QuartzDesk JVM Agent Installation and Upgrade Guide for WildFly AS 8.x, 9.x and 10.x

QuartzDesk Version: 2.x

April 24, 2017

Table of Contents

1. Purpose

This document describes the installation and upgrade process for the QuartzDesk JVM Agent 2.x on WildFly Application Server 8.x, 9.x and 10.x.

QuartzDesk JVM Agent is a Java Virtual Machine (JVM) plugin that must be installed in all JVMs powering applications with embedded Quartz schedulers managed by QuartzDesk. The QuartzDesk JVM Agent enables the following QuartzDesk features:

- Execution History
- Execution Notifications
- Execution Statistics
- Job Chaining
- Health Indicators
- Scheduler / Job / Trigger Monitoring
- Misfired Triggers

Please note that the installation of the QuartzDesk JVM Agent is required only by the QuartzDesk Standard and Enterprise editions. The QuartzDesk Lite edition does not provide access to any of the above features and therefore it does not require the installation of the QuartzDesk JVM Agent.

If the QuartzDesk GUI detects the QuartzDesk JVM Agent is not installed / enabled in a remote JVM it connects to, it displays a warning message and the above listed features are disabled in the QuartzDesk GUI.

If you experience any problems installing or upgrading the QuartzDesk JVM Agent, 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.
WFAC	WildFly Administrative Console.
WFAS	WildFly 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
AGENT_WORK_DIR	/var/quartzdesk-agent	QuartzDesk JVM Agent work directory.
DB_HOST	localhost	QuartzDesk JVM Agent database server host.
DB_PORT	5432	QuartzDesk JVM Agent database server port.
DB_NAME	quartzdesk	QuartzDesk JVM Agent database name.
DB_SCHEMA	quartzdesk	QuartzDesk JVM Agent database schema.
DB_USER	quartzdesk	QuartzDesk JVM Agent database user.
DB_PASSWORD	quartzdesk	QuartzDesk JVM Agent database user password.
WFAS_INSTALL_ROOT	/usr/local/wildfly	WildFly Application Server installation directory.
JDBC_DRIVER_JAR_PATH	/var/jdbc/mysql/ mysql-connector-java-5.1.23-bin.jar	File path of the JDBC driver JAR used by the QuartzDesk JVM Agent.

3. Requirements

3.1 Software Requirements

3.1.1 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.

3.1.2 Java

Sun/Oracle Java (JDK) 6, 7, 8.
IBM Java (JDK) 6, 7, 8.
OpenJDK 6, 7, 8.

3.1.3 Application Server

WildFly Application Server 8.x.
WildFly Application Server 9.x.
WildFly Application Server 10.x.

3.1.4 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

3.1.5 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 currently does not support the datetime2 data type. As a result, all datetime values written by the QuartzDesk 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#01_02.
PostgreSQL	JDBC4 PostgreSQL driver available at http://jdbc.postgresql.org/.

3.1.6 QuartzDesk JVM Agent Library

To install the QuartzDesk JVM Agent, you need to obtain the QuartzDesk JVM Agent JAR. The latest version can be downloaded at www.quartzdesk.com (click Downloads → Latest Release → View files → quartzdesk-agent-x.y.z.jar).

3.1.7 QuartzDesk Public API Library

QuartzDesk JVM Agent requires all applications with embedded Quartz schedulers deployed on the given JVM to have the QuartzDesk Public API Library on their classpath. The latest version can be downloaded at www.quartzdesk.com (click Downloads → Latest Release → View files → quartzdesk-api-x.y.z.jar).

The QuartzDesk Public API library is also available in the Maven Central repository – see http://search.maven.org/#search|ga|1|quartzdesk-api.

3.2 Hardware Requirements

QuartzDesk JVM Agent runs on any physical or virtualized hardware that supports the above software requirements.

4. Installation

4.1 Database

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

Create a new QuartzDesk JVM Agent database named `quartzdesk_agent`[1] (`DB_NAME`) owned by the `DB_USER`.

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

Please refer to the database engine documentation for details on how to perform the above database operations as they are all database-specific.

Please note that you do not have to create any other database objects (tables, keys, indices etc.) in the QuartzDesk JVM Agent database. These objects will be automatically created by the QuartzDesk JVM Agent during its first run.

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 3.1.5.

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

4.3 JVM Agent Work Directory

Create the QuartzDesk JVM Agent work directory (AGENT_WORK_DIR) anywhere on the local file system. The directory must be readable and writeable by the user the WFAS process runs under.

