pw = mypassword
```

Secured entry with credential store key:

```
myrealm.ldap.cred.key = mykey1
```