ORACLE
ENTERPRISE DATA QUALITY

An Oracle White Paper
November 2013

EDQ High Availability Guide

ORACLE

1. Introduction

The following High Availability (HA) implementation strategies are designed primarily for implementations in which Oracle Enterprise Data Quality (EDQ) is integrated with Siebel CRM/UCM, or other similar applications. However, the principles underlying these integrations can be applied to other environments.

In general, the document assumes that EDQ provides stateless services, with the main requirement for High Availability being the high availability of EDQ web services. High Availability of stateless batch jobs is also considered.

A further assumption is that multiple EDQ servers are used, and all servers have the same configuration (at least as far as the configuration that provides the HA services) – for example the same configuration of data quality services for matching and address verification, either modified or unmodified from the provided Customer Data Services Pack (EDQ-CDS).

This document states the basic rules for implementing High Availability for EDQ, and two example deployments; one based on many logical servers in a WebLogic cluster, and another based on multiple separate installations.

2. High Availability Models

EDQ customers have varying requirements for different levels of HA depending on their usage scenarios. This section gives some information on the common deployment models.

2.1 Real-time Customer Data Services

This is the most common deployment model, where EDQ provides highly available stateless DQ services to an external application such as Siebel.
The same model can be used for any other stateless real-time services to any external application, such as validation, cleansing, or matching services.

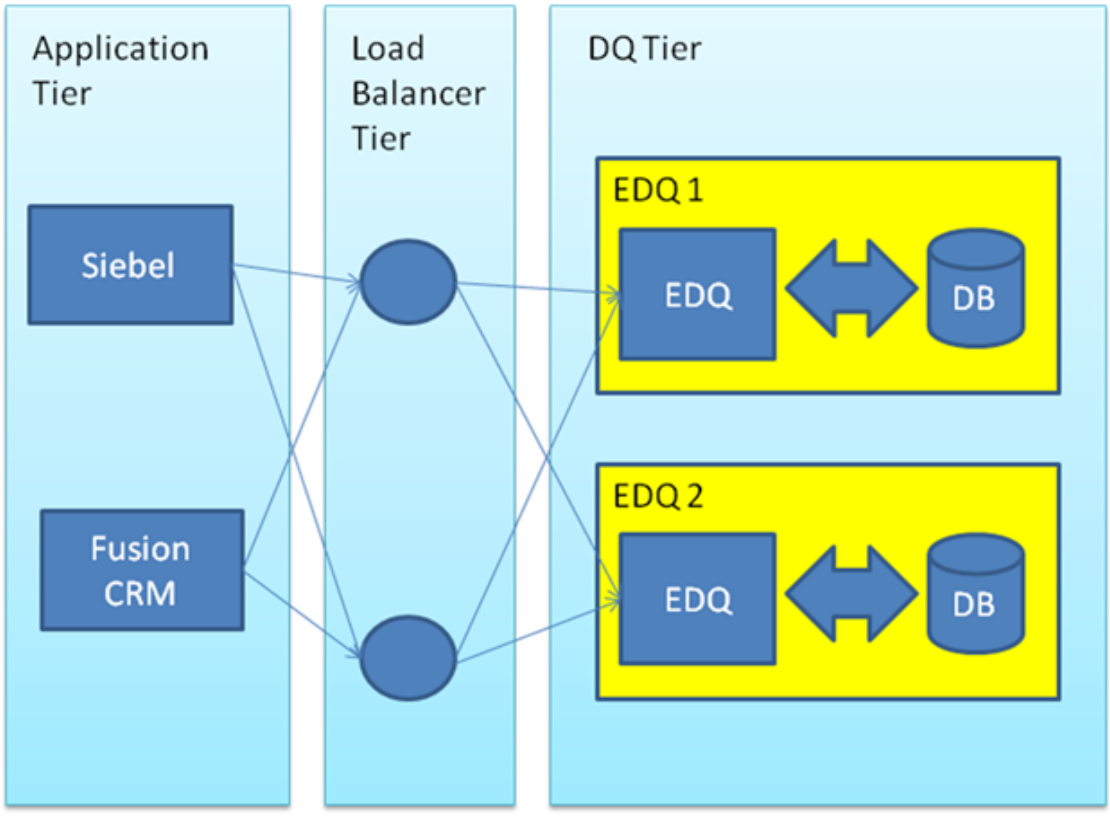

Figure 1: Stateless DQ Services to Applications

In this model, multiple EDQ instances, each with their own database schemas, are used. The instances have the same configuration, which can be controlled centrally using configuration change strategies such as the EDQ Subversion integration, or by the use of locked-down EDQ environments.

