meshIQ

XRay

Custom Installation Guide

Version 1.5

Document Number: XRCIG100.001

Document Title: XRay Custom Installation Guide
Document Number: XRCIG100.001
Document Release Date: March 2023

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

Table of Contents

Chapter 1: Introduction

The recommended methods of installing XRay 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 XRay components within an existing installation. It covers the manual equivalents of the installation and configuration scripts that are automated through the XRay deployment pack. It contains instructions for the minimum steps required to install, configure, and run XRay.

1.1 Recommendations

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

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

```
APIN_HOME=/opt/nastel
```

This should be done in the appropriate `.profile` file for the user (such as `nastel`) that will be used to run all XRay 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: How to install and configure Storm.

Chapter 9: Instructions for starting the installed components (Zookeeper, ActiveMQ, Solr, Kafka, and Storm).

Chapter 10: How to install AutoPilot.

Chapter 11: Instructions for downloading and installing the XRay Server Plugin package and required dependencies.

Chapter 12: Instructions for installing the XRay DBAPI Utility for Solr.

Chapter 13: Covers the configuration of XRay, 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.

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 status of any proposed change, update, or correction.

1.5 Release Notes

Refer to the *XRay Release Notes page* 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 XRay 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 XRay 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 XRay. 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, CentOS and other Linux distributions have an active firewall that block ports which are needed to connect to XRay services remotely. The following ports should be opened, as these are the ports XRay utilizes:

ZooKeeper: 2181
Solr: 8983
ActiveMQ: 8161
Storm: 8088
AutoPilot Domain Server: 2323, 3000
AutoPilot XRay CEP: 3005
AutoPilot XRay UI CEP : 3010
XRay: 8080
XRay Gateway: 6580

The following commands can be used to check the firewall status, temporarily stop/disable the firewall, or re-enable/re-start the default firewall on CentOS 7.

```
>systemctl stop firewalld
>systemctl disable firewalld
>systemctl enable firewalld
```

2.3 Modify limits.conf (Required)

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

```
nastel           soft    nofile   65536
nastel           hard    nofile   65536

nastel           soft    nproc    65536
nastel           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 XRay services:

- export APIN_HOME=<path_to_XRay_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)

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

3.1 Oracle Java 8 JRE (XRay version 1.4)

To install Oracle Java 8 JRE, do the following:

1. Download the tar image distribution file for the *latest version* of Oracle Java JRE 1.8.
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 `nastel`) that will be used to run all XRay processes.

   `JAVA_HOME=$APIN_HOME/jre1.8.0_xxx`

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

3.2 OpenJDK 11 (XRay version 1.5)

NOTE

OpenJDK 11 is distributed with XRay version 1.5. However, it is not required until XRay version 1.6.

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 `nastel`) that will be used to run all XRay processes.

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

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

Chapter 4: Installing and Configuring Zookeeper

The version of Zookeeper to install depends on the version of XRay you are using.

4.1 Zookeeper 3.4.13 (XRay versions 1.4 and earlier - Solr 6)

To install Zookeeper 3.4.13, do the following:

1. Download the tar image distribution file for Apache Zookeeper 3.4.13.
2. Untar the distribution in `$APIN_HOME`.

To configure Zookeeper 3.4.13, 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
     ```

4.2 Zookeeper 3.6.2 (XRay versions 1.5 and later - Solr 8)

Follow the instructions above for Zookeeper 3.4.13. Then add the following lines to the end of `zoo.cfg`:

```
# web admin port (default is 8080, which conflicts with XRay 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

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 6.6.6 (XRay versions 1.4 and earlier)

To install Solr 6.6.6, do the following:

1. Download the tar image distribution file for Apache Solr 6.6.6.
2. Untar the distribution in `$APIN_HOME`.

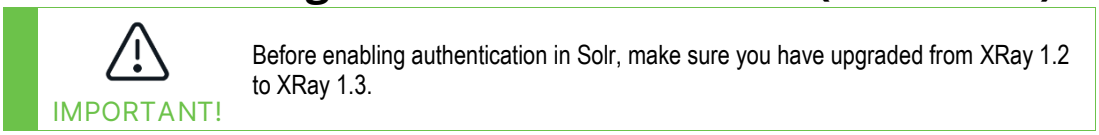
To configure Solr 6.6.6, do the following:

1. Create folder: `$APIN_HOME/solr`.
2. Copy the following file from solr-6.6.6/server/solr to $APIN_HOME/solr/var/data:
   - `solr.xml`
3. Copy the following file from `solr-6.6.6/server/resources` to `$APIN_HOME/solr/var`:
   - `log4j.properties`
4. 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"
LOG4J_PROPS="$APIN_HOME/solr/var/log4j.properties"
SOLR_LOGS_DIR="$APIN_HOME/solr/var/logs"
SOLR_PORT="8983"
SOLR_MODE="solrcloud"
SOLR_JAVA_HOME=$JAVA_HOME
SOLR_JAVA_MEM="-Xmx12g -Xms12g"
```

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

5.1.1 Enabling authentication in Solr (Advanced)



XRay versions 1.3 and 1.4 require that all Solr clusters user the same user and password. In XRay version 1.5, the clusters are able to use different credentials.

Follow these steps to enable Solr authentication with the default user name and password.

1. Leave Zookeeper running.
2. Stop all Solr nodes.
3. Verify that your current directory is jkool-dbapi-solr-1.3.30 and zkchroot of "/".

4. To enable authentication and define the default user "solr" with password "SolrRocks", upload the security.json file into the appropriate zkchroot within Zookeeper, using the following:

   ```
   <solr-home>/bin/solr zk cp file:config/security.json zk:/security.json
   -z <zookeeper-ip>:<zookeeper port>
   ```

5. Uncomment the following lines in <solr-home>/bin/solr.in.sh:

   ```
   SOLR_AUTH_TYPE="basic"
   SOLR_AUTHENTICATION_OPTS="-Dbasicauth=solr:SolrRocks"
   ```

6. Restart Solr nodes.

   Solr should now be running with authentication required, with user "solr" and password "SolrRocks". ALL connections to Solr will now require that credentials be specified.

5.1.1.1 Password Changes

5.1.1.1.1 Changing the default password

To change the default password, you can use the following command.

```
curl --user
solr:SolrRocks http://localhost:8983/solr/admin/authentication -H
'Content-type:application/json' -d '{"set-user": {"solr" : "the-new-
password"}}'
```

See the "Securing Solr" section of the Solr Reference Guide (*https://solr.apache.org/guide/*) for additional commands and information.

5.1.1.1.2 After changing the password

If you have changed the password, then you must follow the steps in the “Enabling authentication in Solr” section above again, changing the references from "SolrRocks" to the new password. (In the Solr files, this is plain text.)

You must also run apnet encrypt to get the encrypted value for the new password to use in global.properties.

Before starting Xray (CEP), add the following to global.properties (to add the encrypted form of the new password value from apnet):

```
property jkool.db.server.user=solr
property_encrypted jkool.db.server.pwd=[encrypted new password]
```

5.1.1.1.3 Effects of reloading security.json

If you reload security.json, the password will be reset back to the default of "SolrRocks". Follow the steps below to address this change.

Before starting Xray (CEP), add the following to global.properties (to add the "SolrRocks" password value in encrypted form from apnet):

```
property jkool.db.server.user=solr
property_encrypted jkool.db.server.pwd=+DSP8b8Q6TafTstcll+IVw==
```

5.1.1.2 Command-Line Utilities Require Credentials for Solr Connection

The following command-line utilities require Solr connection information:

- create-cores.sh
- delete-cores.sh
- jkool-cmd.sh
- jkool-db-maint.sh
- jkool-db-upgrade.sh
- jkool-load-ext-data-src.sh
- reload-cores.sh

When running the command line script provided with Xray, you must include the Solr credentials on the command line as arguments. Run the command with "-help" to get the syntax for specifying credentials.

For example:

`-UD:<solr_user> -PD:<solr_pwd>`

5.2 Solr 8.11.1 (XRay versions 1.5 and later)

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 5.14.5 is used in XRay versions through 1.5. Instructions for installing and configuring brokers in an ActiveMQ two-node cluster using the static transport method are also provided.

6.1 ActiveMQ 5.14.5 (XRay versions 1.5 and earlier)

To install ActiveMQ Classic 5.14.5, do the following:

1. Download the tar image distribution file for ActiveMQ Classic 5.14.5.
2. Untar the distribution in `$APIN_HOME`.

To configure ActiveMQ Classic 5.14.5, edit the `activemq.xml` file in the `conf` subfolder within the ActiveMQ distribution as follows:

- Change the `<broker>` element as follows:
  - Change the `brokerName` attribute to `"JKMessageBus"`
  - Add this new attribute: `schedulePeriodForDestinationPurge="300000"`

  After these changes, the `<broker>` element reads as follows:

  - `<broker xmlns="http://activemq.apache.org/schema/core" brokerName="JKMessageBus" dataDirectory="${activemq.data}" schedulePeriodForDestinationPurge="300000">`
- Add the following `<policyEntry>` element within `<policyEntries>` under `<broker>` (after `<policyEntry topic=">" >`):

