Track

Custom Installation Guide

Version 11.1

Document Number: MTCIG11.001

Document Title: meshIQ Track Custom Installation Guide
Document Number: MTCIG11.001
Document Release Date: September 2024

Published by:
Research & Development
meshIQ
88 Sunnyside Blvd, Suite 101
Plainview, NY 11803

Table of Contents

Chapter 1: Introduction

The recommended methods of installing Track are to use a deployment pack or to install using docker. This document addresses the needs of customers who are performing a custom install (for example, using some components that were previously installed) or upgrading Track components within an existing installation. It covers the manual equivalents of the installation and configuration scripts that are automated through the deployment pack method. It contains instructions for the minimum steps required to install, configure, and run Track.

1.1 Recommendations

We recommend that you create a dedicated OS user (for example, `meshiq`) to be the owner of all the files installed. All meshIQ-related applications described in this document should be executed using the `meshiq` user.

We also recommend that you install all software in the same folder. The standard location is `/opt/meshiq`. Be sure to set the following environment variable to this location:

```
APIN_HOME=/opt/meshiq
```

This should be done in the appropriate `.profile` file for the user (such as `meshiq`) that will be used to run all Track processes.

1.2 How this Guide is Organized

Chapter 1: Introduction to and information about this guide, its history and intended audience, and how to get help.

Chapter 2: Identifies the prerequisite system and environment setup items that must be performed before you can begin installation.

Chapter 3: How to install Java.

Chapter 4: How to install and configure Zookeeper.

Chapter 5: How to install and configure Solr.

Chapter 6: How to install and configure ActiveMQ.

Chapter 7: How to install and configure Kafka.

Chapter 8: Instructions for starting the installed components (Zookeeper, ActiveMQ, Solr, and Kafka).

Chapter 9: How to install Core Services.

Chapter 10: Instructions for downloading and installing the Track Server Plugin package and required dependencies.

Chapter 11: Instructions for installing the Track DB Utility for Solr.

Chapter 12: Covers the configuration of Track, including preparing components for use and installing the web server, user interface, and more.

1.3 History of this Document

Table 1. Document History

Release Date	Doc Number	Summary
March 2023	XRCIG100.001	Initial release.
April 2023	XRCIG100.002	Standardized SSO file names, parameters, and locations.
June 2023	XRCIG100.003	Added 1.6 sections for updating ActiveMQ (5.17.4) and Kafka (3.4.0)
September 2024	MTCIG11.001	Branding changes for version 11; removal of content specific to legacy versions; ActiveMQ 5.18; removal of deprecated functionality or components (Apache Storm, Subscriptions, Triggers).

1.4 User Feedback

We encourage all users and administrators to submit comments, suggestions, corrections, and recommendations for improvement for all documentation. Please send your comments via e-mail to: *support@meshiq.com*. You will receive a response, along with the status of any proposed change, update, or correction.

1.5 Release Notes

Refer to the *meshIQ Platform Release Notes* in the Resource Center.

1.6 Intended Audience

This guide is intended for systems administrators and operating engineers responsible for the installation and administration of the Track environment.

1.7 Technical Support

Use one of the following methods for technical support:

- **Call:** 800-963-9822 ext. 1
  If you are calling from outside the United States: 001-516-801-2100
- **Email:** support@meshiq.com
- **Resource center:** *https://customers.meshiq.com*
- **Automated support system:** *http://support.meshiq.com/* (user ID and password required)

Chapter 2: Prerequisite Setup Items

Follow the prerequisite setup procedures below prior to installing Track components.

2.1 Disable Swap

For the best performance, it is highly recommended that you turn off swap within the operating system on all virtual machine servers running Track. To disable swap, do the following:

1. Identify configured swap devices and files with the following command:

```
cat /proc/swaps
```

2. Turn off all swap devices and files with the following command:

```
swapoff -a
```

Remove any matching references found in /etc/fstab.

2.2 Linux Firewall

By default, some Linux distributions have an active firewall that block ports which are needed to connect to Track services remotely. The following ports should be opened, as these are the ports Track utilizes:

ZooKeeper: 2181
Solr: 8983
ActiveMQ: 8161
meshIQ Domain Server: 2323, 3000
Track CEP: 3005
Track UI CEP : 3010
Track: 8080
Gateway: 6580

2.3 Modify limits.conf (Required)

Modify /etc/security/limits.conf by adding the values below for your user. (Replace "meshiq" with the user that will start Track processes.)

```
meshiq          soft    nofile  65536
meshiq          hard    nofile  65536

meshiq          soft    nproc   65536
meshiq          hard    nproc   65536
```

2.4 User Environment Setup (Optional)

Optional environment variable configuration: Please note that to run the standalone appliance as delivered in the package, it is not necessary to set the environment variables at the user or system level. However in some cases, such as deployment of a single server development environment, it may be desirable to set them.

Add the following lines to the **.bash_profile** of the user who will run the Track services:

- export APIN_HOME=<path_to_Track_filesystem>
- export APIN_LOGS=$APIN_HOME/AutoPilotM6/logs
- export APM6_HOME=$APIN_HOME/AutoPilotM6
- export JAVA_HOME=$APIN_HOME/java/current
- export SOLR_HOME=$APIN_HOME/solr/current
- export KFKA_HOME=$APIN_HOME/kafka/current
- export PATH=$PATH:$APIN_HOME/sbin/:$JAVA_HOME/bin:$SOLR_HOME/bin:$KFKA_HOME/bin


NOTE

Only the environment variable $APIN_HOME requires the full path to the installation directory. All other environment variables will be built from this variable.

2.5 CEP Grouping (for Multiple CEPs)

Track Services can be deployed to one or more CEP servers. Different CEP servers may host services for streaming, computing, writing, and client processing. The grouping of individual services on servers varies and is tailored for each customer based on their requirements and configuration.

Chapter 3: Installing Java

The version of Java to install depends on the version of Track you are using. OpenJDK 11 is required for Track version 11.x.

If you are using multiple nodes for Track, you *must* install Java on *every* Track node.

3.1 OpenJDK 11

To install OpenJDK 11, do the following:

1. Download the tar image distribution file for the *latest version* of OpenJDK 11.
2. Untar the distribution in `$APIN_HOME`.
3. Set `JAVA_HOME` to indicate the root folder where Java was installed. This should be done in the appropriate .profile file for the user (such as `meshiq`) that will be used to run all Track processes.

   `JAVA_HOME=$APIN_HOME/jdk-11.x.y`

If you are using multiple nodes for Track, repeat the above steps on *every* Track node.