Since running EDQ in this mode is stateless, web services calls are received by the load balancing tier and distributed to each EDQ server. The same answer will be provided by any of the EDQ instances. In the event of failure the load balancer can direct traffic to the instances that are still running.

2.2 Real-time Reference Data Matching

This model is used where EDQ needs to provide a highly available service for matching records against slowly changing reference data, for example a service for matching customers or prospective customers against watch lists in an Oracle Watchlist Screening deployment.

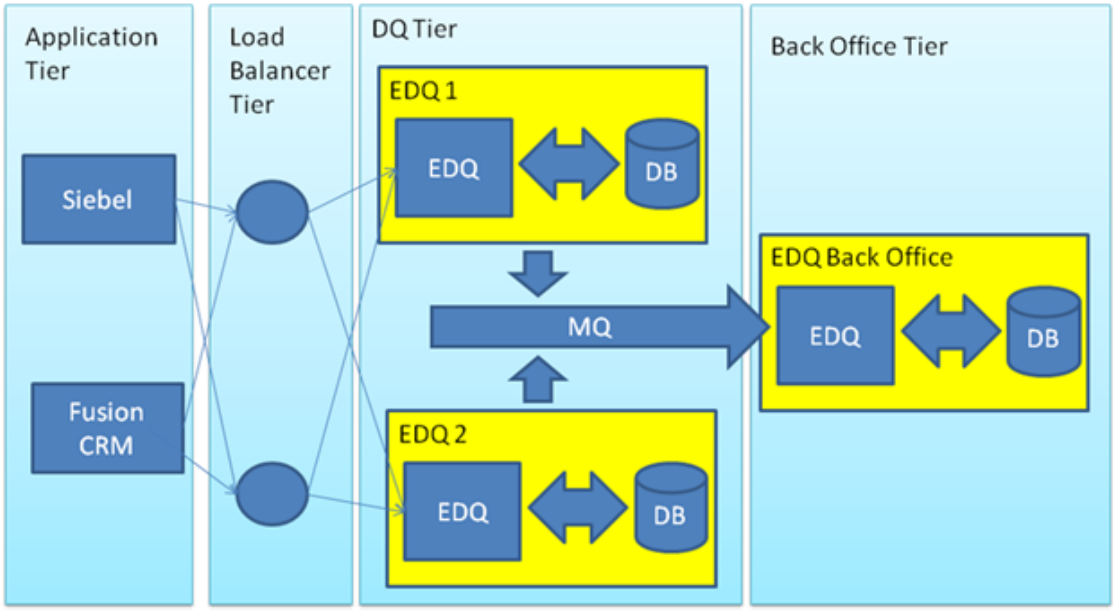

Figure 2: Multi-tier architecture where EDQ needs to manage state (in Case Management)

When deployed for reference data matching where EDQ is also used for Case Management, web service requests are being made of EDQ to identify if customer records match records in reference data (such as watch lists). The response from EDQ can be that the customer record does not match any records, or that there is a potential match that requires investigation. If matches are reviewed or investigated externally, match information is returned to the calling application, so EDQ is stateless and can be deployed using the same mechanisms as Real time Customer Data Services. Alternatively, if EDQ Case Management is used, the case that requires investigation needs to be investigated using a Case Management server that is separate from the 'screening' servers in the DQ Tier. In this case state must be maintained as it is updated by users. When deployed in this situation, the records that are potential matches are dispatched to a further EDQ instance where Case Management will be performed. Normally a message queue is used for this dispatch, though some customers have used file handovers.

Since Case Management is a back office function the system can be run with a warm or cold fail over, accepting a small amount of down time in the event of a failure. The front end of this EDQ deployed system is stateless so can have multiple independent servers with their own databases. This is convenient when the watch list needs to be refreshed since each server can be taken down in turn and have the watch list reloaded. Although this means that each server could produce slightly different answers depending on whether or not the watch list has been reloaded, in practice this is not normally an issue.

2.3 Batch Customer Data Services

When running customer data services in batch mode, an EDQ job is initiated which loads the customer data into the EDQ system. EDQ then processes the data and returns the processed data to the calling system. Batch processing in EDQ is not restartable so in the event of a failure the entire job must be restarted.