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

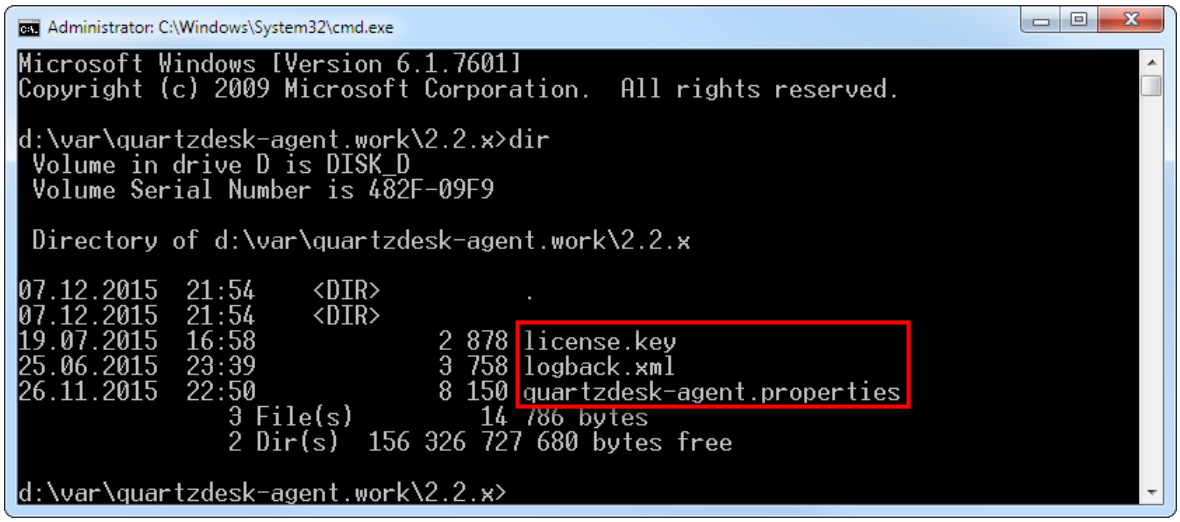
You can obtain a free 30-day trial license key at www.quartzdesk.com (open the Try / Purchase menu).

Open the QuartzDesk JVM Agent JAR file and copy all files from the `extras/work` directory into AGENT_WORK_DIR.

If you cannot open the JAR file directly, rename it to *.zip and then open it. Do not forget to rename the file back to *.jar once you have extracted the required files.

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

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



4.4 JVM Agent Configuration

Open the QuartzDesk JVM Agent configuration file `AGENT_WORK_DIR/quartzdesk-agent.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	>= 5.6	mysql
MySQL (Inno)	>= 5.6	mysql_inno
Oracle	== 8i	oracle8
Oracle	>= 9i	oracle9
PostgreSQL	== 8.1	postgres81
PostgreSQL	>= 8.2	postgres82

Uncomment the Agent JDBC pool configuration section based on the QuartzDesk JVM Agent database type. Make sure the JDBC pool configuration sections for other database types are commented out (prefixed with '#'). The default sample `quartzdesk-agent.properties` file assumes the use of a PostgreSQL database.

Adjust values of the JDBC pool configuration parameters to match your database configuration. You will typically need to change the default host value (localhost) in the `jdbc.url` parameter to point to DB_HOST. Please refer to the JDBC driver manual for a description of the JDBC URL format and related details.

Set the value of the `jdbc.pool.maxActive` JDBC pool configuration parameter to be 10-20% higher then the maximum number of **concurrently executing** Quartz jobs on the JVM the QuartzDesk JVM Agent will be installed on.

To adjust QuartzDesk JVM Agent logging parameters, edit the `AGENT_WORK_DIR/logback.xml` configuration file. The default sample `logback.xml` configuration file creates the QuartzDesk JVM Agent log under the `AGENT_WORK_DIR/logs`

directory that is automatically created when the QuartzDesk JVM Agent starts. Please refer to the Logback Manual for Logback configuration details.

4.5 Install JVM Agent

To manage Quartz schedulers embedded in applications deployed in WFAS, you must first enable remote JMX access to WFAS. Please refer to the **How to Enable Remote JMX Access to Quartz Schedulers** document available at www.quartzdesk.com/documentation/how-tos. Once the remote JMX access has been enabled, continue with the steps below.

4.5.1 Create WildFly AS Module

Create a new module directory `WFAS_INSTALL_ROOT/modules/system/layers/base/com/quartzdesk/agent/main`.

Copy the QuartzDesk JVM Agent JAR to the created module directory and rename the file to `quartzdesk-agent.jar`.

