QuartzDesk Web Application Installation and Upgrade Guide for Oracle WebLogic AS 11g (10.3.x) and 12c (12.x)

QuartzDesk Version: 3.x

January 21, 2019

Table of Contents

1. PURPOSE 4

2. DEFINITIONS 5

3. REQUIREMENTS 6

3.1 SOFTWARE REQUIREMENTS 6
3.1.1 Browser 6
3.1.2 Operating System 6
3.1.3 Java 6
3.1.4 Application Server 6
3.1.5 Database 6
3.1.6 Database JDBC Driver 7
3.1.7 QuartzDesk Web Application Archive 7
3.2 HARDWARE REQUIREMENTS 7

4. INSTALLATION 8

4.1 DATABASE 8
4.2 JDBC DRIVER 8
4.2.1 Windows 8
4.2.2 Unix/Linux 9
4.3 JDBC DATA SOURCE 9
4.3.1 DB2 9
4.3.2 H2 10
4.3.3 Microsoft SQL Server 11
4.3.4 MySQL 12
4.3.5 Oracle 13
4.3.6 PostgreSQL 13
4.4 JDBC DATA SOURCE CONNECTION POOL PARAMETERS 14
4.5 TEST JDBC DATA SOURCE 15
4.6 APPLICATION WORK DIRECTORY 15
4.7 APPLICATION CONFIGURATION 17
4.8 DEPLOY APPLICATION 17
4.9 START APPLICATION 19

5. UPGRADING 22

5.1 STOP EXISTING APPLICATION 22
5.2 BACKUP 22
5.3 REMOVE EXISTING APPLICATION 22
5.4 DEPLOY NEW APPLICATION 23
5.5 START NEW APPLICATION 23

6. QUARTZDESK 2.X TO 3.X MIGRATION NOTES 24

6.1 MINIMUM REQUIRED JAVA VERSION 24
6.2 RENAME CONFIGURATION FILE 24
6.3 RENAME LOG FILES 24
6.4 ACCESS TO MONITORING URLS (REST API) 25
6.5 ACCESS TO JAX-WS ENDPOINTS 26
6.6 REMOVE UNUSED GROUPS 27

7. CLUSTER DEPLOYMENT NOTES 28

7.1 HTTP SESSION REPLICATION AND AFFINITY 28
7.2 SHARED WORK DIRECTORY 28
7.3 LOGGING CONFIGURATION 28
7.3.1 Using Shared Log Files 29

7.3.2 Using Separate Log Files *30*
7.4 INTERNAL QUARTZ SCHEDULER 31

1. Purpose

This document describes the installation and upgrade process for the QuartzDesk web application 3.x on Oracle WebLogic Application Server 11g (10.3.x), 12c (12.1.x) and 12cR2 (12.2.x).

If you experience any problems installing or upgrading the QuartzDesk web application, please let us know at support@quartzdesk.com.

2. Definitions

The following table lists all acronyms and shortcuts used throughout this document.

Acronym / Shortcut	Definition
AS	Application Server.
EAR	Enterprise Application Archive. A file with `.ear` extension.
JAR	Java Application Archive. A file with `.jar` extension.
JVM	Java Virtual Machine.
WLAC	WebLogic Administrative Console.
WLAS	WebLogic Application Server.
WAR	Web Application Archive. A file with `.war` extension.

The following table lists all locations and properties used throughout this document.

Location / Property	Example	Description
DB_HOST	localhost	QuartzDesk web application database server host.
DB_PORT	5432	QuartzDesk web application database server port.
DB_NAME	quartzdesk	QuartzDesk web application database name.
DB_SCHEMA	quartzdesk	QuartzDesk web application database schema.
DB_USER	quartzdesk	QuartzDesk web application database user.
DB_PASSWORD	quartzdesk	QuartzDesk web application database user password.
JAVA_HOME	/usr/local/java	Java home directory.
MW_HOME	/opt/oracle/middleware	Oracle Middleware installation directory.
WL_DOMAIN	domain1	WebLogic Application Server domain.
WL_DOMAIN_HOME	/opt/oracle/user_projects/domain1	WebLogic Application Server domain directory.
WL_HTTP_HOST	localhost	WebLogic HTTP listener host.
WL_HTTP_PORT	7001	WebLogic HTTP listener port.
WL_SERVER	MyServer	WebLogic Application Server name.
WORK_DIR	/var/quartzdesk-web.work	QuartzDesk web application work directory.

3. Requirements

3.1 Software Requirements

3.1.1 Browser

The QuartzDesk web application GUI requires a modern JavaScript-enabled browser. Please make sure JavaScript is enabled and not blocked by third party anti-virus/anti-malware software.

The QuartzDesk web application has been tested with the following browser versions. These are also the minimum browsers versions required.

Browser	Minimum Version
Chrome	17
FireFox	10
Internet Explorer	8
Opera	12
Safari	6

3.1.2 Operating System

Windows XP, Windows Vista, Windows 7, Windows 8, Windows 10.
Linux (any distribution) with kernel 2.6.x and above.
Solaris 11.x and above.

3.1.3 Java

Sun/Oracle Java (JDK) 7, 8, 9, 10.
IBM Java (JDK) 7, 8, 9.
OpenJDK 7, 8, 9, 10.

3.1.4 Application Server

Oracle WebLogic Application Server 11g (10.3.x).
Oracle WebLogic Application Server 12cR1 (12.1.x).
Oracle WebLogic Application Server 12cR2 (12.2.x).

3.1.5 Database

Database	Minimum Version
DB2	10.1
H2	1.3.174
Microsoft SQL Server	2008 R2 SP1
MySQL	5.6.4
Oracle	10.2 (10g R2)
PostgreSQL	9.1

3.1.6 Database JDBC Driver

Database	JDBC Driver
DB2	IBM DB2 JDBC 4.0 driver available at http://www-01.ibm.com/support/docview.wss?uid=swg21363866.
H2	Database engine including the JDBC driver is available at http://www.h2database.com.
Microsoft SQL Server	Microsoft JDBC driver 4.0 for SQL Server available at http://msdn.microsoft.com/en-us/sqlserver/aa937724.aspx. We strongly advise against using the alternative JTDS JDBC driver because it does not support the datetime2 data type at this time. As a result, all datetime values written by the QuartzDesk web application would end up rounded up, or down. For datetime data type rounding details, please refer to http://msdn.microsoft.com/en-us/library/ms187819.aspx.
MySQL	Connector/J JDBC driver available at http://dev.mysql.com/downloads/connector/j/.
Oracle	Oracle JDBC driver available at http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html. For a comprehensive overview of JDBC driver versions vs. supported database versions, please refer to http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-faq-090281.html#02_02.
PostgreSQL	JDBC4 PostgreSQL driver available at http://jdbc.postgresql.org/.

3.1.7 QuartzDesk Web Application Archive

To install QuartzDesk web application, you need to obtain the quartzdesk-web-x.y.z.war file. The latest version can be downloaded at www.quartzdesk.com (click Downloads → Latest Release → View files → quartzdesk-web-x.y.z.war).

3.2 Hardware Requirements

QuartzDesk web application runs on any physical or virtualized hardware that supports the above software requirements.

4. Installation

This chapter describes the standard QuartzDesk installation. If you are only evaluating QuartzDesk, you may be interested in the **one-step installation mode** to dramatically reduce the number of required installation steps. For details, please refer to our FAQs (search for "one-step installation").

4.1 Database

Create a new database user named `quartzdesk` (DB_USER) with an arbitrary password (DB_PASSWORD).

Create a new QuartzDesk web application database named `quartzdesk`[1] (DB_NAME) owned by DB_USER.

In the `quartzdesk` database create a new schema named `quartzdesk` (DB_SCHEMA). The schema must be owned by DB_USER. Make the created DB_SCHEMA the default schema of DB_USER and/or add the schema to the DB_USER's schema search path.

Please contact your DBA, or refer to the database engine documentation for instructions on how to complete the above database-specific tasks.

Please note that you do not have to create any database objects (tables, keys, indices etc.) in the `quartzdesk` database / schema. These objects will be automatically created by the QuartzDesk web application during its first start.