Integrations been source systems and EDQ tend to vary with either enterprise scheduling system delivering data via file transfer and initiating jobs via a command line. It is possible to use a message queue to connect the calling server with a pool of batch processing servers, any of which can process the necessary data and return it to the caller.
In both of the above cases the EDQ server is effectively stateless. So in the event of failure the job can either be resubmitted to one of a pool of similarly configured machines but independent machines, or in the case of submitting jobs via a message queue the message would be automatically resubmitted to another machine that is listening for work from the queue.

2.4 Batch Reference Data Matching

Batch reference data matching jobs are very similar to batch Customer Data Services jobs, except that cases may be generated that require investigation. Where EDQ is used for Case Management, this means EDQ manages state.

Customers approach this in the same way as they have approached real-time reference data matching. The case state is propagated from the batch screening machines to a centralized EDQ system that manages cases. This means the batch processing servers are effectively stateless and the state is held in a separate back office tier for managing cases. Again because this is back office functionality a warm or cold failover machine provides adequate failover capability.

3. Rules for configuring EDQ for HA

There are 2 fundamental rules which must be adhered to when configuring EDQ for High Availability, in all deployment models:

- Each EDQ server needs its own pair of repository database schemas. One (default name 'director') stores configuration information and the other (default name 'results') stores working intermediate tables and indexes.
- Each server needs at least one EDQ config directory for server specific configuration information, including details of database repository connection details. Note that it is possible (but not mandatory) for multiple configuration directories to be used.

In addition, if (and only if) EDQ is deployed with several managed servers on the same machine (for example in a WebLogic Cluster):

- Each managed server must use different ports. Normally FTP and SFTP access is not required, but HTTP, HTTPS and management ports may be. The HTTP and HTTPS ports are configured for each managed server using the WebLogic console. The management port is configured using the management.port property in the director.properties file for each server.
- Each server must reference its configuration directory using the EDQ_CONFIG_PATH variable, or (from EDQ 11g) using a server startup argument (-Dedq.config.path=).

4. Web Service (Real-time) HA

4.1 Deployment

Real-time HA configuration requires a load balancer to manage and distribute requests amongst a number of EDQ servers.

An example deployment using multiple managed servers in a WebLogic Cluster is shown below:

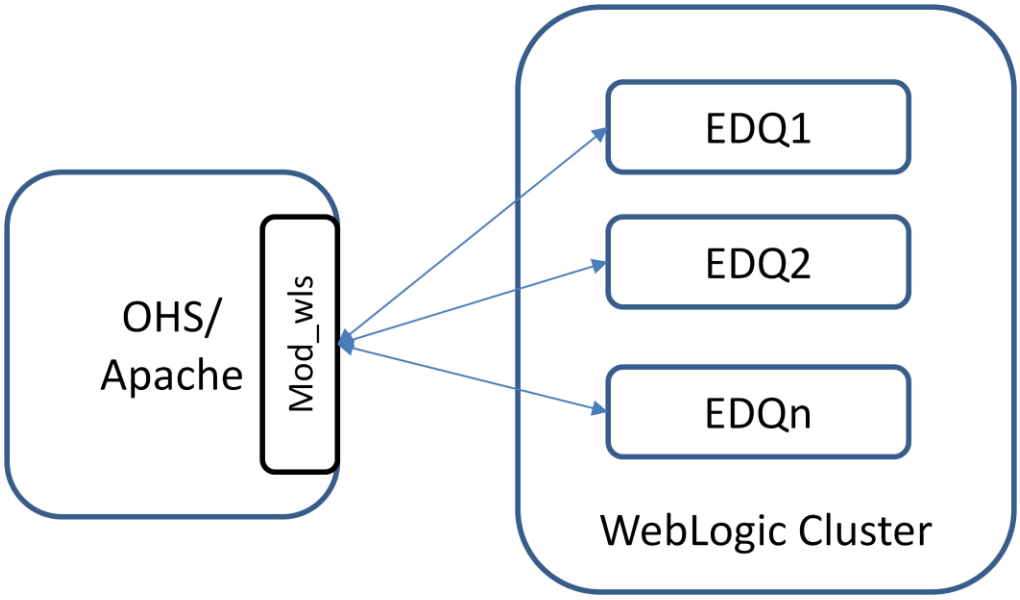

Figure 3 - EDQ in a WebLogic Cluster

An example deployment using completely separate installations of EDQ (on any platform) is shown below:

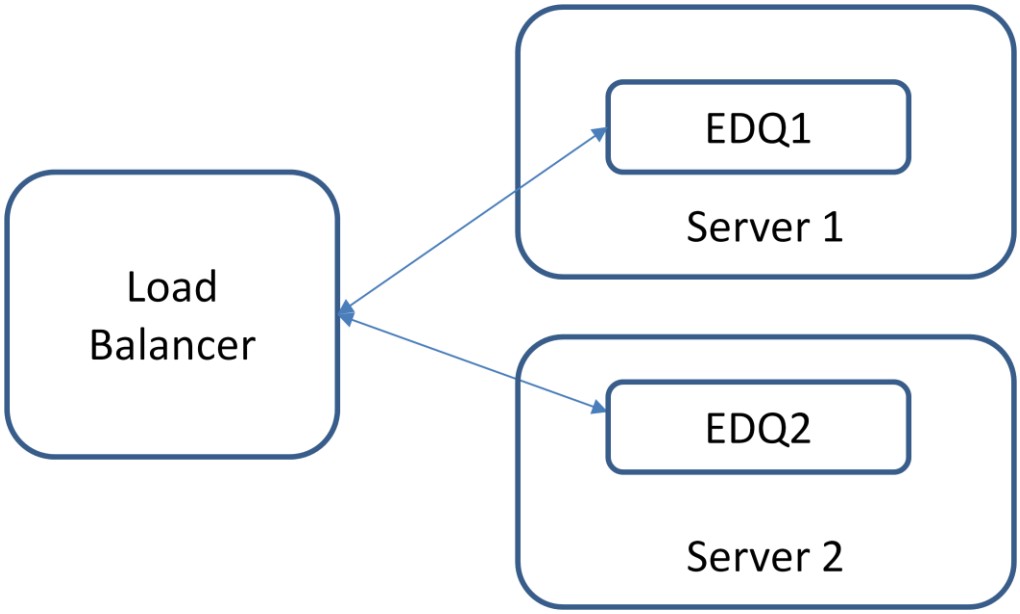

Figure 4 - Multiple EDQ instances

4.2 OHS configuration

OHS (or Apache) may be used to provide a front end to multiple EDQ servers in order to balance web service requests between them. It is configured with the mod_wls WebLogic plug-in, a simple load balancer which distributes connections (web service requests) across a number of EDQ servers, and which can detect if each server is running (whether it is in a cluster or not).

The same basic process works with other load balancing technologies.

Once set up, any external applications that need to send web service requests to EDQ should be configured to use the URLs exposed by the load balancer rather than server-specific URLs. For example, if integrating EDQ with Siebel, update the dnd.properties file for the EDQ Siebel Connector to point to the load-balanced URL rather than a server-specific URL.

There is no need to use HTTPS but if you do so this will require trusted connections.

4.3 EDQ Managed Servers in a WebLogic Cluster

There are advantages if the EDQ servers are in a WebLogic cluster as the WLS cluster is automatically "aware" of the status (running or not running) of each of the servers and updates the mod_wls plug-in. In addition, you can dynamically add servers to the cluster and they will be added to the plug-in's table of available servers automatically. WebLogic should be configured with an Admin server managing a number of separate EDQ servers installed in individual Managed Servers. For installation instructions, see Section 6.

4.4 EDQ User Interfaces

In general, an HA deployment of EDQ is intended as a black box solution for the purpose of load balancing web service requests between EDQ servers with the same configuration. It may therefore be valid not to allow users to access any of the EDQ UIs on the production machines at all.

However, EDQ UIs such as Server Console can be used to monitor activity on servers, and for manual interactions such as scheduling jobs to start at startup time. Importantly, the EDQ UIs should not be accessed through a load balancer, as even if this can be made work, users will simply be non-deterministically logging in to one of the logically separate servers. Any configuration changes will not be propagated to the other servers, and activity on the system will be confined to the work allocated to that server.

If there is a requirement to access the EDQ client applications, this must be done by launching a client from one of the servers, and connecting to the other.