Create a new `module.xml` file in the created WildFly module directory and paste the following data to it:

```
<?xml version="1.0" encoding="UTF-8"?>

<module xmlns="urn:jboss:module:1.3" name="com.quartzdesk.agent">
  <resources>
    <resource-root path="quartzdesk-agent.jar"/>
  </resources>

  <dependencies>
    <module name="javax.xml.bind.api"/>
  </dependencies>
</module>
```

4.5.2 Update WildFly AS Startup Script

This chapter describes changes to the WFAS startup script and assumes that WFAS is run in the Standalone mode. If you are running WFAS in the Domain mode, the required changes are similar, but you need to apply them to a different WFAS startup script (e.g. domain.conf.bat / domain.conf etc.).

Windows

Open the `WFAS_INSTALL_ROOT\bin\standalone.conf.bat` file in a text editor and add the following lines at the end of the file. Replace the red JAR version numbers with the actual JAR version numbers available in the WFAS installation.

```
rem QuartzDesk JVM Agent: Set WILDFLY_HOME used in paths below
set "WILDFLY_HOME=<WILDFLY_INSTALL_ROOT>"

rem QuartzDesk JVM Agent: Agent JAR and JDBC driver JAR(s)
set "JAVA_OPTS=%JAVA_OPTS% -
javaagent:%WILDFLY_HOME%\modules\system\layers\base\com\quartzdesk\agent\m
ain\quartzdesk-agent.jar"

set "JAVA_OPTS=%JAVA_OPTS% -Xbootclasspath/p:<JDBC_DRIVER_JAR_PATH>"
set "JAVA_OPTS=%JAVA_OPTS% -Dquartzdesk-agent.work.dir=<AGENT_WORK_DIR>"

rem QuartzDesk JVM Agent: Use the JBoss logmanager
set "JAVA_OPTS=%JAVA_OPTS% -
Djava.util.logging.manager=org.jboss.logmanager.LogManager"
set "JAVA_OPTS=%JAVA_OPTS% -
Xbootclasspath/p:%WILDFLY_HOME%\modules\system\layers\base\org\jboss\logma
nager\main\jboss-logmanager-2.0.0.Final.jar"
set "JAVA_OPTS=%JAVA_OPTS% -
Xbootclasspath/p:%WILDFLY_HOME%\modules\system\layers\base\org\jboss\log4j
\logmanager\main\log4j-jboss-logmanager-1.1.2.Final.jar"

rem QuartzDesk JVM Agent: Added JBoss logmanager and QuartzDesk agent
system packages (com.quartzdesk.agent and ext)
set "JAVA_OPTS=%JAVA_OPTS% -
Djboss.modules.system.pkgs=org.jboss.byteman,org.jboss.logmanager,com.quar
tzdesk.agent,ext"
```

Unix/Linux

Open the WFAS_INSTALL_ROOT/bin/standalone.conf file in a text editor and add the following lines at the end of the file. Replace the red JAR version numbers with the actual JAR version numbers available in the WFAS installation.

```
# QuartzDesk JVM Agent: Set WILDFLY_HOME used in paths below
WILDFLY_HOME="<WILDFLY_INSTALL_ROOT>"

# QuartzDesk JVM Agent: Agent JAR and JDBC driver JAR(s)
JAVA_OPTS="${JAVA_OPTS} -
javaagent:${WILDFLY HOME}/modules/system/layers/base/com/quartzdesk/agent/
main/quartzdesk-agent.jar"

JAVA_OPTS="${JAVA_OPTS} -Xbootclasspath/p:<JDBC_DRIVER_JAR_PATH>"
JAVA_OPTS="${JAVA_OPTS} -Dquartzdesk-agent.work.dir=<AGENT_WORK_DIR>"

# QuartzDesk JVM Agent: Use the JBoss logmanager
JAVA_OPTS="${JAVA_OPTS} -
Djava.util.logging.manager=org.jboss.logmanager.LogManager"
JAVA_OPTS="${JAVA_OPTS} -
Xbootclasspath/p:${WILDFLY_HOME}/modules/system/layers/base/org/jboss/logm
anager/main/jboss-logmanager-2.0.0.Final.jar"
JAVA_OPTS="${JAVA_OPTS} -
Xbootclasspath/p:${WILDFLY_HOME}/modules/system/layers/base/org/jboss/log4
j/logmanager/main/log4j-jboss-logmanager-1.1.2.Final.jar"

# QuartzDesk JVM Agent: Added JBoss logmanager and QuartzDesk agent system
packages (com.quartzdesk.agent and ext)
JAVA_OPTS="${JAVA_OPTS} -
Djboss.modules.system.pkgs=org.jboss.byteman,org.jboss.logmanager,com.quar
tzdesk.agent,ext"
```

4.6 Install Public API Library

The QuartzDesk Public API Library (quartzdesk-api-<version>.jar) works an an interface between the Quartz library (typically distributed as quartz-<version>) used by an application and the QuartzDesk JVM Agent. **The QuartzDesk Public API Library must be loaded by the same Java classloader that loads the Quartz library.**

In WFAS, there are two typical cases how the Quartz library is deployed.