4.2 JDBC Driver

Download and install the JDBC driver for the created database. For a list of supported JDBC drivers please refer to chapter 3.1.6.

Third-party JDBC driver files must be added to the WLAS classpath. To add the JDBC driver files to the WLAS classpath, please follow these steps:

4.2.1 Windows

Edit `MW_HOME/wlserver/common/bin/commEnv.cmd` (or `MW_HOME/oracle_common/common/bin/commEnv.sh` on some platforms) and add the following lines at the end of the file:

```
rem
rem JDBC driver used by the QuartzDesk Web Application.
rem
set WEBLOGIC_CLASSPATH=<JDBC_DRIVER_HOME>\<jdbc-driver-jar>;%WEBLOGIC_CLASSPATH%
```

[1] DB2 restricts the database name length to the maximum of 8 characters. Please adjust the database name accordingly (e.g. qdesk).

Where <JDBC_DRIVER_HOME> is the installation directory of the JDBC driver and <jdbc-driver-jar> is the JDBC driver JAR file. If the JDBC driver requires multiple JAR files, add these JAR files to the WEBLOGIC_CLASSPATH as well.

Make sure the JDBC driver JAR files are readable by the user the WLAS process is started under.

4.2.2 Unix/Linux

Edit MW_HOME/wlserver/common/bin/commEnv.sh (or MW_HOME/oracle_common/common/bin/commEnv.sh on some platforms) and add the following lines at the end of the file:

```
#
# JDBC driver used by the QuartzDesk Web Application.
#
WEBLOGIC_CLASSPATH="<JDBC_DRIVER_HOME>/<jdbc-driver-jar>:${WEBLOGIC_CLASSPATH}"
export WEBLOGIC_CLASSPATH
```

Where <JDBC_DRIVER_HOME> is the installation directory of the JDBC driver and <jdbc-driver-jar> is the JDBC driver JAR file. If the JDBC driver requires multiple JAR files, add these JAR files to the WEBLOGIC_CLASSPATH as well.

Make sure the JDBC driver JAR files are readable by the user the WLAS process is started under.

4.3 JDBC Data Source

In WLAC (WL_DOMAIN → Services → Data Sources) create a new Generic Data Source (New → Generic Data Source) for the QuartzDesk database.

The following steps depend on the QuartzDesk database type and are described in the following sub-chapters.

4.3.1 DB2

In Step 1, enter the following values:

Name: QuartzDeskDS
JNDI Name: jdbc/QuartzDeskDS
Database Type: DB2

In Step 2, select the JDBC driver:

Database Driver: IBM's DB2 Driver (Type4) for JDBC and SQLJ; Versions: 8.X and later

Click Next.

In Step 3:

Supports Global Transactions: uncheck

Click Next.

In Step 4, enter DB connection parameters:

Database Name: DB_NAME
Host Name: DB_HOST
Port: DB_PORT
Database User Name: DB_USER
Password: DB_PASSWORD
Confirm Password: DB_PASWORD

Click Next.

In Step 5, confirm the JDBC driver class name, URL and other data source parameters:

Driver Class Name: com.ibm.db2.jcc.DB2Driver
URL: jdbc:db2://DB_HOST:DB_PORT/DB_NAME
Test Table Name: SQL select 1 from sysibm.sysdummy1

Click Next.

In Step 6, map the created data source to the desired WLAS targets.

Click Finish.

4.3.2 H2

We recommend using H2 for evaluation and/or experimental purposes only. We strongly discourage using H2 in production environments.

In Step 1, enter the following values:

Name: QuartzDeskDS
JNDI Name: jdbc/QuartzDeskDS
Database Type: Other

In Step 2, select the JDBC driver:

Database Driver: Other

Click Next.

In Step 3:

Supports Global Transactions: uncheck

Click Next.

In Step 4, enter DB connection parameters:

Database User Name: DB_USER
Password: DB_PASSWORD
Confirm Password: DB_PASWORD
(Properties: user= DB_USER)

Click Next.

In Step 5, enter the JDBC driver class name, URL and other data source parameters:
Driver Class Name: org.h2.Driver
URL: jdbc:h2:file:<H2_DB_FILE_PATH>
Test Table Name: SQL select 1

Please note that H2 can be configured to run in various operating modes by adjusting the database URL value. For details, please refer to the H2 documentation at http://www.h2database.com/html/features.html#database_url.

Click Next.

In Step 6, map the created data source to the desired WLAS targets.

Click Finish.

4.3.3 Microsoft SQL Server

In Step 1, enter the following values:

Name: QuartzDeskDS
JNDI Name: jdbc/QuartzDeskDS
Database Type: MS SQL Server

In Step 2, select the JDBC driver:

Database Driver: Microsoft's MS SQL Server Driver (Type 4) Versions:2005 and later

Click Next.

In Step 3:

Supports Global Transactions: uncheck

Click Next.

In Step 4, enter DB connection parameters:

Database Name: DB_NAME
Host Name: DB_HOST
Port: DB_PORT
Database User Name: DB_USER
Password: DB_PASSWORD
Confirm Password: DB_PASWORD

Click Next.

In Step 5, confirm the JDBC driver class name, URL and other data source parameters:

Driver Class Name: com.microsoft.sqlserver.jdbc.SQLServerDriver

URL: jdbc:sqlserver://DB_HOST:DB_PORT
Test Table Name: SQL select 1

Click Next.

In Step 6, map the created data source to the desired WLAS targets.

Click Finish.

4.3.4 MySQL

In Step 1, enter the following values:

Name: QuartzDeskDS
JNDI Name: jdbc/QuartzDeskDS
Database Type: MySQL

In Step 2, select the JDBC driver:

Database Driver: MySQL's Driver (Type 4) Versions:using com.mysql.jdbc.Driver

Click Next.

In Step 3:

Supports Global Transactions: uncheck

Click Next.

In Step 4, enter DB connection parameters:

Database Name: DB_NAME
Host Name: DB_HOST
Port: DB_PORT
Database User Name: DB_USER
Password: DB_PASSWORD
Confirm Password: DB_PASWORD

Click Next.

In Step 5, confirm the JDBC driver class name, URL and other data source parameters:

Driver Class Name: com.mysql.jdbc.Driver
URL: jdbc:mysql://DB_HOST:DB_PORT/DB_NAME
Test Table Name: SQL select 1

Click Next.

In Step 6, map the created data source to the desired WLAS targets.

Click Finish.

4.3.5 Oracle

In Step 1, enter the following values:
Name: QuartzDeskDS
JNDI Name: jdbc/QuartzDeskDS
Database Type: Oracle

In Step 2, select the JDBC driver:

Database Driver: *Oracle's Driver (Thin) for Instance connections; Versions:9.0.1 and later

Click Next.

In Step 3:

Supports Global Transactions: uncheck

Click Next.

In Step 4, enter DB connection parameters:

Database Name: DB_NAME
Host Name: DB_HOST
Port: DB_PORT
Database User Name: DB_USER
Password: DB_PASSWORD
Confirm Password: DB_PASWORD

Click Next.

In Step 5, confirm the JDBC driver class name, URL and other data source parameters:

Driver Class Name: oracle.jdbc.OracleDriver
URL: jdbc:oracle:thin:@DB_HOST:DB_PORT/DB_NAME
Test Table Name: SQL select 1 from dual

Click Next.

In Step 6, map the created data source to the desired WLAS targets.

Click Finish.

4.3.6 PostgreSQL

In Step 1, enter the following values:

Name: QuartzDeskDS
JNDI Name: jdbc/QuartzDeskDS
Database Type: PostgreSQL

In Step 2, select the JDBC driver:

Database Driver: PostgreSQL's Driver (Type 4) Versions:Any

Click Next.

In Step 3:
Supports Global Transactions: uncheck

Click Next.

In Step 4, enter DB connection parameters:

Database Name: DB_NAME
Host Name: DB_HOST
Port: DB_PORT
Database User Name: DB_USER
Password: DB_PASSWORD
Confirm Password: DB_PASWORD

Click Next.