```
<!-- Delete jKool client result queues that are no longer used
(inactive for 2 minutes) -->

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

6.2 Installing and Configuring Brokers in an ActiveMQ Two-Node Cluster using the Static Transport Method (XRay version 1.5)

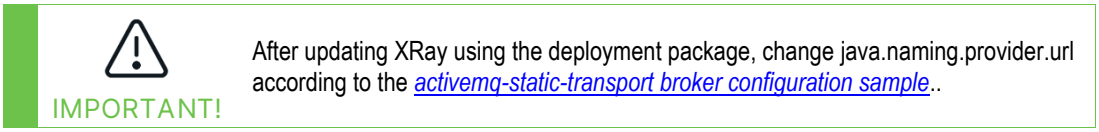
Using the XRay 1.5 deployment package, install the ActiveMQ component on two cluster nodes. Then Follow the steps below:

1. Install ActiveMQ.
2. Configure ActiveMQ for JMS by modifying the jndi.properties, Apache tomcat server.xml, and activemq.xml files, as described below.
   a. Make the changes below on the Client Processing Group XRay CEP.


NOTE

You may have to apply the same changes to other CEP nodes.

i. Refer to the sample jndi.properties (for ActiveMQ broker cluster config) file in the appendix to make changes in the jndi.properties file located in AutoPilotM6/jkool/config/jndi.properties. (See also the *activemq-static-transport broker configuration sample*.)



ii. Refer to the sample server.xml (for ActiveMQ broker cluster config) file in the appendix to make changes in Apache tomcat server.xml. (See also the *activemq-static-transport broker configuration sample*.)

iii. For the two-node ActiveMQ cluster configuration, add the following in the activemq.xml <networkConnectors> XML Element for only one of the brokers— *either broker1 or broker2, but not both*. And host/ipad is the host/ipad of the other broker.

```
<networkConnectors>
        <networkConnector uri="static:(tcp://<activeMQBRK1>:61616)" duplex="true"/>
</networkConnectors>
```

Refer to the activemq.xml (config on brk1) and activemq.xml (config on brk2) files in the appendix.

Chapter 7: Installing and Configuring Kafka

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

7.1 Kafka 2.X (XRay versions 1.5 and earlier)

To install Kafka 2.8.0, do the following

1. Download the tar image distribution file for Kafka 2.8.0 . (The Kafka 2.8.0 client is included in AutoPilot.)
2. Untar the distribution in `$APIN_HOME`.

To configure Kafka 2.8.0, edit `server.properties` in `config` subfolder within the Kafka distribution:

1. Change the following properties:
   - `num.network.threads` from `3` to `8`
   - `num.io.threads` from `8` to `16`
   - `log.dirs` (if you do not want to use the default folder)
   - `num.partitions` from `1` to `16`
   - `log.retention.hours` from `168` to `1`
   - `zookeeper.connect` from default value to `localhost:2181/xraykafka`
2. Add the following properties:
   - `message.max.bytes=10485760`
   - `replica.fetch.max.bytes=10485760`

Chapter 8: Installing and Configuring Storm

The version of Storm to install depends on the version of XRay you are using.

8.1 Apache Storm 1.1.3 (XRay versions 1.4 and earlier)

To install Apache Storm 1.1.3, do the following:

1. Download the tar image distribution file for Apache Storm 1.1.3.
2. Untar the distribution in `$APIN_HOME`.

To configure Apache Storm 1.1.3, edit `storm.yaml` in the `conf` subfolder within the Storm distribution by adding the following lines:

```
storm.local.dir: "storm-local"
storm.log4j2.conf.dir: "log4j2"

storm.zookeeper.servers:
    - "localhost"
storm.zookeeper.port: 2181
storm.zookeeper.root: "/xraystorm"

nimbus.seeds: ["localhost"]

supervisor.slots.ports:
    - 6700
    - 6701
    - 6702
    - 6703

ui.port: 8088

worker.childopts: "-Xmx2g -Xms1g"
```

8.2 Apache Storm 2.4.0 (XRay versions 1.5 and later)

To install Apache Storm 2.4.0, do the following:

1. Download the tar image distribution file for Apache Storm 2.4.0.
2. Untar the distribution in `$APIN_HOME`.

To configure Apache Storm 2.4.0, edit `storm.yaml` in the `conf` subfolder within the Storm distribution as described above in the Apache Storm 1.1.3 subsection.