(1) Quartz library is embedded in the application, typically in its `WEB-INF/lib` folder. In this case, the QuartzDesk Public API Library must be copied to this folder.

Please note that the QuartzDesk Public API Library is available in the Maven Central repository and if you add it as a runtime dependency to the application's POM, it can be automatically copied to the application's `WEB-INF/lib` folder by Maven.

(2) Quartz library is deployed as a WildFly module and this module is then used by the application. In this case, the QuartzDesk Public API Library must be copied to the Quartz module directory and added to the list of resources in the module's deployment descriptor file (module.xml).

No application code changes are required to install the QuartzDesk Public API Library.

4.7 Enable Log Message Interception

To enable interception of log messages produced by executed Quartz jobs, it is necessary to modify the logging configuration of Quartz-enabled applications running on the JVM the

QuartzDesk JVM Agent is installed on. The required configuration changes are simple and vary for individual logging frameworks.

For details please refer to the **How to Enable Log Message Interception in Applications** document available at www.quartzdesk.com/documentation/how-tos.

Please note that this step is optional. When the log message interception is not configured, the following QuartzDesk features will not be available:

1. Viewing logs in the Execution History panels.
2. Viewing logs of currently executing jobs in the Currently Executing Jobs panel.
3. Accessing and analyzing log messages in JavaScript expressions in Execution Notification rules.
4. Attaching logs to messages sent by Executon Notification rules.

4.8 Stop WildFly AS

Stop WFAS. Please refer to the WildFly AS administration documentation for details on how to stop WFAS.

Wait for the action to complete.

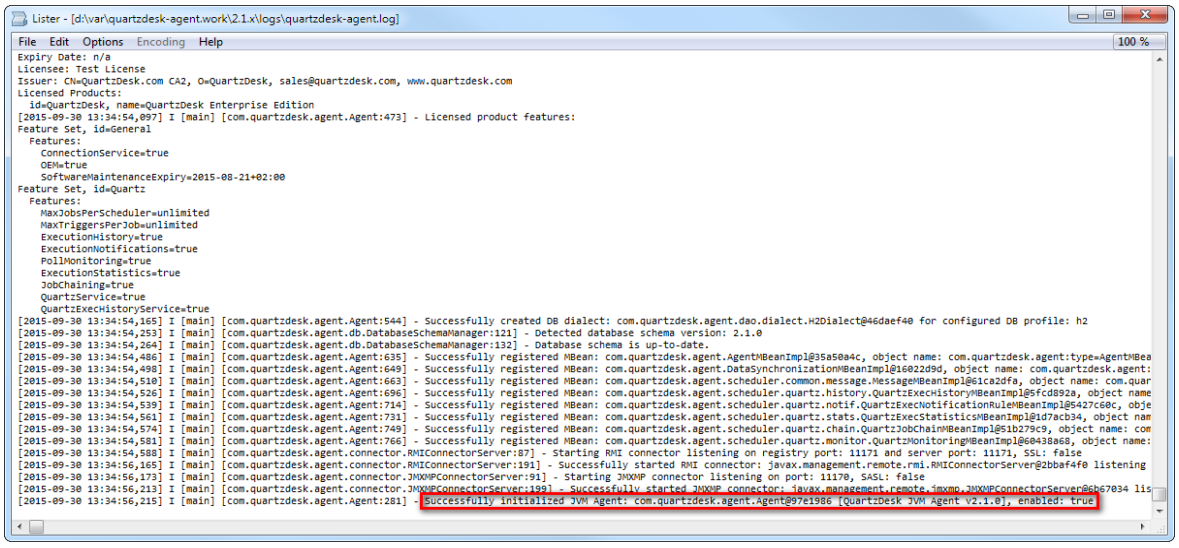
4.9 Start WildFly AS

Start WFAS. Please refer to the WildFly AS administration documentation for details on how to start WFAS.

Wait for the action to complete.