In Step 5, confirm the JDBC driver class name, URL and other data source parameters:

Driver Class Name: org.postgresql.Driver
URL: jdbc:postgresql://DB_HOST:DB_PORT/DB_NAME
Test Table Name: SQL select 1

Click Next.

In Step 6, map the created data source to the desired WLAS targets.

Click Finish.

4.4 JDBC Data Source Connection Pool Parameters

In WLAC (WL_DOMAIN → Services → Data Sources) click on the QuartzDeskDS data source. In the Configuration → Connection Pool tab change the following parameters:

Initial Capacity: 2
Maximum Capacity: 10
Statement Cache Size: 100

Expand the Advanced section and change the following parameters:

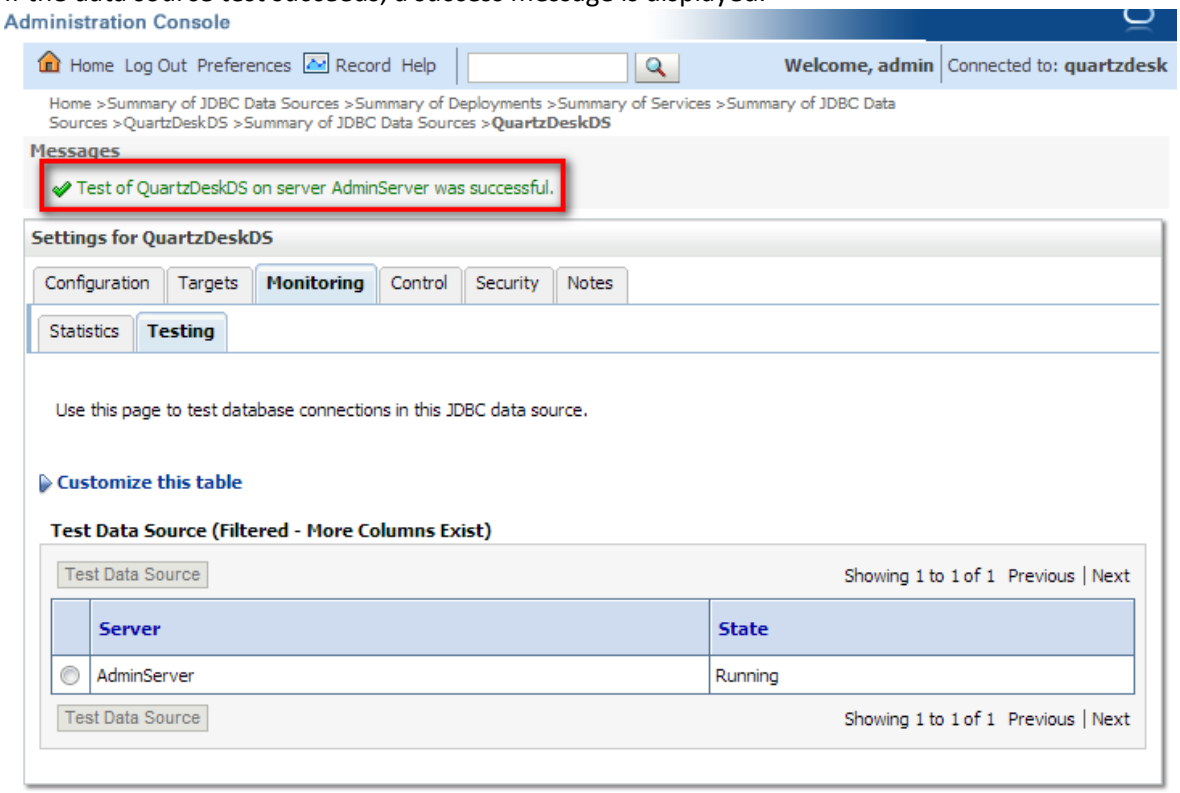
Test Connections on Reserve: check
Connection Reserve Timeout: 5

Click Save.

4.5 Test JDBC Data Source

In WLAC (WL_DOMAIN → Services → Data Sources) click on the QuartzDeskDS data source. In the Monitoring → Testing tab select the WLAS targets to test the data source on. Click the Test Data Source button.

If the data source test succeeds, a success message is displayed.



If the data source connection pool test fails, an error message is displayed and an exception is logged in the application server log (`WL_DOMAIN_HOME/servers/<WL_SERVER>/<WL_SERVER>.log`), where <WL_SERVER> is the name of the WLAS the data source was tested on.

4.6 Application Work Directory

Create a QuartzDesk web application work directory (WORK_DIR) anywhere on the local file system. The directory must be readable and writeable by the user the WLAS process is running under.

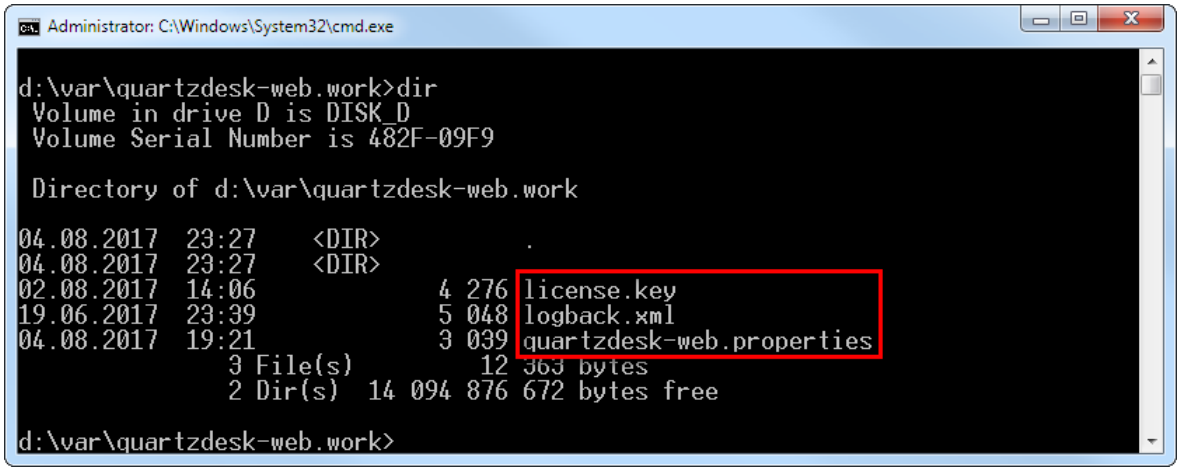
Copy your QuartzDesk license key file (`license.key`) to WORK_DIR.

You can obtain a free 30-day trial license key at www.quartzdesk.com (go to Try / Purchase > Get Trial License Key).