Note that:

- The Director and Server Console client applications support simultaneous connections to multiple EDQ servers, and will save connection details to these multiple servers once configured. They also both support disconnection to servers; the client can continue to run disconnected from one or both servers.
- Other UIs (Match Review, Case Management, Dashboard etc.) are single-server only.
- It is not normally recommended to update configuration on production HA servers manually via Director. Updates to configuration can be done using the Subversion integration. If configuration updates are done manually, this should be by promoting configuration from a test system using a DXI file onto all servers, and done in a period of downtime on each server.

4.5 Checking that EDQ web services are available

OHS and other load balancers detect whether or not a server is running. They cannot, however, easily detect if a particular EDQ web service is 'live' in a managed server when that service depends on a given job running (and being ready to service requests) in EDQ.

In the main, the best approach to this is to ensure that the web services are always running when the application server is running. This can be achieved by:

1. Scheduling the real-time services to run at application server start-up. Note: This can be done from the EDQ Server Console UI, or using the command line interface.

2. Exiting the application server in the event of an error in an EDQ job.

For more advanced monitoring, it is also possible to monitor the status of a given web service using EDQ's JMX interface.

For each web service that you need to monitor, connect to the EDQ management port on a given EDQ server (this is defined using the management.port line in director.properties, or if undefined, the port defaults to 8090) and using the path as shown below, check the 'open' attribute of the service. It will be 1 if the service is running, or 0 if it is not:

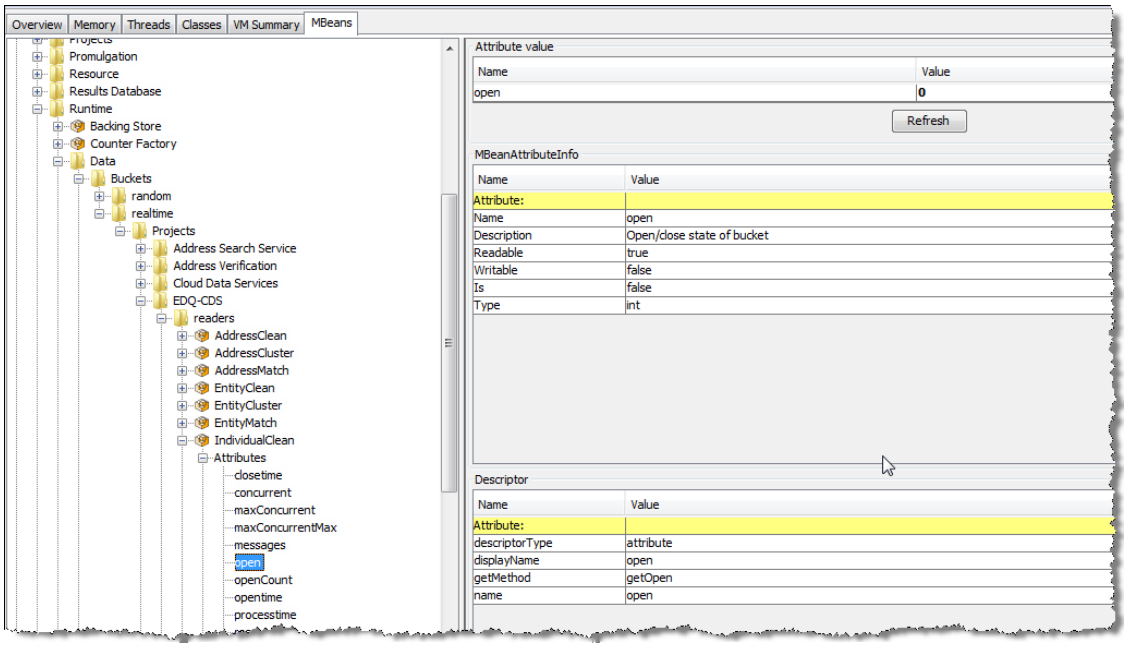

Figure 5: EDQ MBeans for Web Service monitoring using the JMX interface

Depending on your load balancing technology, it may therefore be possible to use a script that monitors specific web service availability and stops routing requests if a web service is not available on one of the servers.

5. High Availability of EDQ Batch Jobs

Please note that EDQ does not support HA of batch jobs when attached to Siebel. This is because:

- The EDQ Siebel Connector only supports connection to a single EDQ server's JMX interface for the purposes of running batch jobs, and has no concept of retrying a connection to another server if unavailable
- There are few requirements for HA of batch services attached to Siebel since the 'UCM Batch' workflow uses EDQ web services, and the batch DQ interface with Siebel does not invoke survivorship of matches, meaning it is rarely used. Normally a batch run can simply be retried once the connected EDQ server is available again.