Chapter 4: Installing and Configuring Zookeeper

Track version 11 uses Zookeeper version 3.9.1.

4.1 Zookeeper 3.9.1 (Solr 8)

To install Zookeeper 3.9.1, do the following:

1. Download the tar image distribution file for Apache Zookeeper 3.9.1.
2. Untar the distribution in `$APIN_HOME`.

To configure Zookeeper 3.9.1, do the following:

1. In the `conf` subfolder within the Zookeeper distribution, create a new file, `zoo.cfg`, as a copy of the supplied sample (`cp zoo_sample.cfg zoo.cfg`).
2. Edit `zoo.cfg` as follows:
   - Change `dataDir` if you do not want to use the default folder.
   - Uncomment out `autopurge.snapRetainCount` and set the value to 50:

     ```
     autopurge.snapRetainCount=50
     ```

   - Uncomment out `autopurge.purgeInterval`:

     ```
     autopurge.purgeInterval=1
     ```

   - Add following to the end of the file (after `autopurge.purgeInterval`):

     ```
     maxSessionTimeout=120000

     # web admin port (default is 8080, which conflicts with Track web
     server)

     admin.serverPort=8188

     # enable Solr web UI to connect to ZooKeeper

     4lw.commands.whitelist=mntr,conf,ruok
     ```

Chapter 5: Installing and Configuring Solr

Track version 11 uses Solr version 8.11.1.

If you are using a cluster of nodes, you must install Solr on *every* server that will run Solr. All servers must run the same version of Solr.

5.1 Solr 8.11.1

To install Solr 8.11.1, do the following:

1. Download the tar image distribution file for Apache Solr 8.11.1.
2. Untar the distribution in `$APIN_HOME`.

To configure Solr 8.11.1, do the following:

1. Create folder: `$APIN_HOME/solr`.
2. Copy the following files from `solr-8.11.1/server/solr` to `$APIN_HOME/solr/var/data`:
   - `solr.xml`
3. Edit `bin/solr.in.sh` within the Solr distribution, adding the following lines:

```
ZK_HOST="localhost:2181/xraysolr"
SOLR_PID_DIR="$APIN_HOME/solr/var"
SOLR_HOME="$APIN_HOME/solr/var/data"
SOLR_LOGS_DIR="$APIN_HOME/solr/var/logs"
SOLR_PORT="8983"
SOLR_MODE="solrcloud"
SOLR_JAVA_HOME=$JAVA_HOME
SOLR_JAVA_MEM="-Xmx12g -Xms12g"
GC_TUNE= " \
-XX:+UseG1GC \
-XX:+AggressiveOpts \
-XX:+PerfDisableSharedMem \
-XX:+ParallelRefProcEnabled \
-XX:G1HeapRegionSize=4m \
-XX:MaxGCPauseMillis=500
"
```

If you are using a cluster of nodes, repeat these steps on *all* Solr nodes.

Chapter 6: Installing and Configuring ActiveMQ

ActiveMQ 5.18.3 is used in Track version 11.

6.1 ActiveMQ 5.18.3

NOTE

The following procedure assumes that the ActiveMQ distribution is installed in `$APIN_HOME/actmq`.

1. Stop ActiveMQ under `$APIN_HOME/sbin` by using the command `./stop.sh mq`.
2. To update the ActiveMQ component, download the ActiveMQ 5.18.3 version from *https://activemq.apache.org/components/classic/download/classic-05-18-03*.
3. Make the following changes to `$APIN_HOME/actmq/current/conf/activemq.xml`.
   a. Update the <broker> element by adding `schedulePeriodForDestinationPurge="300000"`, as shown in **bold**.

   ```
   <broker xmlns="http://activemq.apache.org/schema/core"
   brokerName="JKMessageBus" dataDirectory="${activemq.data}"
   schedulePeriodForDestinationPurge="300000">
   ```

   b. (Optional.) Change brokerName in the line from step a.
   c. Within the policyEntries element found here:

   ```
   <destinationPolicy>
               <policyMap>
                 <policyEntries>
   ```

   Add the following:

   ```
   <policyEntry queue="jkool.client.>" gcInactiveDestinations="true"
   inactiveTimoutBeforeGC="120000" sendAdvisoryIfNoConsumers="true"/>
   ```

4. To be able to connect to the web console remotely, it must be run on a local machine. Restore the "host" value for bean "jettyPort" from "127.0.0.1" back to "0.0.0.0" by updating line 123 of `$APIN_HOME/actmq/current/conf/jetty.xml` (shown in **bold**):

   ```
       <bean id="jettyPort" class="org.apache.activemq.web.WebConsolePort"
   init-method="start">
                 <!-- the default port number for the web console -->
           <property name="host" value="0.0.0.0"/>
           <property name="port" value="8161"/>
       </bean>
   ```

5. Start ActiveMQ under `$APIN_HOME/sbin` by using the command `./start.sh mq`

Chapter 7: Installing and Configuring Kafka

Track version 11 uses Kafka 3.4.0.

After you install and configure Kafka, you will create topics. Creating topics is covered later in this document.

7.1 Kafka 3.4.0

1. Stop the Kafka service by using the command
   `./kafka-server-stop.sh ../config/server.properties`
   in `$APIN_HOME/kafka_a.b-x.y.z`.
2. To update the Kafka component, download the latest version of Kafka 3.x from *https://kafka.apache.org/downloads* and place the file in `$APIN_HOME/kafka`.
3. To configure Kafka 3.4.0, edit `server.properties` in `config` subfolder within the Kafka distribution:
   - Change the following properties:
     - under Socket Server Settings:

       `num.network.threads` from `3` to `8`

       `num.io.threads` from `8` to `16`
     - By default, `log.dirs` is set to the folder shown below. But you can change it if you do not want to use the default folder.

       `log.dirs=/tmp/kafka-logs`
     - under Log Basics:

       `num.partitions` from `1` to `16`
     - under Log Flush Policy, uncomment the following line:

       `log.segment.bytes=1073741824`
     - `zookeeper.connect` from default value to `localhost:2181/xraykafka`
   - Add the following properties under Server Basics:
     - `message.max.bytes=10485760`
     - `replica.fetch.max.bytes=10485760`
4. After the above steps have been completed, restart the Kafka service by using the command `./start.sh kafkasrv` to apply the changes.
5. Recreate Kafka topics using the command:
   ```
   cd $APM6_HOME/jkool/scripts
   ./create-kafka-topics.sh -zh $ZKHOST -k $APIN_HOME/kafka_a.b-x.y.z -zr $ZKROOT
   ```

ZKHOST is the IP Address where ZooKeeper is running. The -zr argument may or may not be needed, depending on your Kafka setup.

