Oracle® Enterprise Data Quality

LDAP Integration Guide

Release 11g R1 (11.1.1.7)

E40042-02

October 2013

This document describes how Oracle Enterprise Data Quality (EDQ) can be integrated with external user management systems based on the Lightweight Directory Access Protocol (LDAP) standard, thus allowing Administrators to manage user accounts externally to EDQ.

EDQ can be integrated with LDAP servers in two ways:

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

Once integrated, external user management works in the same way. Groups of users that exist on the LDAP system ("External Groups") are mapped to groups in EDQ using the Administration pages on the EDQ Launchpad in order to grant users in the external groups permissions to the EDQ system.

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

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

In integrations with AD, it is possible to enable Single Sign-On (SSO), allowing authorized users of the AD domain to access EDQ without the need to log in to the EDQ user applications.

1 Audience

This document is intended for experienced administrators responsible for integrating EDQ with external user management systems.

2 LDAP Integrations using OPSS on the WebLogic Server

In a default installation of EDQ on WebLogic Server, the integration with OPSS is enabled by default. This means that users are managed by the identity store configured in WebLogic.

The integration is enabled by the installation of the file `login.properties` in `oedq_home/security`.

A setting in this file configures the default mapping for administrators that ensures that EDQ's Administration interface on the EDQ Launchpad can be accessed by a

WebLogic Administrator in order to complete the mapping of external groups to internal groups.

Provided the WebLogic identity store (either WebLogic itself, or a configured LDAP server) has an "Administrators" group, there is no need to adjust any of the settings in login.properties in a WebLogic Server environment.

However, if the LDAP server does not contain an "Administrators" group, it is necessary either to:

- create an "Administrators" group on the LDAP server and restart the server managing EDQ in WebLogic, or
- modify the default administrators group mapping (as detailed in the following section).

2.1 Modifying the Default Administrators Group Mapping

If the LDAP server configured for use in WebLogic does not contain a group with a name of "Administrators", or if the internal EDQ "Administrators" user group has been deleted, it is necessary to override the shipped login.properties file with a new version that adjusts the default Administrators group mapping.

To do this:

1. Create a subdirectory called `security` in `oedq_local_home`.
2. Copy the shipped `login.properties` file from `oedq_home/security` and paste it into `oedq_local_home/security`.
3. Modify the group mapping line from:

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

   to:

   `opss.xgmap =` *`name of external group`* `->` *`name of EDQ group`*

   The EDQ group name normally being "Administrators", although this is not guaranteed.

   > **Note:** The property accepts a delimited list of mappings. For example, the following syntax is valid for mapping both "DQ Admins" and "Administrators" external groups to the internal EDQ "Administrators" group:
   >
   > `opss.xgmap = Administrators -> Administrators, DQ Admins -> Administrators`.

4. Save the file
5. Restart the application server

> **Note:** This default mapping is important in order to enable initial configuration of group permissions. Once this task has been completed, other external groups may be granted administration rights to EDQ. The mapping specified in the file is a fixed mapping that is displayed, but cannot be adjusted, using the Administration pages on the EDQ Launchpad.

3 Direct LDAP Integrations

For installations of EDQ on other application servers (IBM WebSphere, or Apache Tomcat), EDQ offers direct integration with LDAP servers. This is again driven by the login.properties file, but by default, no external LDAP integration is enabled. Instead, a template is provided (`login.properties.template`) with example settings corresponding to the different supported LDAP providers

3.1 Configuring a Direct LDAP Integration

To integrate EDQ with an LDAP server directly, it is necessary to create and configure a new version of the `login.properties` file in the `oedq_local_home > security` directory. This file will overrride settings of the `login.properties` file in the `oedq_home > security` directory.

To do this, use the following procedure:

1. Navigate to the `security` directory in the EDQ Local Home configuration directory.
2. Open the `login.properties.template` file with a text editor.
3. Uncomment and edit the parameters to correspond with the LDAP server on the installation environment.
4. Save the file as `login.properties` in the same directory.
5. Restart the application server.

The remainder of this document expands on step, by describing the settings of the `login.properties` file in more detail.

3.1.1 LDAP Profiles

The profile associated with an LDAP configuration provides information about the schema used in the LDAP server used to represent users and groups. EDQ provides the following built-in profiles:

- adsldap: Microsoft AD.
- inetorgoidldap: OID.
- inetorgopenldap: OpenLDAP using inetOrgPerson style schemas.

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

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

4 Global Settings

EDQ supports integration with multiple realms. Each realm may use different LDAP server technologies. For example, a single EDQ server may support external authentication from both an AD realm and an OID realm, if required.

This section describes the `login.properties` properties that are normally set globally (that is, for all realms). Where noted, the settings may be overridden at the realm level

if required. Realm-level settings are more specific and will always override global settings if specified.