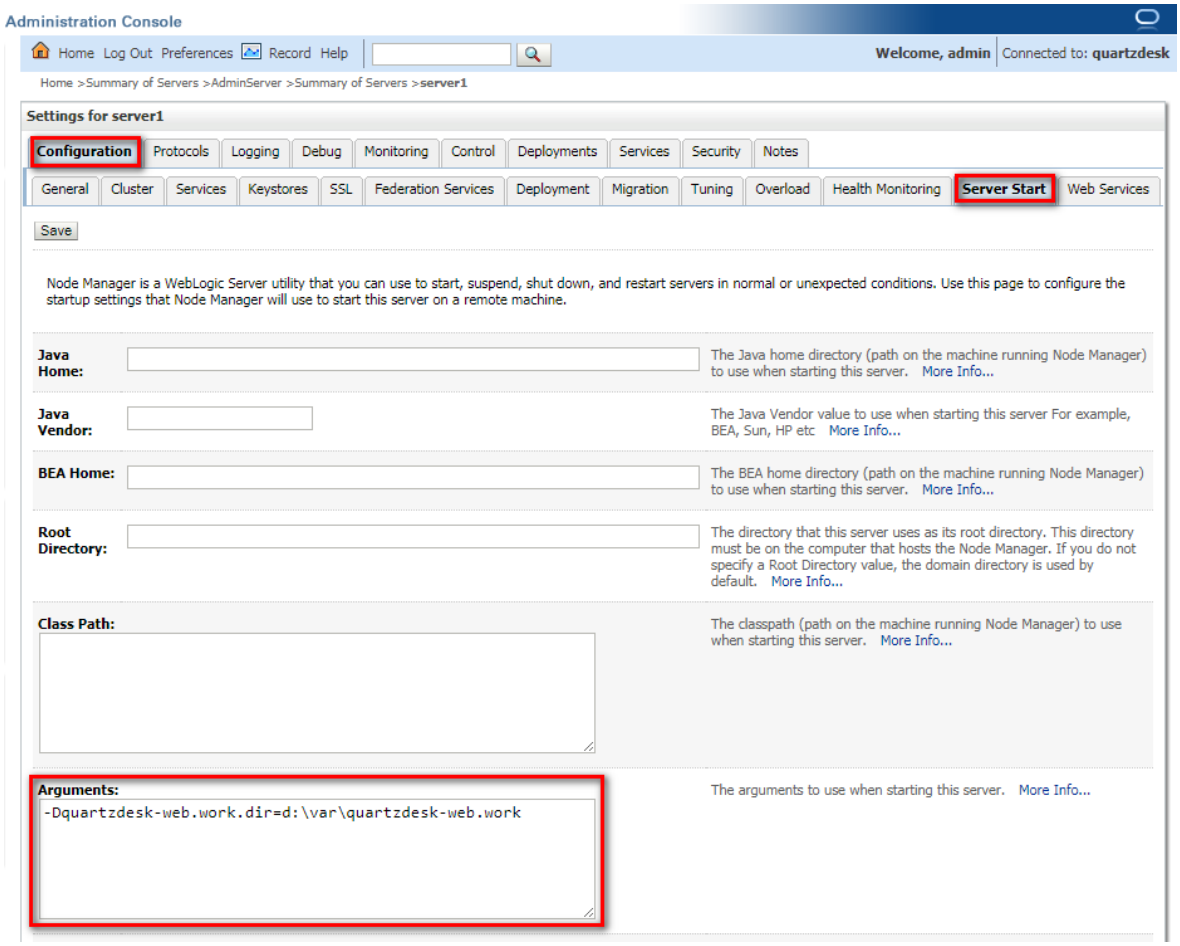
Open the QuartzDesk web application archive (`quartzdesk-web-x.y.z.war`) and copy all files from the `extras/work` directory into WORK_DIR.

If you cannot open the WAR file directly, rename it to *.zip. Do not forget to rename the file back to *.war once you have extracted the required files.

In the following figure you can see an example of a QuartzDesk web application work directory correctly set up on a Microsoft Windows machine.



In WLAC edit server start configuration (WL_DOMAIN → Environment → Servers → WL_SERVER → Configuration → Server Start) and in the Arguments field add a new JVM system property quartzdesk-web.work.dir pointing to the set up WORK_DIR.



Click the Save button.

Restart the updated server (WL_SERVER).

4.7 Application Configuration

Open the QuartzDesk web application configuration file `WORK_DIR/quartzdesk-web.properties`.

Based on the type and version of the database created in step 4.1, change the value of the `db.profile` configuration property according to the following table.

Database	Database Version	db.profile Value
DB2	>= 10.0	db2
H2	>= 1.3.170	h2
Microsoft SQL Server	>= 2008	mssql
MySQL (MyISAM)	>= 5.6	mysql
MySQL (InnoDB)	>= 5.6	mysql_innodb
Oracle	== 8i	oracle8
Oracle	>= 9i	oracle9
PostgreSQL	== 8.1	postgres81
PostgreSQL	>= 8.2	postgres82

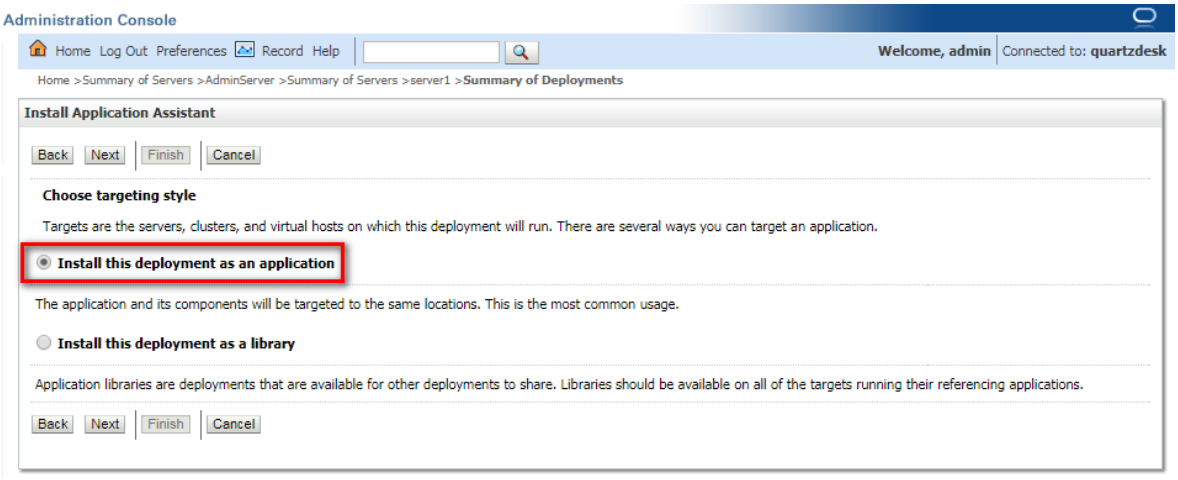
Optionally, you can adjust the QuartzDesk web application logging parameters by editing the `WORK_DIR/logback.xml` configuration file. The default sample `logback.xml` configuration file makes QuartzDesk web application log under the `WORK_DIR/logs` directory that is automatically created when the web application starts. Please refer to the Logback Manual for Logback configuration details.

4.8 Deploy Application

In WLAC go to WL_DOMAIN → Deployments. Click the Install button.

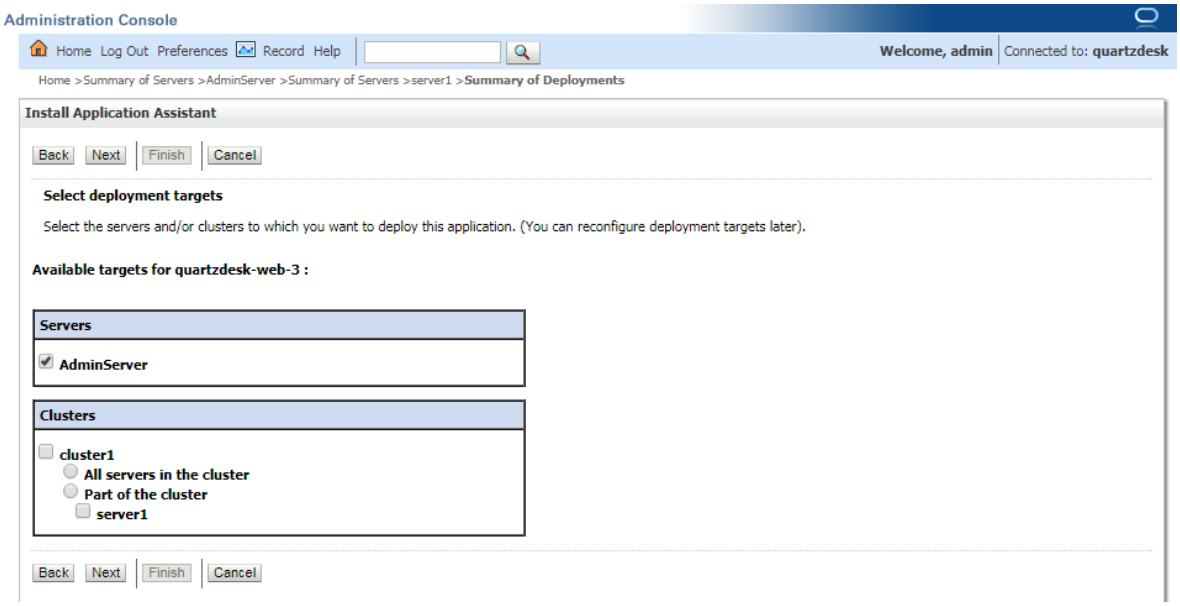
Select or enter the location of the `quartzdesk-web-x.y.z.war` file.
Click Next.