Chapter 8: Start Components

Before continuing with configuring Track, the following components must be started. (For those that are clustered, *all nodes in the cluster must be started*.)

Zookeeper should be started first since the others (except for ActiveMQ) require it. Otherwise, the order in which these components are started is not important.

8.1 Start Zookeeper

To start Zookeeper, enter the following at the command prompt:

```
cd $APIN_HOME/apache-zookeeper-x.y.z-bin

./zookeeper-server-start.sh -daemon ../config/zookeeper.properties
```


NOTE

The root folder generally has the following format (depending on the version of ZooKeeper that is installed, the bracketed portions may or may not be present): `[apache-]zookeeper-x.y.z[-bin]`

8.2 Starting ActiveMQ

To start ActiveMQ, enter the following at the command prompt:

```
cd $APIN_HOME/apache-activemq-x.y.z

./bin/activemq start
```

8.3 Starting Solr

The following step must be performed on *all* Solr servers.

To start Solr, enter the following at the command prompt:

```
cd $APIN_HOME/solr-x.y.z

./bin/solr start
```

8.4 Starting Kafka

For Kafka clusters, the following step must be performed on *all* Kafka servers.

To start Kafka, enter the following at the command prompt:

```
cd $APIN_HOME/kafka_a.b-x.y.z

./kafka-server-start.sh -daemon ../config/server.properties
```

Chapter 9: Installing Core Services

Download and install the appropriate version of Core Services for the version of Track being installed Refer to the *Core Services Installation Guide* in the *meshIQ Platform Documentation Library* for detailed instructions on how to install Core Services and plug-ins. In this document it is assumed that the environment variable AUTOPILOT_HOME points to the root folder for Core Services. The default location is `/opt/meshiq/platform`. For the environment variable $APIN_HOME, also in this document, the default location is `/opt/meshiq`.

9.1 Installing and Configuring the Job Scheduler Plug-in

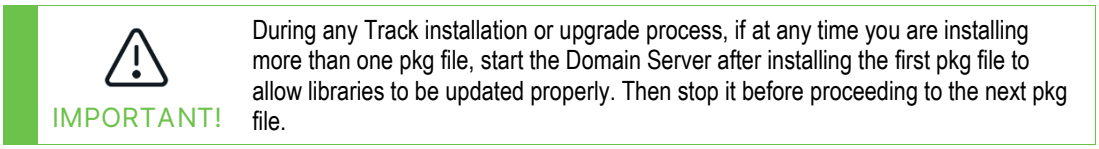
Refer to the *How do I use the Job Scheduler Expert?* article in the meshIQ Resource Center for information on how to install and configure the Job Scheduler Plug-in.

Chapter 10: Installing the DATA_SERVICE Plug-in

Track Server has the following dependencies:

- Job Scheduler. (The version of Job Scheduler that is required depends on the version of Track being installed.)

1. Download the appropriate Track Server Plugin package (along with the required dependencies):
   - DATA_SERVICE-11.0.1.pkg (for AP11_SU01+)
2. Using pkgman, install the pkg files for your version of Track. See the important note below.



After installing the last pkg file, leave the Domain Server running. Do not start the CEP yet.

Chapter 11: Installing the ds-db-utils Utility

Since the utilities provided require scripts that are included in Solr, the Track ds-db-utils utility should be installed on one of the Solr servers.

1. Download the tar image distribution file that matches the DATA_SERVICE version you installed in the previous chapter:

   - ds-db-utils-11.1.1.tar.gz

   Place the file in $APIN_HOME/misc/dbapi/ds-db-utils-11.1.1.tar.gz.

2. Untar the distribution in $APIN_HOME/misc/dbapi.

Chapter 12: Configuring Track

This chapter covers the configuration of Track itself, including servers, service experts, the web server and, if applicable, machine learning.

12.1Configuring Track Servers

Track server configuration includes setting properties and preparing the components for use by creating Kafka topics and initializing the Solr database.

12.1.1 System properties

Properties files exist at both the global and node levels.

12.1.1.1 Global

The DATA_SERVICE pkg will define properties in the global.properties file in the core services installation folder (for example, /opt/meshiq/platform). The default values are fine for most installations, but the following changes will need to be made.

Update:

- `jkool.kafka.server`.

  The default value for the `jkool.kafka.server` property assumes that Kafka is running locally on the default port. If Kafka is running remotely or if it is not on the default port, change the value(s) to match your actual Kafka location.

Add:

- `property jkool.db.url=localhost:2181/xraysolr`

  The value for the `jkool.db.url` property *must* match the value of ZK_HOST defined earlier in the *Installing and Configuring Solr* chapter.

- `property jkool.auth.service.class=com.nastel.jkool.auth.DomainAuthService`

  If you are using MP Domain Server user authentication instead of the default Track user authentication, you must define the value above for the `property jkool.auth.service.class` property.

12.1.1.2 Node

The installation creates some entries in node.properties. These default entries can be left unchanged.

A sample of the node.properties file is provided below.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; set properties first
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

property server.type = Server
property server.naming.url.port = 2325
property server.user.url.port   = 3005
property server.shutdown.grace  = 3000

property server.facts.capacity  = 100000
property server.net.sessions.poolsize = 5
property server.net.agents.poolsize = 1000
property server.log.size = 500000

property fatpipes.config.list.file={autopilot.home}/localhost/fatpipes4j.property.file.list
property fatpipes.pipeline.config.list.file={autopilot.home}/localhost/fatpipes4j.pipelines.list

; register server name (/pop uses host name, or can be explicit value, Nodename = TESTCEP)
HostName
register NodeName = /pop
property jkool.install.dir={autopilot.home}/jkool/
property fatpipes.jndi.file={jkool.install.dir}config/jndi.properties
property fatpipes.data.dir=.
property JOB_SCHED=Job_Scheduler
property jkool.ml.ts.defaults.file={jkool.install.dir}config/mlmodel-defaults.json
property java.net.preferIPv4Stack=true
property fatpipes.infinispan.tag={NodeName}
```

12.1.2 Creating Kafka topics

Before creating Kafka topics, make sure that both Kafka and Zookeeper are running.

The script for creating Kafka topics requires scripts that are included in Kafka. Therefore it must be run on a node where Kafka is installed. If Kafka is not installed locally, copy all the Kafka scripts from `$AUTOPILOT_HOME/jkool/script/` to the Kafka server. (For Kafka clusters, run it once on any of the Kafka servers.)

To create Kafka topics, run the following:

```
$AUTOPILOT_HOME/jkool/script/create-kafka-topics.sh
```

12.1.3 Initializing Solr

Solr is initialized using utilities provided in the Track ds-dbapi-solr Utility. As mentioned above in Installing the ds-db-utils Utility, this utility must be installed on one of the Solr servers.

Before proceeding, make sure that both Solr and Zookeeper are running. If Solr is clustered, all nodes must be running.

12.1.3.1 Creating collections

To create the Solr collections, use the `create-cores.sh` script. The syntax for this script is as follows:

```
-solr <solr-home> -sh <solr-host> -sp <solr-port> -zh <zookeeper-host>
-zp <zookeeper-port> -zr <zookeeper-root> -shards <numShards> -repl
<replicationFactor> -shardspernode <maxShardsPerNode> -suser <solr-
user> -spwd <solr-pwd> <cor-names ...>
```

The example below assumes that there is a single node, with each collection split into 4 shards. The Zookeeper information (the parameters starting with "`z`") MUST match the value for `ZK_HOST` defined in *Installing and Configuring Solr*.

```
$APIN_HOME/jkool-dbapi-x.y.z/schemas/create-cores.sh -solr
$APIN_HOME/solr-A.B.C -sh localhost -sp 8983 -zh localhost -zp 2181 -zr
/xraysolr -shards 4 -repl 1 -shardspernode 4
```

NOTE

If you are using a Solr cluster, you can enable replication. A minimum of four nodes is recommended if using replication. To enable replication, use:
`-repl 2`

The formula for the maximum number of shards is as follows:

SHARDSPERNODE=$(($NUMSHARDS * $REPLFACTOR / $ClusterSize))

The ClusterSize, REPLFACTOR, and NUMSHARDS variables in this formula are based on the following user prompts:

Enter Number of SOLR Nodes in The Cluster:" ClusterSize
Enter Number of SOLR Replications:" REPLFACTOR
Enter Number of SOLR Shards:" NUMSHARDS

If you need to reload Solr cores, use the `reload-cores.sh` script. The syntax for this script is as follows:

```
-solr <solr-home> -sh <solr-host>-sp <solr-port> -zh <zookeeper-host> -
zp <zookeeper-port> -zr <zookeeper-root> -suser <solr-user> -spwd
<solr-pwd> <cor-names ...>
```

If you need to delete Solr cores, use the `delete-cores.sh` script. The syntax for this script is as follows:

```
-solr <solr-home> -sh <solr-host> -sp <solr-port> -zh <zookeeper-host>
-zp <zookeeper-port> -zr <zookeeper-root> -suser <solr-user> -spwd
<solr-pwd> <cor-names ...>
```


12.1.3.2 Domain Server User Authentication

Domain Server User Authentication is recommended for on-premises installations.

To authenticate the Administrator user when Domain Server user authentication is being used, start the Domain Server now, if is not already running.

Edit the `jkool-cmd.sh` script so that it will use Domain Server user authentication to authenticate the Administrator user.

- To enable domain server authentication, set the following env variable: `DOMAINAUTH=true`
- Set values for the following environmental variables (if their values differ from the defaults below). See the suggested methods below.
  - `DOMAINHOST=localhost`
  - `DOMAINPORT=2323`
  - `DOMAINSERVER=DOMAIN_SERVER`
  - `DOMAINNAME=DOMAIN`

  You can set these values in any of the following ways:

  - Edit the individual scripts that use them.
  - Add them to apin_env.sh.
  - Add them to the profile script (~/.profile) for the system user being used to run the scripts.

12.1.3.3 Loading license

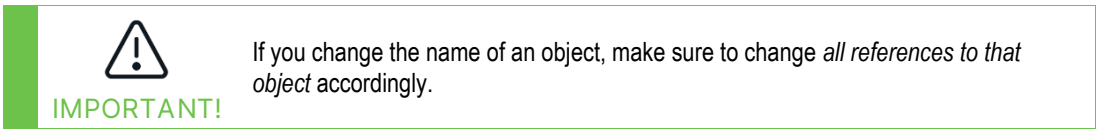
The Support team will provide you with a meshIQ Platform license file. Follow the instructions in *Installing meshIQ platform licenses when upgrading to version 11* or *Updating your expiring meshIQ Platform 11 license*.

12.1.3.4 Loading administration information

To create the default administration records, load the administration definitions as follows:

```
jkool-dbapi-x.y.z/bin/jkool-cmd.sh -run -f:jkool-dbapi-
x.y.z/scripts/xray-admin.jkql -C:localhost:2181/xraysolr -
U:Administrator -P:admin
```

The names of the default administration items can be changed by editing the `xray-admin.jkql` file.



12.1.3.5 Loading reference information

If you want to have XRay convert IP addresses for streamed data to GeoLocations, load the IP-To-GeoLocation mappings, as follows:

```
jkool-dbapi-x.y.z/bin/jkool-cmd.sh -load -f:jkool-dbapi-
x.y.z/scripts/ip2loc.csv -C:localhost:2181/xraysolr -U:Administrator -
P:admin
```

Note that running this command could take as long as 60 minutes.

```
jkool-dbapi-x.y.z/bin/jkool-cmd.sh -load -f:jkool-dbapi-
x.y.z/scripts/feature.csv -C:localhost:2181/xraysolr -U:Administrator -
P:admin
```

12.1.3.6 Loading external data sources


NOTE

Loading the AutoPilot integration external data source configuration requires using Domain Server user authentication. Refer to the *Domain Server User Authentication* subsection of the *Configuring Track* chapter.

To view AutoPilot Policy and Sensor information in the XRay UI, you need to load the AutoPilot integration external data source configuration into XRay.

To load the AutoPilot integration external data source configuration, use the `jkool-load-ext-data-src.sh` script:

```
jkool-dbapi-x.y.z/bin/jkool-load-ext-data-src.sh -f jkool-dbapi-
x.y.z/config/ext-data-src/config-autopilot.xml -pwd admin -solr
$APIN_HOME/solr-A.B.C -sh localhost -sp 8983 -zh localhost -zp 2181 -zr
/xraysolr
```

12.2 Deploying Track Service experts

Now you are ready to start the CEP and deploy the Track Service Experts. There are 13 individual experts that can be deployed, but some are optional, as noted below. To deploy the experts, launch the Core Services Enterprise Manager, and refer to the Deploying Process Wrapper instructions below. The Track experts are under the Data Services submenu. For all of these, you can keep the default property values.

This first group is required:

- Client Service
- Query Service
- Data Streaming Gateway
- Stitching Service
- Compute Service
- DB Writer Service

The remaining ones are optional, depending on the functionality being used:

Table 2. Optional Experts

Functionality	Deploy these Experts
Overall metrics about the status of the data store (Solr)	Metrics Service
jKQL Views	Scheduler Service
Machine Learning	Scheduler Service Script Service Prediction Service
jKQL Scripts	Script Service
Save copies of all records written to data store to the Cold Storage backup	Cold Storage Service

12.2.1 Deploying Experts

Experts can be deployed on CEP servers (including the domain server) within the meshIQ network. Built-in experts can be used immediately after installation. In addition, several plug-ins are available which offer experts that are unique to each managed application.

This section provides you with instructions for the deployment of experts within the meshIQ network. The actual deployment of built-in experts should take just a few seconds in most cases. The following section is a typical example for deploying experts.

12.2.2 Deploying Process Wrapper

To deploy and configure an instance of a process wrapper:

1. Right click the desired CEP server.
2. Select **Deploy Expert > Wrappers > Process Wrapper**.
3. *Configure Process Wrapper* as required for your application. See *meshIQ Platform Core Services Admin and User's Guide* in the meshIQ Resource Center for detailed configuration instructions.

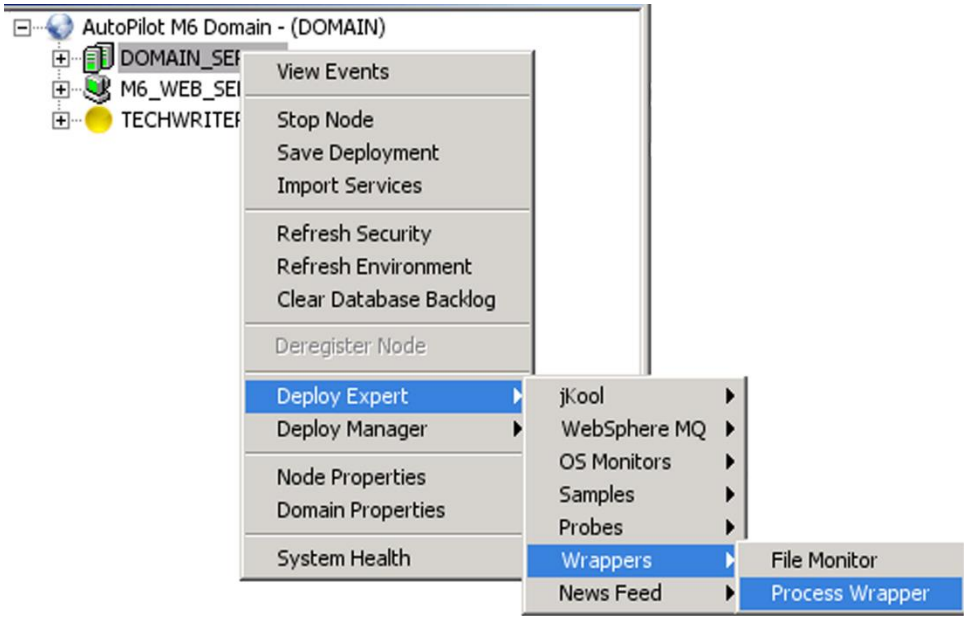


Figure 4-36. Deploying Process Wrapper Experts

NOTE

The system assigned unique name is *Service* and timestamp in milliseconds. The name can be user defined but has to be unique within the M6 domain.

4. Click **Deploy** button to deploy the expert on the CEP server. The *Create Process Wrapper* screen is displayed. The name of the expert being deployed is repeated along with the node where it will be deployed.

   OR

   Click **Deploy On** to deploy on multiple CEP servers. The *Deploy Across Network* screen is displayed. Select the nodes to receive the expert. You can hold the **Ctrl** key and click multiple nodes in the domain to select them. Click **Deploy** to deploy the expert on the selected nodes. When you have deployed, a check mark indicates which nodes have the expert deployed.
5. Right-click on CEP server(s) that had the expert deployed on and click **Save Deployment**.

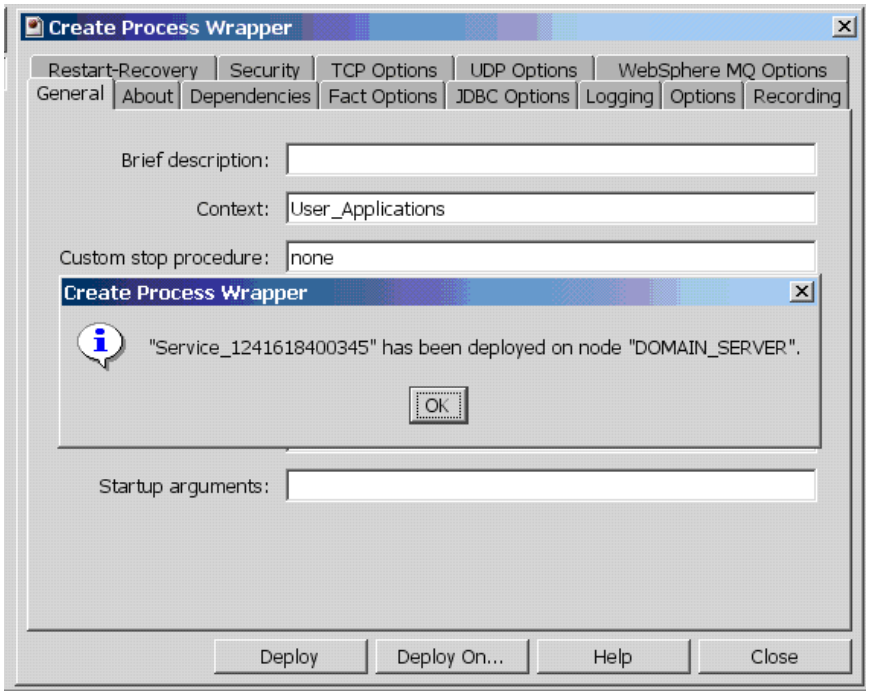


Figure 4-37. Deploying Wrappers

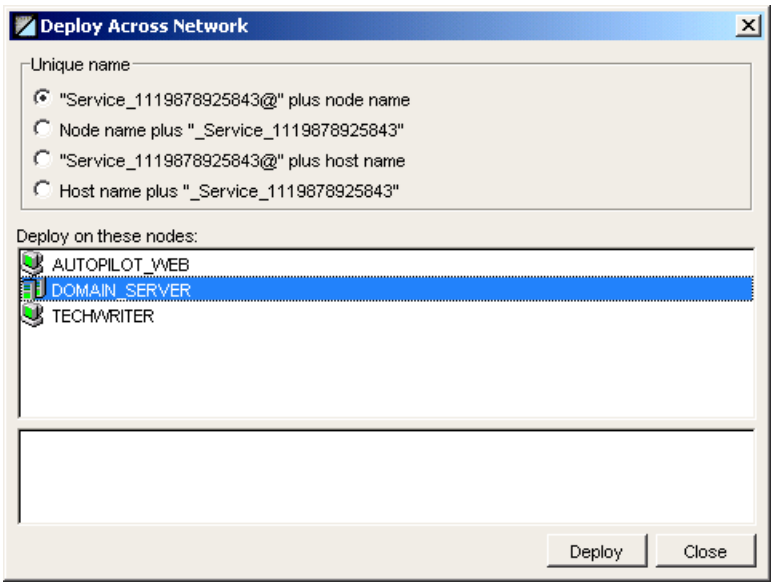


Figure 4-38. Deploy Wrapper on Multiple Nodes

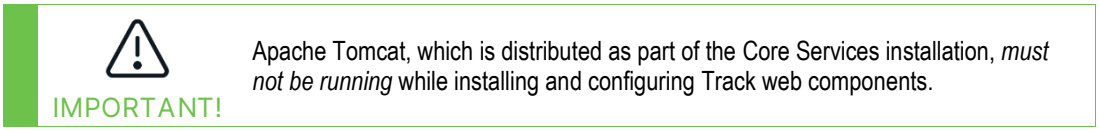
12.2.3 Check Logs and Connections

We recommend that you do the following before proceeding:

- Check the CEP debug logs and verify that there are no errors.
- Check that Track experts have connected to Solr, Kafka, and ActiveMQ.

12.3Installing Web Server

Installing the web server consists of installing and configuring three components: Apache Tomcat, the Track user interface, and the DS-API REST endpoint.



12.3.1 Installing and Configuring Tomcat

This section covers the setup requirements to install Track into an existing Apache Tomcat implementation.

12.3.1.1 Setting up Tomcat properties in Catalina.sh

There are four Tomcat properties that must be set up in Catalina.sh. These properties are used as parameters when the server is started. The table below shows which properties are required by XRay version.

	Purpose	Required 1.4	Required 1.5
-Djkool.service.conn.str ***ActiveMQ Address*** *If ActiveMQ is not local to CEP or Tomcat, this property must be set.*	Allows XRay to read messages from ActiveMQ and execute them.	X	X
-Djkool.stream.url ***Streaming URL*** Typically this URL uses the HTTP or HTTPS protocols. **Ports used:** 6580 or 6585	Allows users to import data from .CSV, .XLS/XLSX (Microsoft® Excel), Apache log, or other custom file formats into XRay using the Import Data wizard.	X	X
-Djkool.administrator.token ***Administrator Root Token***	Used to create an authenticated connection.		X
-Dtnt4j.default.event.factory	Used to configure tnt4j	X	X

tnt4j Logging	log files (also applies to Navigator).		

Add the following settings to `$AUTOPILOT_HOME/apache-tomcat/bin/catalina.sh`:

```
JAVA_OPTS="$JAVA_OPTS -Djkool.stream.url=https://localhost:6580"
JAVA_OPTS="$JAVA_OPTS -Djkclient.message.expiry.msec=0"
JAVA_OPTS="$JAVA_OPTS -
Dtnt4j.default.event.factory=com.jkoolcloud.tnt4j.sink.impl.slf4j.SLF4J
EventSinkFactory"
JAVA_OPTS="$JAVA_OPTS -Dfatpipes.infinispan.config=fatpipes-infinispan-
local.xml"
JAVA_OPTS="$JAVA_OPTS -Djkool.administrator.token=1298eae0-b27c-4175-
9081-aa665c49e653"
```

And, if ActiveMQ is not local to CEP or Tomcat, set the -Djkool.service.conn.str property:

```
JAVA_OPTS="$JAVA_OPTS -Djkool.service.conn.str=123.16.6.210:61616"
```

12.3.1.2 Configuring Tomcat Server.xml for XRay

This section contains examples for deploying the track.war and ds-api.war web applications by configuring the Apache Tomcat server.xml file using <Context> XML elements. Additional changes are required for configuring the Apache Tomcat server.xml file for Navigator. See *Installing XRay, Navigator, and Security Manager into Apache Tomcat* in the Resource Center for details.

12.3.1.2.1 Context Element Example: track.war

The following example shows how to configure the Apache Tomcat server.xml file using the <Context> XML element for track.war.


NOTE

The Resource name="jms/FPConnectionFactory" XML child element shown in **boldface** below is an example configuration for ActiveMQ broker URI using two-node cluster static transport method.

Refer to the ActiveMQ documentation at *https://activemq.apache.org/failover-transport-reference.*

```
<Context path="/track"
                reloadable="true"
                docBase="track.war">

        <!-- Client is using ActiveMQ to communicate w/ Data Services Client Service -->
        <!-- The brokerURL host name or IP address should be that of the Data Services Client
Service (which is hosting ActiveMQ broker)

static:(tcp://localhost:61616,tcp://remotehost:61617?trace=false,vm://localbroker)?initialReconn
ectDelay=100 -->
        <Resource name="jms/FPConnectionFactory"
                auth="Container"
                type="org.apache.activemq.ActiveMQConnectionFactory"
```

factory="org.apache.activemq.jndi.JNDIReferenceFactory"

brokerURL="failover:(tcp://<activeMQBRK1>:61616,tcp://<activeMQBRK2>:61616)?randomize=false&timeout=10000&nested.connectionTimeout=120000&nested.wireFormat.maxInactivityDuration=300000&nested.wireFormat.maxInactivityDurationInitalDelay=60000"/>

```
        <!-- ActiveMQ queue on which to send requests - physicalName must match definition on Data Services Client Service -->
        <Resource name="jms/queue/client-requests"
                auth="Container"
                type="org.apache.activemq.command.ActiveMQQueue"
                factory="org.apache.activemq.jndi.JNDIReferenceFactory"
                physicalName="jkool.service.requests"/>

        <!-- ActiveMQ queue on which to send admin requests - physicalName must match definition on Data Services Client Service -->
         <Resource name="jms/queue/admin-requests"
                auth="Container"
                type="org.apache.activemq.command.ActiveMQQueue"
                factory="org.apache.activemq.jndi.JNDIReferenceFactory"
                physicalName="jkool.service.admin.requests"/>