The remainder of this section is therefore for other use cases where stateless batch jobs are available on a number of EDQ instances. The configuration below will be used by the integration of EDQ with the Oracle Sales Cloud.

5.1 Configuration

A shared staging database is required. This is shared between the service consumers and the EDQ servers and is used to hand over data for EDQ to process, and for the external application to pick up processed data. Note that this is NOT the same as the EDQ repository database, though if a large Oracle RAC database is available, it can be used for both this purpose and to provide the repository schemas for the EDQ servers.

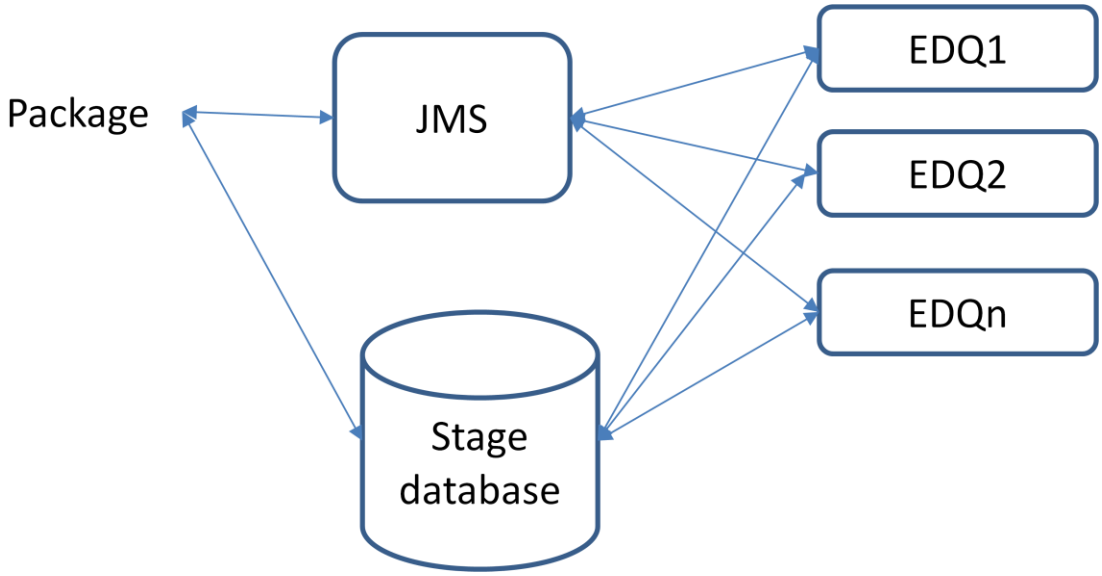

Figure 6: High Availability of stateless EDQ Batch Jobs

5.2 JMS

A JMS queue is required. The consuming services place requests for batch job processing onto this queue. The requests are consumed and actioned by individual EDQ servers which report status and completion or failure information back via a JMS topic. The EDQ servers take packages off the JMS queue when they have capacity.

A JMS queue is used for the packages from the requesting services. There is an agreed format between the two ends. Examples can be seen in the EDQ-CDS 11.1.1.7.3 package.

JMS Topics are used to broadcast job status information.

There are multiple EDQ servers. These do not need to be in a cluster, but must be configured to consume the JMS queue and use the same batch job configuration.

5.3 Triggers

The actions performed for each packaged are based on EDQ triggers. EDQ triggers are Groovy scripts which access the EDQ API to run jobs. Examples can be seen in the EDQ-CDS 11.1.1.7.3 package.

Return status and job completion information is actioned via a periodic trigger which updates status and confirms end of job and any details about the return status.

6. Installing EDQ to operate in a WebLogic Cluster

These instructions are generally intended to apply to EDQ version 9.0.x. From EDQ 11g, the majority of the configuration is achieved using the Configuration Wizard for WebLogic that is included in the EDQ product, and specific notes are included for deploying in a cluster.

6.1 Installation

Follow the standard installation instructions for installing each managed server on WebLogic, with the following notes and modifications:

- Use a single WebLogic domain with a single Admin server and several managed servers in a cluster.
- Use different HTTP and HTTPS ports for each EDQ managed server
- Deploy EDQ to the domain
- Set up multiple config directories and configure each managed server's server startup arguments to use these using the instructions in Section 6.2 below.
- Set up the required JNDI connections to each repository database schema in WebLogic console
- Edit the director.properties file in each server's configuration directory to use the allocated pair of JNDI connections for each server using the instructions below
- Start each managed server in turn

Note that the above assumes that the reader has knowledge of how to apply High Availability strategies with WebLogic and therefore understands how to replicate the whole domain on another physical machine for redundancy purposes.