Select the “Install this deployment as an application” option.



Click Next.

On the next screen select the desired deployment targets.

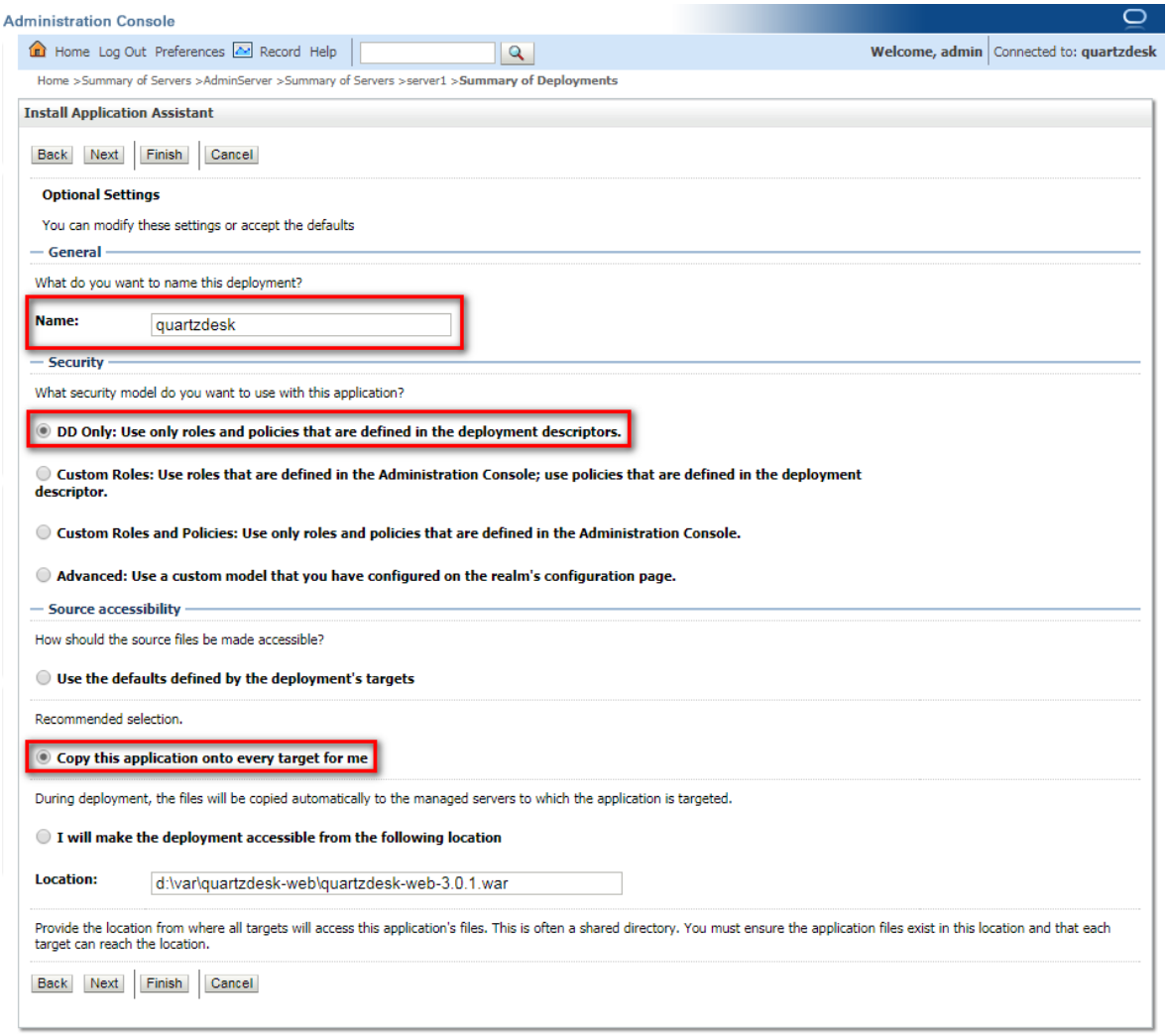


Click Next.

On the next screen set the following application properties:

General / Name: quartzdesk

Security / select “DD Only: Use only roles and policies that are defined in the deployment descriptors.”



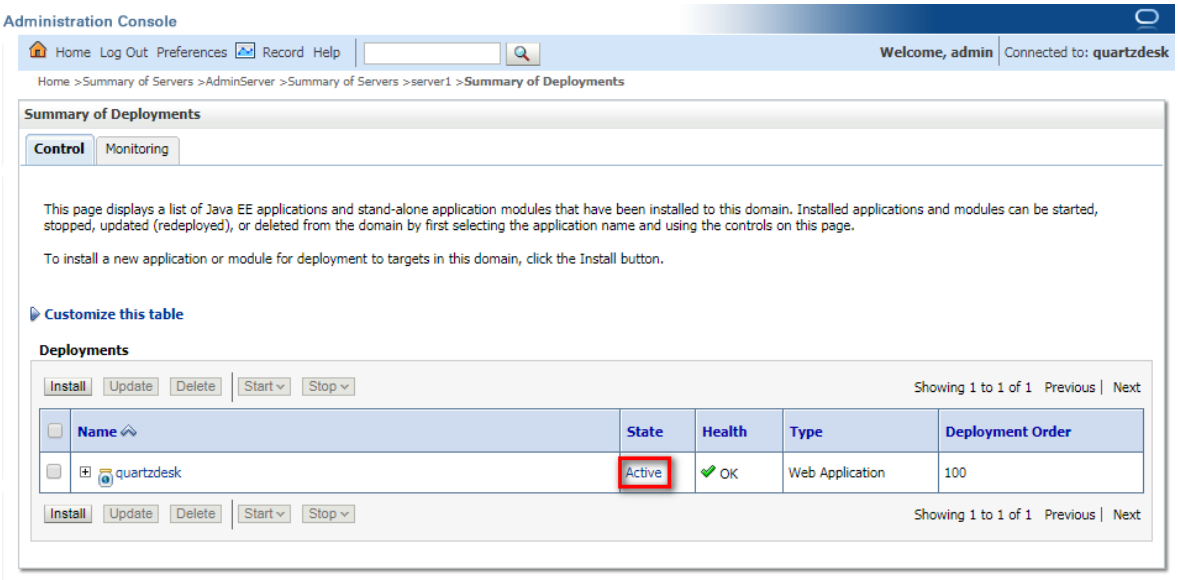
Click Finish.

4.9 Start Application

In WLAS applications are typically started automatically once their deployment process completes.

To start the QuartzDesk web application manually, open WLAC and go to WL_DOMAIN → Deployments. Select the QuartzDesk web application and click Start → Servicing all requests.

Once the QuartzDesk web application starts, its state indicator (under WL_DOMAIN → Deployments) should change to “Active” as shown in the following figure.



Check the WLAS log file `WL_DOMAIN_HOME/servers/<WL_SERVER>/<WL_SERVER>.log` for errors, where <WL_SERVER> is the name of the WLAS the application has been deployed to.

There should be no errors and/or exceptions related to the QuartzDesk web application deployment.

Check the QuartzDesk web application logs (by default located in the `WORK_DIR/logs` directory) for errors.

If there are no errors, point your browser to `http://WL_HTTP_HOST:WL_HTTP_PORT/quartzdesk/` and verify that the QuartzDesk web application GUI is accessible.

Check the version number of the deployed QuartzDesk web application.

To log in, use the default administrator login credentials:

Username: admin
Password: admin123

Once logged in, you can go to Settings > Users to manage users with access to the QuartzDesk web application GUI. Users can be assigned different access permissions based on their intended roles.

In Settings > Groups, you can manage groups and assign access permissions to these groups. A group can contain users (members) who inherit access permissions of the group. A user can be a member of any number of groups.

Effective access permissions of a user are permissions associated directly with the user plus access permissions of all groups the user is a member of.