        <!-- ActiveMQ queue on which to send update requests - physicalName must match definition on Data Services Client Service -->
        <Resource name="jms/queue/update-requests"
                auth="Container"
                type="org.apache.activemq.command.ActiveMQQueue"
                factory="org.apache.activemq.jndi.JNDIReferenceFactory"
                physicalName="jkool.service.update.requests"/>

        <!-- ActiveMQ topic on which to send status requests - physicalName must match definition on Data Services Client Service -->
         <Resource name="jms/topic/status-requests"
                auth="Container"
                type="org.apache.activemq.command.ActiveMQTopic"
                factory="org.apache.activemq.jndi.JNDIReferenceFactory"
                physicalName="jkool.service.status.requests"/>
    </Context>
```

12.3.1.2.2Context Element Example: ds-api.war

The following example shows how to configure the Apache Tomcat server.xml file using the <Context> XML Element for ds-api.war.

IMPORTANT!

Do not modify the <Resource> child elements shown in the following example.

Before adding other transport options, be sure to perform testing in which you stop one of the ActiveMQ brokers and determine whether Track works seamlessly with the other broker.

```
<Context path="/ds-api"
 reloadable="true"
 docBase="ds-api.war">

 <!-- Client is using ActiveMQ to communicate w/ Data Services Client Service -->
 <!-- The brokerURL host name or IP address should be that of the Data Services Client Service (which is hosting ActiveMQ broker) -->
 <Resource name="jms/FPConnectionFactory"
   auth="Container"
   type="org.apache.activemq.ActiveMQConnectionFactory"
   factory="org.apache.activemq.jndi.JNDIReferenceFactory"
   brokerURL="failover:(tcp://<activeMQBRK1>:61616,tcp://<activeMQBRK2>:61616)?randomize=false&timeout=10000&nested.connectionTimeout=120000&nested.wireFormat.maxInactivityDuration=300000&nested.wireFormat.maxInactivityDurationInitalDelay=60000"/>

 <!-- ActiveMQ queue on which to send requests - physicalName must match definition on Data Services Client Service -->
 <Resource name="jms/queue/client-requests"
  auth="Container"
  type="org.apache.activemq.command.ActiveMQQueue"
  factory="org.apache.activemq.jndi.JNDIReferenceFactory"
  physicalName="jkool.service.requests"/>

 <!-- ActiveMQ queue on which to send admin requests - physicalName must match definition on Data Services Client Service -->
   <Resource name="jms/queue/admin-requests"
  auth="Container"
  type="org.apache.activemq.command.ActiveMQQueue"
  factory="org.apache.activemq.jndi.JNDIReferenceFactory"
  physicalName="jkool.service.admin.requests"/>

 <!-- ActiveMQ queue on which to send update requests - physicalName must match definition on Data Services Client Service -->
 <Resource name="jms/queue/update-requests"
  auth="Container"
  type="org.apache.activemq.command.ActiveMQQueue"
  factory="org.apache.activemq.jndi.JNDIReferenceFactory"
  physicalName="jkool.service.update.requests"/>

 <!-- ActiveMQ topic on which to send status requests - physicalName must match definition on Data Services Client Service -->
   <Resource name="jms/topic/status-requests"
  auth="Container"
```

```
  type="org.apache.activemq.command.ActiveMQTopic"
  factory="org.apache.activemq.jndi.JNDIReferenceFactory"
  physicalName="jkool.service.status.requests"/>
</Context>
```

12.3.1.2.3Context Element Example: Connector settings

For Track and Manage, specific <Connector> element settings are expected in the server.xml file.


NOTE

- The default port assignments of 8080 and 8443 can be configured based on your organization's standards.
- If you enforce SSL security, then JKS certificates need to be created and the path to the JKS certificate file needs to be provided in the XML attribute keystoreFile=<path to JKS certificate file>/<your.jks>.

To set up connector settings, uncomment the SSL connection for HTTPS connections (remove the bold text below). Then provide the proper SSL entries, as shown below.

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />

<!-- TO Enable SSL connection: -->

 <!--
   <Connector
         protocol="org.apache.coyote.http11.Http11NioProtocol"
         port="8443" maxThreads="200"
         scheme="https" secure="true" SSLEnabled="true"
         keystoreFile="/opt/meshiq/platform/ssl/your.jks" keystorePass="password"
         clientAuth="false" SSLProtocol="TLSv1.2"
    />
  -->
```

12.3.1.3 (Optional) Reviewing Context.xml

After track.war has been extracted, you can find context.xml located at \track\META-INF\context.xml. This context file includes properties for cache settings and the location of the SAML SSO configuration file. Additional Single Sign-On setup is required. For assistance in setting up Single Sign-On, contact meshIQ support at *support@meshiq.com*.

1. Place the context.xml file in the $CATALINA_HOME/conf directory. In this location, it will take precedence over the context.xml located in the track.war file.
2. Review the cachingAllowed and cacheMaxSize settings in this file. An example is provided below.
3. If you are implementing Single Sign-On, follow the steps below. Otherwise, no changes to the xray.samlsso.manager.config property are required.
4. Place the SSO configuration file (xray_samlsso.xml) in the expected system location, which is the Tomcat config directory. For example:

   $CATALINA_HOME/conf/xray_samlsso.xml

5. Identify the SSO Configuration file in the context.xml file.

a. In the $CATALINA_HOME/conf directory, Edit the file using the local text editor.
b. Add the following lines into the context.xml file within the XML element <Context>:

<!--samlsso configuration file -->

<Parameter name="xray.samlsso.manager.config"
value="${CATALINA_HOME}/conf/xray_samlsso.xml"/>