Chapter 9: Start Components

Before continuing with configuring XRay, 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.

9.1 Start Zookeeper

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

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

./bin/zkServer.sh start
```


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]`

9.2 Starting ActiveMQ

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

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

./bin/ activemq start
```

9.3 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
```

9.4 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

nohup ./bin/kafka-server-start.sh config/server.properties >/dev/null
2>&1 &
```

9.5 Starting Storm

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

```
cd $APIN_HOME/apache-storm-A.B.C
```

Run the following at the command prompt:

```
nohup bin/storm nimbus &
nohup bin/storm supervisor &
nohup bin/storm ui &
```

Chapter 10: Installing AutoPilot

Download and install the appropriate version of AutoPilot, based on the version of XRay being installed Refer to the *AutoPilot M6 Documentation Library* for detailed instructions on how to install AutoPilot and plug-ins. When a given version of XRay requires a specific AutoPilot Service Update (SU) version or other dependency, that requirement is noted here. Also, in this document it is assumed that the environment variable AUTOPILOT_HOME points to the root folder for AutoPilot M6.

10.1 Installing and Configuring the Job Scheduler Plug-in

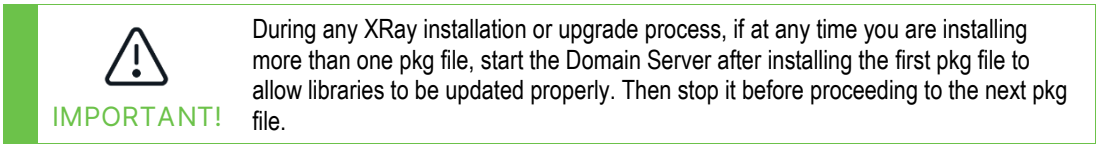
Refer to the *AutoPilot® M6 Plug-in for Job Scheduler Expert Installation Guide* for information on how to install and configure the Job Scheduler Plug-in.

Chapter 11: Installing the XRay JKOOL_SERVICE Plug-in

XRay Server has the following dependencies:

- Job Scheduler. (The version of Job Scheduler that is required depends on the version of XRay being installed.)
- WGS REST Plug-in (starting with JKOOL_SERVICE-1.5.10)

1. Download the appropriate XRay Server Plugin package (along with the required dependencies):
    - JKOOL_SERVICE-1.4.19[.r].pkg (for SU32)
    - JKOOL_SERVICE-1.4.z.pkg (where z >= 20, for SU33+)
    - JKOOL_SERVICE-1.5.z.pkg (for SU33+)
2. Using pkgman, install the pkg files for your version of XRay. See the important note below.



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

Chapter 12: Installing the XRay DBAPI Utility

Since the utilities provided require scripts that are included in Solr, the XRay DBAPI utility should be installed on one of the Solr servers.

1. Download the tar image distribution file that matches the JKOOL_SERVICE version you installed in the previous chapter:
   - jkool-dbapi-solr-1.4.x.tar.gz
   - jkool-dbapi-solr-1.5.x.tar.gz
2. Untar the distribution in $APIN_HOME.

Chapter 13: Configuring XRay

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

13.1Configuring XRay Servers

XRay server configuration includes setting properties and preparing the components for use by creating Kafka topics, initializing the Solr database, and loading Storm topologies.

13.1.1 System properties

Properties files exist at both the global and node levels.

13.1.1.1 Global

The JKOOL_SERVICE pkg will define properties in the global.properties file in the AutoPilotM6 folder. 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 AutoPilot Domain Server user authentication instead of the default XRay user authentication, you must define the value above for the `property jkool.auth.service.class` property.

13.1.1.2 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 java.awt.headless=true
property jkool.install.dir={autopilot.home}/jkool/
property fatpipes.jndi.file={jkool.install.dir}config/jndi.properties
property fatpipes.data.dir=.


; Defines Netty's automatic ByteBuf leak detection level - Don't define in production
;property io.netty.leakDetection.level=advanced

;property fatpipes.infinispan.stats.enable=true
;property
jkool.fatpipes.cache.register=com.nastel.fatpipes4j.impl.infinispan.SpanningCacheRegister
property jkool.ml.ts.defaults.file={jkool.install.dir}config/mlmodel-defaults.json
property JOB_SCHED=Job_Scheduler
property java.net.preferIPv4Stack=true
```

13.1.2 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
```

In XRay 1.4, this script connects to Zookeeper, so you must specify the Zookeeper connection information. This *must* match the value of `zookeeper.connect` defined earlier in the Installing and Configuring Kafka chapter. For example:

```
create-kafka-topics.sh -k $APIN_HOME/kafka_A.B-x.y.z -zh localhost -zp
2181 -zr /xraykafka
```

In XRay 1.5, this script connects directly to the Kafka broker, so you must specify Kafka connection information. In the example below, Kafka is running locally on the default port:

```
create-kafka-topics.sh -k $APIN_HOME/kafka_A.B-x.y.z -kh localhost -kp
9092
```

13.1.3 Initializing Solr

Solr is initialized using utilities provided in the XRay DBAPI Utility. As mentioned above in Installing the XRay DBAPI 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.