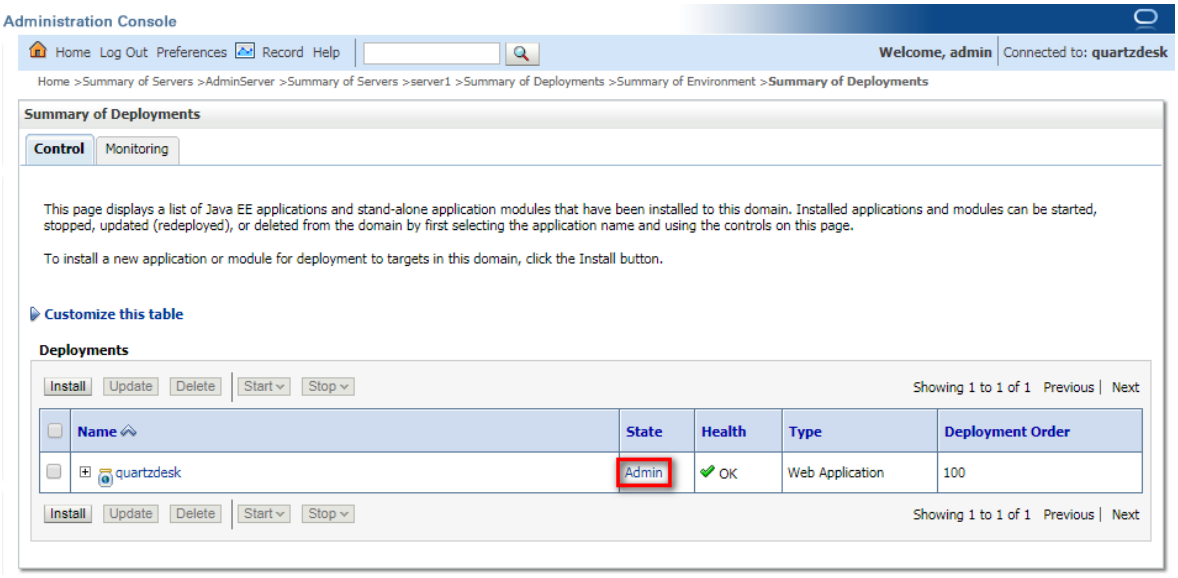
5. Upgrading

5.1 Stop Existing Application

In WLAC go to WL_DOMAIN → Deployments. Select the QuartzDesk web application and click Stop → When work completes. Wait for the action to complete.

If the application cannot be stopped using Stop → When work completes, use Stop → Force stop now.

Upon successful stopping, the state indicator (under WL_DOMAIN → Deployments) should change to "Admin" as shown in the following figure.



5.2 Backup

Backup your QuartzDesk web application database. We recommend performing a **full database backup**.

Backup the contents of the QuartzDesk web application work directory.

Make sure you still have the WAR file of the existing QuartzDesk web application.

Store the backup files in a safe place so that you can restore the original QuartzDesk web application version if the need arises.

5.3 Remove Existing Application

In WLAC go to WL_DOMAIN → Deployments. Select the checkbox next to the existing QuartzDesk web application in the Deployments list. Click the Delete button at the top of the list. Wait for the action to complete.

Upon successful removal, the QuartzDesk web application disappears from the Deployments list.

5.4 Deploy New Application

Deploy the new version of the QuartzDesk web application by following the deployment steps outlined in 4.8.

5.5 Start New Application

Start the new version of the QuartzDesk web application by following the steps outlined in 4.9.

6. QuartzDesk 2.x to 3.x Migration Notes

To upgrade QuartzDesk web application 2.x to 3.x, follow the upgrade steps outlined in 5.

Before deploying the new QuartzDesk web application WAR file (quartzdesk-web-x.y.z.war), as outlined in step 5.4, make sure you have implemented changes described in this chapter.

6.1 Minimum Required Java Version

QuartzDesk web application 3.x requires Java 7 or higher. Java 6 is no longer supported.

Make sure WLAS is configured to use Java 7 or higher.

6.2 Rename Configuration File

The name of the QuartzDesk web application 3.x configuration file has changed from `quartzdesk.properties` to `quartzdesk-web.properties`.

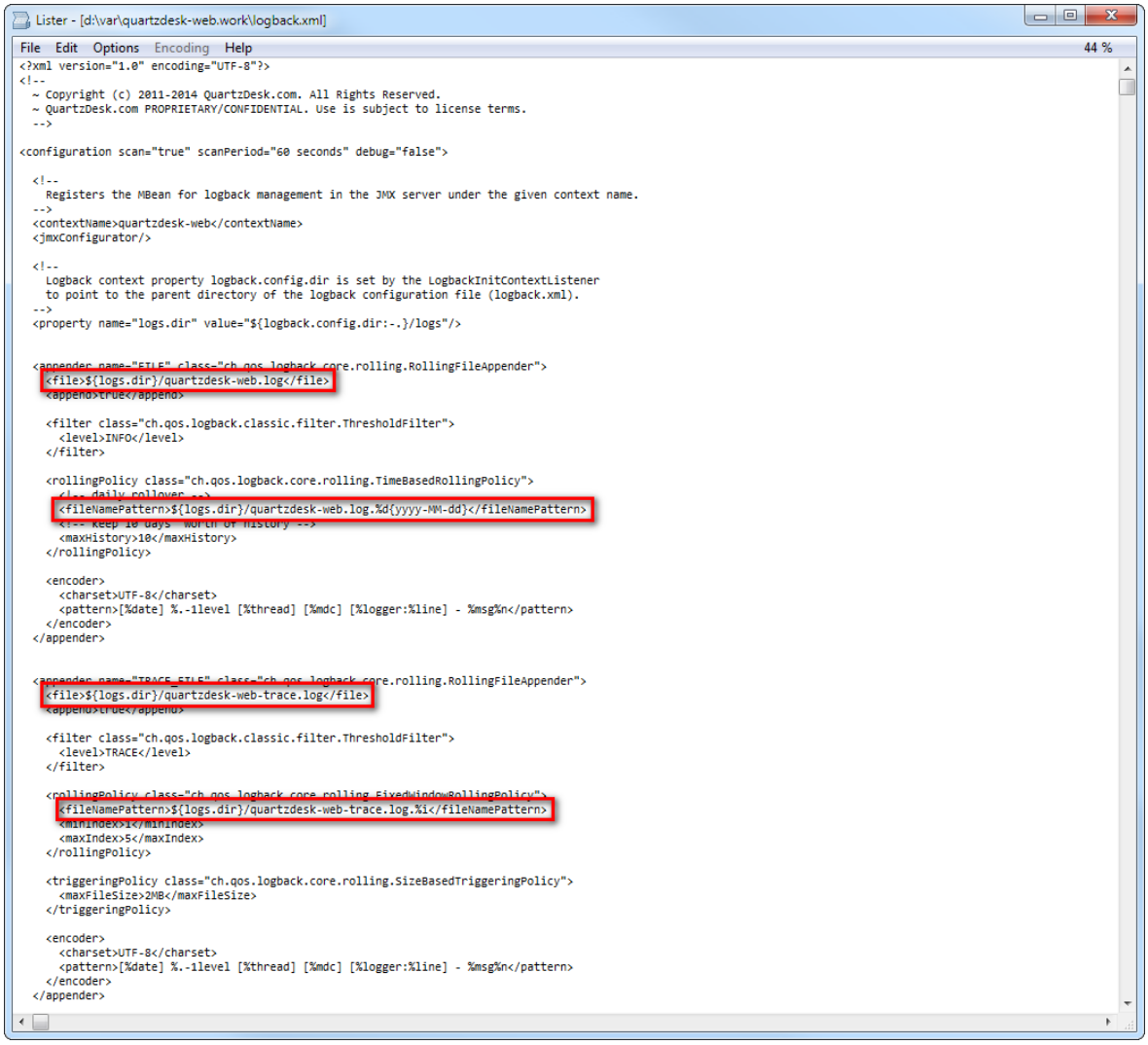
Rename the existing configuration file `quartzdesk.properties` located in the QuartzDesk web application work directory.