12.3.1.3.1Context.xml Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
See the NOTICE file distributed with this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use
this file except in compliance with the License. You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0.
Unless required by applicable law or agreed to in writing, software distributed under the License is
distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied. See the License for the specific language governing permissions and limitations under
the License.
-->
<!-- The contents of this file will be loaded for each web application -->
<Context>
<!-- Default set of monitored resources. If one of these changes, the -->
<!-- web application will be reloaded. -->
<WatchedResource>WEB-INF/web.xml</WatchedResource>
<WatchedResource>WEB-INF/tomcat-web.xml</WatchedResource>
<WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>
<!--samlsso configuration file -->
<Parameter name="apwmq.samlsso.manager.config"
value="${CATALINA_HOME}/conf/apwmq_samlsso.xml"/>
<Parameter name="xray.samlsso.manager.config"
value="${CATALINA_HOME}/conf/xray_samlsso.xml "/>
<!-- Uncomment this to disable session persistence across Tomcat restarts -->
<!--
<Manager pathname="" />
-->
<Resources
cachingAllowed="true"
cacheMaxSize="100000"
/>
</Context>
```

12.3.2 Installing and Configuring the Track User Interface

To install the Track user interface, do the following:

1. Download the appropriate distribution file. You must choose the version of the file that matches that of the DATA_SERVICES Plug-in you installed (see *Installing the DATA_SERVICE Plug-in*).

- For Track 11, the file name is: `track-ui-11.x.x[.r]`

2. Untar the distribution and put the track.war file in `$AUTOPILOT_HOME/apache-tomcat/webapps`
   The app server will create the \track directory structure.

To configure the Track user interface, do the following:

3. Add the code below to `$AUTOPILOT_HOME/apache-tomcat/conf/server.xml` (within the `<Host>` element). Be sure to make the following changes if needed:

   - The value for `brokerURL` must match the host and port that ActiveMQ is running on. (By default, this sample code presents ActiveMQ running on a local machine using the default port.)

```
      <Context path="/track"
                      reloadable="true"
                      docBase="track.war">

           <!-- Client is using ActiveMQ to communicate w/ Data
Services Client Service -->
           <!-- The brokerURL host name or IP address should be
that of the Data Services Client Service (which is hosting
ActiveMQ broker) -->
           <Resource name="jms/FPConnectionFactory"
                     auth="Container"
                     type="org.apache.activemq.ActiveMQConnection
Factory"
                     factory="org.apache.activemq.jndi.JNDIRefere
nceFactory"
                     brokerURL="tcp://localhost:61616?connectionT
imeout=120000&amp;wireFormat.maxInactivityDuration=300000&amp;wir
eFormat.maxInactivityDurationInitalDelay=60000"/>

           <!-- ActiveMQ queue on which to send requests -
physicalName must match definition on Data Services Client
Service -->
           <Resource name="jms/queue/client-requests"
                    auth="Container"
                    type="org.apache.activemq.command.ActiveMQQue
ue"
                    factory="org.apache.activemq.jndi.JNDIReferen
ceFactory"
                    physicalName="jkool.service.requests"/>

           <!-- ActiveMQ queue on which to send admin requests -
physicalName must match definition on Data Services Client
Service -->
             <Resource name="jms/queue/admin-requests"
                    auth="Container"
                    type="org.apache.activemq.command.ActiveMQQue
ue"
                    factory="org.apache.activemq.jndi.JNDIReferen
ceFactory"
                    physicalName="jkool.service.admin.requests"/>
```

```
            <!-- ActiveMQ queue on which to send update requests -
physicalName must match definition on Data Services Client
Service -->
            <Resource name="jms/queue/update-requests"
                      auth="Container"
                      type="org.apache.activemq.command.ActiveMQQue
ue"
                      factory="org.apache.activemq.jndi.JNDIReferen
ceFactory"
                      physicalName="jkool.service.update.requests"/
>

            <!-- ActiveMQ topic on which to send status requests -
physicalName must match definition on Data Services Client
Service -->
              <Resource name="jms/topic/status-requests"
                      auth="Container"
                      type="org.apache.activemq.command.ActiveMQTop
ic"
                      factory="org.apache.activemq.jndi.JNDIReferen
ceFactory"
                      physicalName="jkool.service.status.requests"/
>
        </Context>
```

12.3.3 Installing and Configuring the Track REST endpoint

The DS-API REST Service is required if you plan to issue jKQL requests using the REST API.

To install the DS-API REST Service, do the following:

1. Download the appropriate distribution file: `DATA_SERVICE-x.y.z[.r]`. You must choose the version of the file that matches that of the DATA_SERVICE Plug-in you installed (see *Installing the DATA_SERVICE Plug-in*).
2. Untar the distribution and put the war file in `$AUTOPILOT_HOME/apache-tomcat/webapps`.

To configure the DS-API REST Service, do the following:

1. Add the following code to `$AUTOPILOT_HOME/apache-tomcat/conf/server.xml` (within the `<Host>` element).

```
        <Context path="/ds-api"
                        reloadable="true"
                        docBase="ds-api.war">

            <!-- Client is using ActiveMQ to communicate w/ Data
Services Client Service -->
            <!-- The brokerURL host name or IP address should be
that of the Data Services Client Service (which is hosting
ActiveMQ broker) -->
            <Resource name="jms/FPConnectionFactory"
                      auth="Container"
                      type="org.apache.activemq.ActiveMQConnection
Factory"
```

```
                        factory="org.apache.activemq.jndi.JNDIRefere
nceFactory"
                        brokerURL="tcp://localhost:61616?connectionT
imeout=120000&amp;wireFormat.maxInactivityDuration=300000&amp;wir
eFormat.maxInactivityDurationInitalDelay=60000"/>

          <!-- ActiveMQ queue on which to send requests -
physicalName must match definition on Data Services Client
Service -->
          <Resource name="jms/queue/client-requests"
                  auth="Container"
                  type="org.apache.activemq.command.ActiveMQQue
ue"
                  factory="org.apache.activemq.jndi.JNDIReferen
ceFactory"
                  physicalName="jkool.service.requests"/>

          <!-- ActiveMQ queue on which to send admin requests -
physicalName must match definition on Data Services Client
Service -->
            <Resource name="jms/queue/admin-requests"
                  auth="Container"
                  type="org.apache.activemq.command.ActiveMQQue
ue"
                  factory="org.apache.activemq.jndi.JNDIReferen
ceFactory"
                  physicalName="jkool.service.admin.requests"/>

          <!-- ActiveMQ queue on which to send update requests -
physicalName must match definition on Data Services Client
Service -->
          <Resource name="jms/queue/update-requests"
                  auth="Container"
                  type="org.apache.activemq.command.ActiveMQQue
ue"
                  factory="org.apache.activemq.jndi.JNDIReferen
ceFactory"
                  physicalName="jkool.service.update.requests"/
>

          <!-- ActiveMQ topic on which to send status requests -
physicalName must match definition on Data Services Client
Service -->
            <Resource name="jms/topic/status-requests"
                  auth="Container"
                  type="org.apache.activemq.command.ActiveMQTop
ic"
                  factory="org.apache.activemq.jndi.JNDIReferen
ceFactory"
                  physicalName="jkool.service.status.requests"/
>
        </Context>
```

2. If needed, update the value for `brokerURL` to match the host and port that ActiveMQ is running on. (By default, this sample code presents ActiveMQ running on a local machine using the default port.)