6.2 Configuring multiple EDQ configuration directories

Each EDQ server must have its own configuration directory but some parts of the configuration directory can be shared, and referenced by many servers. These parts must be read-only.

This document recommends setting up the following directories:

1. **base**, to contain standard information that is never changed after installation (read only)

2. **shared**, to contain implementation-specific, but not server-specific information, such as LDAP configuration, which is the same across all servers (read only), and

3. server-specific configuration directories (e.g. **edqserver1config**, **edqserver2config**), to contain server specific files and directories such as:

   - director.properties (which specifies each server's connection to repository database schemas)
   - log files (the logs subdirectory)
   - the file landing area (the landingarea subdirectory).

These instructions are one way to perform this task for 3 directories. Either use this mechanism or modify to your own requirements.

1. Create an 'edqconfig' directory and navigate to it

2. Create the following subdirectories of 'edqconfig':

   - 'base'
   - 'shared'
   - 'edqserver1config', 'edqserver2config' etc. (one for each server)

3. Extract 'config.zip' from the EDQ zip distribution into 'base' and set as read-only.

4. Move any directories and files for shared configuration from base into 'shared'. A common example is the 'login.properties' file in the 'security' directory for LDAP configuration which is shared across all servers.

5. Set 'shared' as read-only.

6. Move server-specific directories such as logs and landingarea from base into edqserver1config, edqserver2config etc.

7. Move the server-specific director.properties file from base into edqserver1config, edqserver2config etc. This file must be edited to connect to the repository schemas for each server (see next section)

8. [For EDQ 9.0.x] Set the EDQ_CONFIG_PATH environment variable in setDomainEnv.sh using a script that sets the variable differently depending on the managed server. For example:

```
if [ "$SERVER_NAME" = "edq_server1" -o "$SERVER_NAME" =
"edq_server2" -o ... ]
then USER_MEM_ARGS="-Xmx1536m -XX:MaxPermSize=512m"
     EDQ_CONFIG_PATH=/opt/edqconfig/base:/opt/edqconfig/shared

     if [ "$SERVER_NAME" = "edq_server1" ]
     then
EDQ_CONFIG_PATH=$EDQ_CONFIG_PATH:/opt/edqconfig/edqserver1config
     elif [ "$SERVER_NAME" = "edq_server2" ]
     then
EDQ_CONFIG_PATH=$EDQ_CONFIG_PATH:/opt/edqconfig/edqserver2config
     elif ...
     fi

     export EDQ_CONFIG_PATH
fi
```

Note: For EDQ versions from EDQ 11.1.1.7.3, the configuration path can be set using a server startup argument. This is configured automatically using the EDQ WebLogic Configuration Wizard but can also be specified manually if a specific hierarchy of configuration directories is required. Server startup arguments are configured on the Server Start tab of a managed server in the WebLogic admin console:

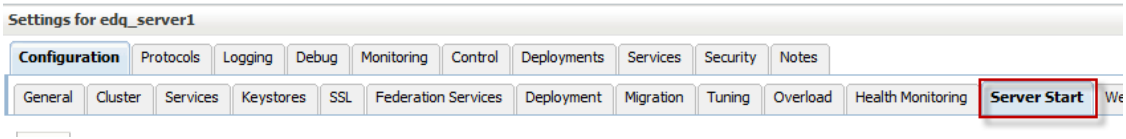

Figure 7: Server Start tab when configuring a server in WebLogic Admin Console

The arguments might be set as follows:

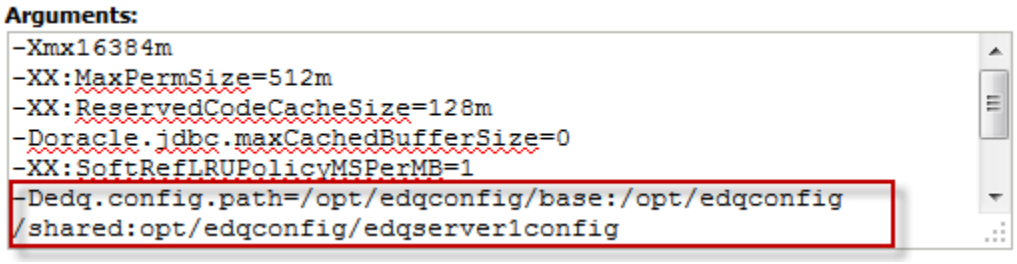

Figure 8: Example startup arguments for an EDQ managed server (assumes EDQ 11g or later)

(Note that the first two JVM settings above such as –Xmx16384m to set a maximum heap of 16GB are only suitable for large servers. Use the EDQ documentation to set these correctly for your environment.)