6.3 Rename Log Files

The names of QuartzDesk web application 3.x log files have changed.

Original Log File Name (2.x)	New Log File Name (3.x)
`quartzdesk.log`	`quartzdesk-web.log`
`quartzdesk-trace.log`	`quartzdesk-web-trace.log`

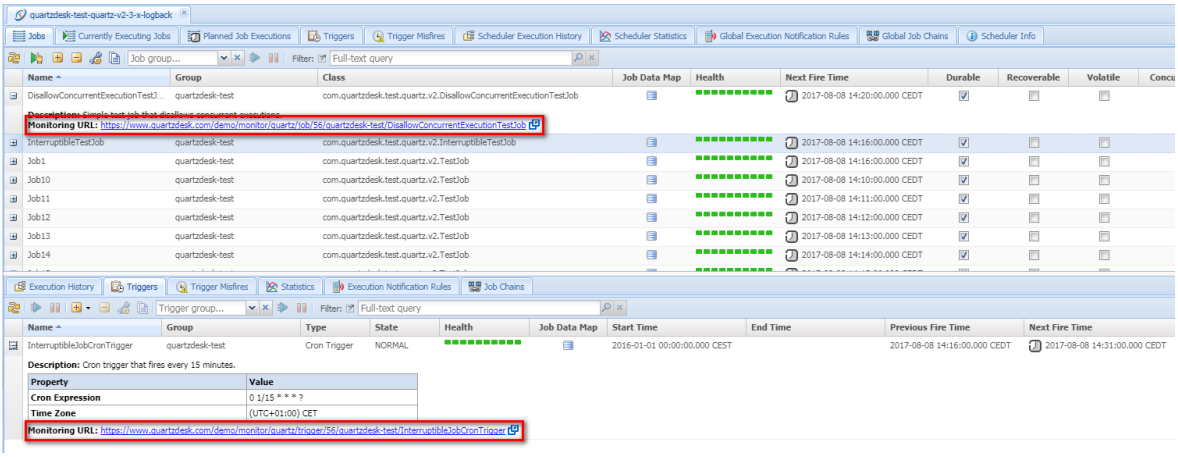
To use these new log file names, edit the QuartzDesk web application logging configuration file (`WORK_DIR/logback.xml`) and change the following lines:



Alternatively, extract the default `logback.xml` configuration file from the QuartzDesk web application 3.x WAR (`quartzdesk-web-x.y.z.war/extras/work/logback.xml`) and copy it to WORK_DIR.

6.4 Access to Monitoring URLs (REST API)

In QuartzDesk web application 2.x, the monitoring REST API URLs could be accessed by users with the QuartzDeskMonitor J2EE security role. In QuartzDesk web application 3.x, these monitoring URLs can be accessed by all authenticated users.



We recommend that you create a dedicated user account to access these monitoring URLs. The user account can be created in Settings → Users in the QuartzDesk GUI.

All monitoring URLs in QuartzDesk 3.x support the HTTP Basic authentication scheme where the user's authentication credentials are passed in the `Authorization` HTTP header. Please note that the same authentication scheme was used by monitoring URLs in QuartzDesk 2.x.

6.5 Access to JAX-WS Endpoints

In QuartzDesk web application 2.x, all JAX-WS web service endpoints could be accessed by users with the QuartzDeskService J2EE security role. In QuartzDesk web application 3.x, these web service end points can only be accessed by authenticated users with particular access permissions.

The following table lists all JAX-WS web services and the security permissions that are required to access these web services.

JAX-WS Service	Required Permission
Connection Service	`WS_CONNECTION`
Security Service	`WS_SECURITY`
Quartz Service	`WS_QUARTZ`
Quartz Execution History Service	`WS_QUARTZ_EXEC_HISTORY`
Quartz Execution Notification Rule Service	`WS_QUARTZ_EXEC_NOTIF_RULE`
Quartz Job Chain Service	`WS_QUARTZ_JOB_CHAIN`

We recommend that you create a dedicated user account to access these JAX-WS endpoints. The user account can be created in Settings → Users in the QuartzDesk GUI. Do not forget to assign the user the relevant permission(s).

All JAX-WS web service endpoints in QuartzDesk 3.x support the HTTP Basic authentication scheme where the user's authentication credentials are passed in the `Authorization` HTTP header. Please note that the same authentication scheme was used by JAX-WS endpoints in QuartzDesk 2.x.

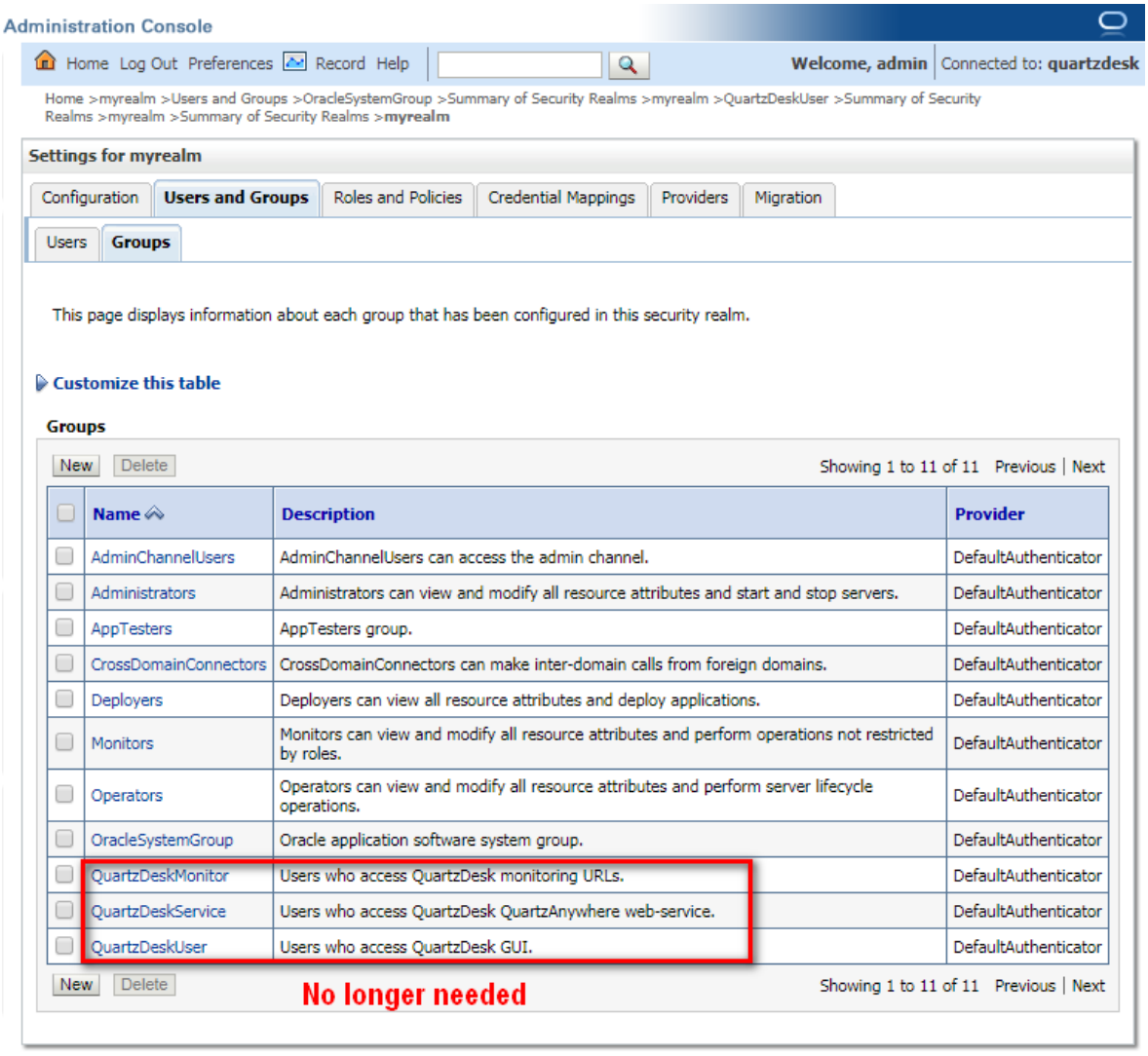
6.6 Remove Unused Groups

In WLAC go to WL_DOMAIN → Security Realms. Click on the security realm that was used by QuartzDesk web application 2.x

Select Users and Groups → Groups tab and remove the following new groups:

QuartzDeskUser
QuartzDeskMonitor
QuartzDeskService

These groups are no longer used by QuartzDesk web application 3.x.