Check the WFAS logs for errors.

Check the QuartzDesk JVM Agent logs located in `AGENT_WORK_DIR/logs` directory for errors and verify the release number of the installed QuartzDesk JVM Agent.



Verify that all applications deployed to WFAS work as expected.

5. Upgrading

5.1 Stop WildFly AS

Stop WFAS by following the steps outlined in 4.8.

5.2 Backup

Backup your QuartzDesk JVM Agent database. We recommend performing a **full database backup**.

Backup the contents of the QuartzDesk JVM Agent work directory.

Store the backups in a safe place so that you can restore the original QuartzDesk JVM Agent version if the need arises.

5.3 Upgrade JVM Agent

Replace the old QuartzDesk JVM Agent JAR file in the WildFly module (refer to chapter 4.5.1 for details) with the new QuartzDesk JVM Agent JAR file.

Rename the `AGENT_WORK_DIR/quartzdesk-agent.properties` configuration file to `quartzdesk-agent.properties.old`.

Open the QuartzDesk JVM Agent archive (`quartzdesk-agent-x.y.z.jar`) and copy the `extras/work/quartzdesk-agent.properties` configuration file to `AGENT_WORK_DIR`.

If you cannot open the JAR file directly, rename it to *.zip and then open it. Do not forget to rename the file back to *.jar once you have extracted the required files.

Adjust the values of the configuration properties in the new configuration file `AGENT_WORK_DIR/quartzdesk-agent.properties` to match your system setup. You can use the old configuration file as a reference.

Please refer to 4.4 for a description of the configuration parameters that you need to adjust.

5.4 Upgrade Public API Library

The steps necessary to upgrade this library depend on the way it has been deployed. Please refer to 4.6 for details.

5.5 Start WildFly AS

Start WFAS by following the steps outlined in 4.9.

1. Cluster Deployment Notes

When configuring the QuartzDesk JVM Agent in a WildFly 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 in cluster deployments.

1.1 Shared Work Directory

We recommend that you put the QuartzDesk JVM Agent work directory, described in chapter 4.3, on a shared drive and make this work directory available to all WildFly cluster members.

1.2 Logging Configuration

If you set up your cluster to use a shared QuartzDesk JVM Agent work directory, as described in the previous chapter, you will need to edit the QuartzDesk JVM Agent logging configuration file `AGENT_WORK_DIR/logback.xml` and decide where QuartzDesk JVM Agent 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.

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

1.2.1 Using Shared Log Files

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

```
...

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


<appender name="TRACE_FILE"
class="ext.ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${logs.dir}/quartzdesk-agent-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="ext.ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
    <!-- daily rollover -->
    <fileNamePattern>${logs.dir}/quartzdesk-agent-trace.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 JVM Agent's performance, we generally discourage using the prudent mode and shared log files.

1.2.2 Using Separate Log Files

In order to make individual QuartzDesk JVM Agent 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 then be referred to from the `AGENT_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 JVM Agent instances are stored:

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

```
...

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

  ...

  <rollingPolicy
class="ext.ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
    <!-- daily rollover -->
    <fileNamePattern>${logs.dir}/quartzdesk-agent-
${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="ext.ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${logs.dir}/quartzdesk-agent-${cluster.member.instanceId}-
trace.log</file>
  <append>true</append>

  ...

  <rollingPolicy
class="ext.ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
    <fileNamePattern>${logs.dir}/quartzdesk-agent-
${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"/>

...
```

1.3 Installation and Upgrade Roll-Out

As described in chapter 4.1, the QuartzDesk JVM Agent automatically creates all required database objects in the configured database upon its first start. Similarly, upon every QuartzDesk JVM Agent upgrade the agent automatically applies required changes to the configured database.

If you have configured multiple QuartzDesk JVM Agents to use the same database, collisions are likely to occur if multiple agents are started concurrently and all attempt to realize the database initialization/upgrade procedure described above. To avoid these collisions, please start a single JVM with the configured QuartzDesk JVM Agent and let the agent apply the database changes. Once the database changes have been successfully applied, it is possible to start the other agents (JVMs).

You can check for the following line in the QuartzDesk JVM Agent log to see if the database has been successfully initialized/upgraded. This log line indicates that the agent has been successfully started at which point all database schema changes have been applied.

```
...

[2015-09-30 13:34:56,215] I [main] [com.quartzdesk.agent.Agent:281] -
Successfully initialized QuartzDesk JVM Agent:
com.quartzdesk.agent.Agent@97e1896 [QuartzDesk JVM Agent v2.1.0], enabled:
true

...
```