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. This property is only necessary in order 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. A single keytab must be defined at the global level. A single keytab can contain entries for several realms.	/etc/krb5.keytab Note that if a specific path is not specified, the Operating System default path is used.	No
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, t 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, t value is HOST/
x509	Enables the use of x509 certificates (client SSL certificates) for client authentication in EDQ. Note that there is a small performance cost associated with setting this to true. May be overridden at realm level.	true	No If not set, the d

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. This should be set to "false"for performance purposes unless the membership of the primary group has any relevance for EDQ. May be overridden at realm level	false	No

5 Realm Settings

This section provides details of the properties that are normally set at realm level. Realm settings may be specified in two different ways:

1. In the `login.properties` file, using the syntax *`realm name.property name = value`*.
2. In a file named *`realm name`*`.properties` in a realms subdirectory of the security directory.

Property	Description	Example Value	Mandatory?
realm	The LDAP (e.g. AD or Kerberos) domain name.	EXAMPLE.COM	Yes
ldap.profile	Specifies the LDAP profile name used to configure a number of 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 is used to specify which method is used to validate user credentials. See the following section 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.	example	No. If not used, the realm name fro property is use
gss	Specifies whether or not the realm sup-ports Kerberos/GSSAPI for SSO. Possible values are true or false.	false	No. If not set, defau

Property	Description	Example Value	Mandatory?
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*.	192.168.1.48:389, server2	No. If not specified, is used to look servers
ldap.basedn	Base of LDAP hierarchy.	dc=example, dc=com	No. In many server AD, this can be the RootDSE (t Directory Servi
ldap.security	Security mode for LDAP connection. Possible values are "ssl" or "tls".	tls	No 'tls' should be u possible.
ldap.auth	Authentication mode for LDAP connection. Possible values are "simple", "digest-md5" or "gss".	digest-md5	No If not specified, to 'gss'
ldap.user	The LDAP username used to authenticate EDQ with the LDAP server. **Note**: This property must be set if ldap.auth is not set to "gss".	cn=user, ou=users, dc=example3,dc=com	Yes, if authenti 'simple'. No, if the mode 'digest-md5'.
ldap.pw	The password associated with the LDAP username.	password	Yes, if authenti '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. See the *Global Settings* table in the preceding section for further information on this parameter.	true	No. If not set, t value is that sp Global level.
ldap.prof.defaultusergroup	Name of default group containing all EDQ users, used for display of users in issue, alert, and case assignment lists.	edqusers	Recommended If not set, this d 'Domain Users' which will ofte and cause displ memory issues
ldap.prof.groupsearchfilter	Additional filter for groups; an LDAP search filter, see RFC4515.	(cn=edq*) (This will include all groups with a name beginning with 'edq')	Recommended If not set, no fil all groups will the External Gr configuration p

5.1 Validating Credentials when SSO is not used

As described in the table in the preceding section, in installations where SSO is not used and the `auth` property is set to `ldap` it is necessary to specify how user credentials are validated with the `auth.method` property.

This property has three possible values, detailed in the following sections.

5.1.1 auth.method = bind

The EDQ server connects to the LDAP server to verify the user credentials. This is the default setting.

Where the bind method is used, two additional properties should be set:

`auth.binddn`: the actual user name 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`: The authentication method used to connect to the LDAP server. The possible values are `simple` or `digest-md5`. The latter value encrypts the password on the network, and is therefore the recommended setting.

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

5.1.2 auth.method = password

The EDQ server looks up the user record on the LDAP server, and compares the submitted password to the stored password.

> **Note:** This method cannot be used with AD servers.

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

auth.password = search: attr

where attr is the LDAP attribute.

5.1.3 auth.method = compare

The LDAP "compare" method is used to validate the password. This method is more secure than "bind" as a session is not created in the LDAP server.

The LDAP attribute used to store the password is specified with the auth.attribute property, which has a default value of userPassword. This default is the correct value for OID LDAP integrations.

6 Directory Structure

For convenience, it is possible to split the `login.properties` file into multiple files.

Settings for a realm may specified in the `login.properties` file using the realm name as the prefix, or in a separate file named *`realm`*`.properties` in the `security/`*`realms`* directory. Note, however, that the `realname` prefix is not used in the properties specified in the *`realm`*`.properties` file.

Similarly, profile settings and overrides can be specified in the `login.properties` file, or in a file named *`profilename`*`.properties` in the `security/`*`profiles`* folder. Again, the `realname` prefix is not used for properties in the *`profilename`*`.properties` file.

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) 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 on which EDQ runs.

8 Example Configuration: Oracle Internet Directory

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

These example settings are included in the provided `login.properties.template` file with an additional # character at the beginning of each line, which should be removed to uncomment each property.

```
# 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
```

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

9 Example Configuration: Active Directory

This section gives an example of typical `login.properties` settings required to integrate EDQ with an AD LDAP server.

These example settings are included in the provided `login.properties.template` file with an additional # character at the beginning of each line, which should be removed to uncomment each property.