7. Cluster Deployment Notes

When deploying the QuartzDesk web application to a WebLogic cluster you need to follow the configuration steps described in preceding chapters. In addition to these, there are several extra configuration steps that must be performed for a cluster deployment.

7.1 HTTP Session Replication and Affinity

The QuartzDesk web application makes use of HTTP sessions and to store some short-lived and user-specific data. To achieve high-availability (HA), it is necessary to make the session data available on all application cluster members so that when one cluster member becomes unavailable, the remaining cluster members can take over and handle user requests without the user noticing any service interruption. To make the session data available on all application cluster members, the HTTP session replication process must be enabled on the cluster.

The amount of data stored by the QuartzDesk web application in an HTTP session is kept at the absolute minimum to reduce the session replication overhead. The total size of data stored in the session does not exceed 1KB.

When configuring session replication, we recommend that you also enable session affinity (sticky-sessions) on the load-balancer so that all user requests are preferably passed to the WebLogic instance that handled the first user request that established the session.

Please refer to the WebLogic and load-balancer documentation for details on how to configure session replication and session affinity because the actual steps may vary depending on the WebLogic cluster topology and configuration.

7.2 Shared Work Directory

We recommend that you put the QuartzDesk web application work directory, described in chapter 0, on a shared drive and make this work directory available to all cluster members. Not only does this make application and configuration upgrading easier, it is actually requited by all “Save” (for example, Save Log, Save Chart etc.) actions provided by the QuartzDesk web application GUI. These actions trigger two subsequent HTTP requests where the first request prepares the data and stores it in the `WORK_DIR/tmp` directory and the second request downloads the data and makes the browser open the Save As dialog.

During a fail-over or if the session affinity is not enabled, it can easily happen that the first request is handled by cluster member A and the second request is handled by cluster member B. If A and B are not configured to use the same `WORK_DIR/tmp` directory, then B will fail to serve the data prepared by A during the preceding request because the data will not be found.

7.3 Logging Configuration

If you set up your cluster to use a shared QuartzDesk web application work directory, as described in the previous chapter, you will need to edit the QuartzDesk web application logging configuration file `WORK_DIR/logback.xml` and decide where QuartzDesk web application instances running on individual cluster members should log. There are two options:

1) Logging into the same (shared) log files.
2) Logging into separate log files.

The QuartzDesk web application uses two log files – `quartzdesk-web.log` and `quartzdesk-web-trace.log` that are stored in `WORK_DIR/logs` directory. The following chapters discuss these two options.

7.3.1 Using Shared Log Files

In order to make individual QuartzDesk web application instances log into the same log files, you must enable the prudent mode on both file appenders used in the `WORK_DIR/logback.xml` configuration file:

```
...

<appender name="FILE"
class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${logs.dir}/quartzdesk-web.log</file>
  <append>true</append>
  <prudent>true</prudent>
  ...
</appender>


<appender name="TRACE_FILE"
class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${logs.dir}/quartzdesk-web-trace.log</file>
  <append>true</append>
  <prudent>true</prudent>
  ...

  <!--
    We must use the TimeBasedRollingPolicy because the
    FixedWindowRollingPolicy is not supported in prudent mode!
  -->
  <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
    <!-- daily rollover -->
    <fileNamePattern>${logs.dir}/quartzdesk-web.log.%d{yyyy-MM-
dd}</fileNamePattern>
    <!-- keep 10 days' worth of history -->
    <maxHistory>10</maxHistory>
  </rollingPolicy>

  <!--
    The SizeBasedTriggeringPolicy removed because it is used only in
    conjunction with the FixedWindowRollingPolicy.
  -->

  <encoder>
    <charset>UTF-8</charset>
    <pattern>[%date] %.-1level [%thread] [%mdc] [%logger:%line] -
%msg%n</pattern>
  </encoder>
</appender>

...
```

For details on the Logback prudent mode, please refer to http://logback.qos.ch/manual/appenders.html#FileAppender.

Because prudent mode relies on exclusive file locks to manage concurrent access to the log files and these locks can have negative impact on the QuartzDesk web application's performance, we generally discourage using the prudent mode and shared log files.

7.3.2 Using Separate Log Files

In order to make individual QuartzDesk web application instances log into separate log files, you can use a JVM system property set on all cluster member JVMs. The value of this property must be unique for all cluster members. The property can be referred to from the `WORK_DIR/logback.xml` logging configuration file.

The following examples assume the use of the `cluster.member.instanceId` JVM system property, but any JVM system property name can be used.

There are two common approaches as to where the separate log files produced by individual QuartzDesk web application instances are stored:

1) Log files created under a common log root directory.

```
...

<appender name="FILE"
class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${logs.dir}/quartzdesk-web-${cluster.member.instanceId}.log</file>
  <append>true</append>

  ...

  <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
    <!-- daily rollover -->
    <fileNamePattern>${logs.dir}/quartzdesk-web-
${cluster.member.instanceId}.log.%d{yyyy-MM-dd}</fileNamePattern>
    <!-- keep 10 days' worth of history -->
    <maxHistory>10</maxHistory>
  </rollingPolicy>

  ...

</appender>


<appender name="TRACE_FILE"
class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${logs.dir}/quartzdesk-web-${cluster.member.instanceId}-
trace.log</file>
  <append>true</append>

  ...

  <rollingPolicy
class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
    <fileNamePattern>${logs.dir}/quartzdesk-web-
${cluster.member.instanceId}-trace.log.%i</fileNamePattern>
    <minIndex>1</minIndex>
    <maxIndex>5</maxIndex>
  </rollingPolicy>

  ...

</appender>

...
```

2) Log files created in separate (cluster member specific) log root directories.

```
...

<!--
  Logback context property logback.config.dir is set by the
  LogbackInitContextListener to point to the parent directory of the Logback
  configuration file (logback.xml).
-->
<property name="logs.dir" value="${logback.config.dir:-
.}/${cluster.member.instanceId}/logs"/>

...
```

7.4 Internal Quartz Scheduler

The QuartzDesk web application ships with an embedded Quartz scheduler to periodically execute its internal jobs. When deploying the QuartzDesk web application to a cluster, it is necessary to **assign unique instance IDs to Quartz scheduler instances** running in the clustered QuartzDesk web application instances.

For these purposes the QuartzDesk web application configuration (`quartzdesk-web.properties` file) provides the `scheduler.org.quartz.scheduler.instanceIdGenerator.class` configuration property. The value of this property must be a fully-qualified class name of a Java class that implements the `org.quartz.spi.InstanceIdGenerator` Quartz API interface. Quartz API provides two out of the box implementations suitable for clustered QuartzDesk web application deployments:

Implementation	Description
`org.quartz.simpl.HostnameInstanceIdGenerator`	This implementation is suitable for QuartzDesk web application deployments where individual clustered QuartzDesk web application instances run on distinct hosts and each of these hosts is assigned a unique hostname. This is the default implementation used by the QuartzDesk web application. No configuration changes are necessary to use this instance ID generator.
`org.quartz.simpl.SystemPropertyInstanceIdGenerator`	This implementation is suitable for QuartzDesk web application deployments where some of the clustered QuartzDesk web application instances run on the same host. This implementation extracts the Quartz scheduler instance ID from the `org.quartz.scheduler.instanceId` JVM system property that must be explicitly set. Please refer to the WebLogic documentation for details on how to add a new JVM system property.

Please refer to the table above and optionally modify the value of the `scheduler.org.quartz.scheduler.instanceIdGenerator.class` configuration property according to the cluster configuration.