6.3 RAC configuration and JNDI connections

For all HA deployments of EDQ, it is recommended that a RAC-enabled Oracle database is used to host all repository schemas. Each EDQ server will have a minimum of 2 schemas on the RAC database; one for 'director' and one for 'results'. So the first EDQ instance will have e.g. 'director1' and 'results1' and the second 'director2' and 'results2' and so on.

Connection to the repository schemas should be via JNDI data sources defined and managed in the WebLogic Admin Server and configured in the director.properties file in the server-specific configuration directory.

Note that the installation process for EDQ 11g and later on WebLogic includes the ability to set up this configuration using the Configuration Wizard. For EDQ 9.0 and earlier, these must be configured manually in director.properties.

The following explains the required syntax for such connections:

```
dataSource.jndiname         = jdbc/[name of JNDI
connection to EDQ server 1 configuration schema]
resultsDataSource.jndiname = jdbc/[name of JNDI
connection to EDQ server 1 results schema]
```

For example, an EDQ managed server may have the following director.properties file:

```
#EDQ database connection configuration file, EDQ
Server 1
licence.file=director.lic
dataSource.jndiname = jdbc/edqserver1director
resultsDataSource.jndiname = jdbc/edqserver1results
```

The second server would have a file with different repository DB connections, such as:

```
#EDQ database connection configuration file, EDQ
Server 2
licence.file=director.lic
dataSource.jndiname = jdbc/edqserver2director
resultsDataSource.jndiname = jdbc/edqserver2results
```

7. Installing EDQ for HA using separate installations

If WebLogic clustering is not used, for example because EDQ has been deployed on Tomcat or WebSphere, it is still possible to load balance web service requests between multiple, logically separate, EDQ servers with the same configuration.

7.1 Installation

Follow the standard installation instructions for installing each EDQ server, with the following notes:

1. Use the same deployment context path and ports on all servers, so that HTTP connections are all in the same format, for example:
   - http://server1:8001/edq
   - http://server2:8001/edq

   Note: If you use the Windows Installer with the standard options, an HTTP port of 9002 will be used, so the paths will be:
   - http://server1:9002/dndirector
   - http://server2:9002/dndirector
2. After installation, modify the repository database connections to use Oracle RAC. Note that if you are using EDQ 9.0 or earlier, RAC connection strings cannot be specified using the EDQ database configuration page, so director.properties must be edited manually. Use JNDI connection strings if possible (see RAC configuration and JNDI connections section above). If JNDI is not an option (for example because EDQ is deployed on Tomcat), you can edit director.properties to point at the allocated schemas for the server directly, using standard Oracle JDBC URLs. Below is an example of the syntax for the connection strings. These will need to be modified to the RAC connection strings for your database:

```
dataSource.url=jdbc:oracle:thin:@(DESCRIPTION = \
    (ADDRESS = (PROTOCOL = TCP)(HOST =
servername)(PORT = 1521)) \
    (CONNECT_DATA = \
      (SERVER = DEDICATED) \
      (SERVICE_NAME = orcl) \
    ) \
  )
resultsDataSource.url=jdbc:oracle:thin:@(DESCRIPTION =
\
    (ADDRESS = (PROTOCOL = TCP)(HOST =
servername)(PORT = 1521)) \
    (CONNECT_DATA = \
```

```
      (SERVER = DEDICATED) \
      (SERVICE_NAME = orcl) \
    ) \
)
```

3. Install any required extensions and configuration – for example the EDQ Customer Data Services Pack – onto each of the servers

4. Update your load balancer to load balance requests between the servers, and your calling application to use the web service URLs exposed by the load balancer.

EDQ High Availability Guide
November 2013
Author: Mike Matthews
Contributing Authors: Gerry Kelley, Richard Evans

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together