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

```
# 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
```

10 Example Configuration: Open LDAP

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

These example settings are included in the provided `login.properties.template` file with an additional # character at the beginning of each line, which should be removed to uncomment each property.

```
# 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
```

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

Also, if the inetOrgPerson schema is used, the `example2.ldap.profile` property should be set to `inetorgopenldap`.

11 Example Configuration: Novell eDirectory LDAP

This section describes the typical `login.properties` settings required to integrate EDQ with a Novell eDirectory LDAP server.

```
# 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
```

The EDQ installation does not come with a preconfigured profile for integrating with Novell eDirectory LDAP. Therefore, if required, a novell.properties file must be created and saved in the `security/profiles` directory. The following is an example of such a 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
```

12 Appendix: Customizing Active Directory Password Expiry Settings

If a user of an EDQ installation integrated with AD logs in when their AD password has expired, a standard dialog is displayed.

It is possible to customize the message presented to users when their AD password has expired, or is about to expire. To do this, certain values must be added to the `login.properties` file in `oedq_local_home/security`.

> **Note:** This also applies to any LDAP server that supports the standard LDAP password policy response control.

12.1 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.

12.2 Customizing 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 (the message text, formatting and URL are included as examples, and can be edited 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>
```

12.3 Customizing the Password Expiring message

Use the following property to create a custom Password Expiring message, substituting 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.

12.4 Customizing the Expiry Time

By default, EDQ inherits the password expiry time from AD, and the warning threshold (the number of days before the expiry time that users will see the Password Expiring message) is set to seven days. If required, these values can be set in `login.properties` instead.

To set a custom password expiry time, add the following value:

```
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.

13 Appendix: Configuring Parent/Child Active Directory Domains

Multiple domains can be simply defined in login.properties by defining separate realms for each domain. This approach will also work for Parent/Child domains in AD. However, where these domains have full trust relationships, it is possible to define only the Parent domain.

The example in this appendix assumes there is a parent domain EXAMPLE.COM and a child domain CHILD.EXAMPLE.COM.

13.1 Example Settings for Parent/Child Domains

This section provides example settings that illustrate how to configure `login.properties` for parent/child domains with only the parent domain configured as a realm.

```
# 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. Significant differences are:

```
parent.ldap.referral = follow
```

This enables LDAP 'referrals'; when a search completes on the parent domain, it issues a referral reply causing 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:.*}
```

The `idmatch` property is used to select the realm based on the identity of a user. After a Kerberos/SSO handshake, the server obtains the client's identity from the handshake and then needs to determine which of the configured realms is associated with the user. The property is a regular expression with ${*realm*:.*} replaced with the realm name from login.properties. The default value for single-domain AD is:

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

In this case, it expands to:

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

It would 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/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, etc. The group can be in either domain but must have Universal Scope, allowing it to contain members from both domains.

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

The list shown on the `Administration > External Groups` configuration page will contain groups from both domains . If no filter has been set up you will see two of each of the standard groups (two Domain Users, two Backup Operators, etc). If you create EDQ-related groups in both domains, give then different names in each domain so you can distinguish them in the list.

13.1.1 Caveats

When only the parent domain is configured as a specific realm, EDQ effectively treats both parent and child domains as a single realm. This means that:

In EDQ the 'identity' of a user is user@REALM - in this case user@EXAMPLE.COM. So 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. It may be set to the `userPrincipalName` attribute of each user, for example, using the following line:

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

14 Appendix: Kerberos Keytabs for Active Directory accounts

Where EDQ is installed on a UNIX (Solaris, Linux, AIX, or HP-UX) server, it can be configured to use AD for user authentication. This is done 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 therefore an essential step in configuring SSO on UNIX.

14.1 What is the Keytab?

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

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

Here is some example output from 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 /), but 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. See the Using Apache Directory Studio later in this chapter.

14.2 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.

14.3 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 /etc/krb5.conf. 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`*

14.4 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.

14.4.1 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.

14.4.2 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. On 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.

14.5 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/ prefix), then it is possible to use SASL/GSSAPI authentication between EDQ and the AD server. The realm details section in `login.properties` is amended 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 pages, dashboard, etc). To enable this, add the line:

```
http.gss = true
```

Normally this should be done 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.

15 Related Documents

For more information, see the following documents in the documentation set:

- *Oracle Enterprise Data Quality Installation Guide*

See the latest version of this and all documents in the Oracle Enterprise Data Quality Documentation website at:

http://download.oracle.com/docs/cd/E48549_01/index.htm

Also, see the latest version of the *EDQ Online Help*, bundled with EDQ.

16 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.

Oracle Enterprise Data Quality LDAP Integration, Release 11g R1 (11.1.1.7)
E40042-02

Copyright © 2006, 2013, Oracle and/or its affiliates. All rights reserved.

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

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

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

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

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

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

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

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.