13.1.3.1 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 ...>
```

13.1.3.2 User Authentication

13.1.3.2.1XRay User Authentication

If you are using built-in XRay user authentication, you need to define the Administrator user.

To define the Administrator user, run the following:

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

13.1.3.2.2Domain Server User Authentication

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. The jkool.auth.service.class property must be set.

- Find the comment that mentions “using External Authentication module”, and uncomment out the next two lines, which define Domain Server information. If the Domain Server is not running locally or is not on the default port, edit the values as needed.
  Example:

  jkool.auth.service.class=com.nastel.jkool.auth.DomainAuthService

13.1.3.3 Loading license

The Support team will provide you with an XRay license file.

Load the XRay license using `jkool-cmd.sh` as follows:

```
jkool-dbapi-x.y.z/bin/jkool-cmd.sh - loadlic -f:<path-to-lic-file> -
C:localhost:2181/xraysolr -U:Administrator -P:admin
```

To load the supported feature set, use `jkool-cmd.sh` as follows:

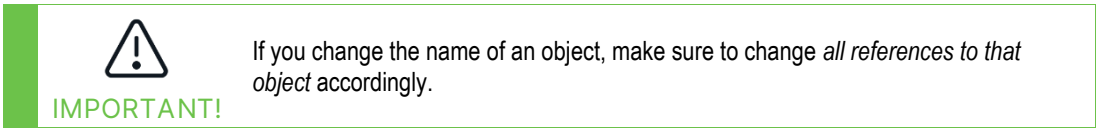
```
jkool-dbapi-x.y.z/bin/jkool-cmd.sh -load -f:jkool-dbapi-
x.y.z/scripts/feature.csv -C:localhost:2181/xraysolr -U:Administrator -
P:admin
```

13.1.3.4 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.



13.1.3.5 Loading reference information

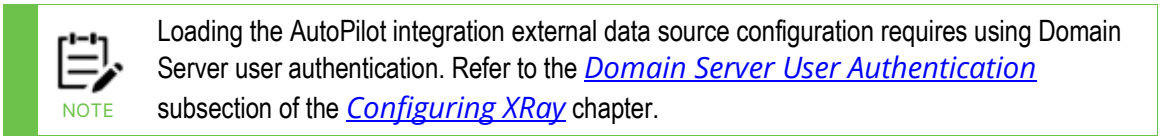
If you want to have XRay convert IP addresses for streamed data to GeoLocations, load the IP-To-GeoLocation mappings, as follows:

```
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
```

13.1.3.6 Loading external data sources



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
```

13.1.4 Loading Storm topologies (includes Subscription and Trigger topologies)

If you are using XRay Real-time Subscriptions or Triggers (Alerts), you need to upload the XRay Storm topologies to Storm.

Since the script for loading Storm topologies uses scripts provided with the Storm distribution, Storm is required in addition to the topologies.

Before uploading the topologies, start Storm and Zookeeper, if they are not already running. • The files required for this are located under `$AUTOPILOT_HOME/jkool`. The upload script makes use of the jars in the `lib` folder.

To upload Storm topologies, do the following:

1. Run the bin/start-storm-topology.sh script.
2. Switch back to `$AUTOPILOT_HOME/jkool` and run the following:

```
bin/start-storm-topology.sh $APIN_HOME/apache-storm-A.B.C
```

13.2 Deploying XRay Service experts

Now you are ready to start the CEP and deploy the XRay Service Experts. There are 13 individual experts that can be deployed, but some are optional, as noted below. To deploy the experts, launch the AutoPilot Enterprise Manager, and refer to the Deploying Process Wrapper instructions below. The XRay experts are under the jKool 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 1. 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

Table 1. Optional Experts	
Functionality	**Deploy these Experts**
Save copies of all records written to data store to the Cold Storage backup	Cold Storage Service
Real-time subscriptions	Subscription Service
XRay Triggers (alerts)	Trigger Service

13.2.1 Deploying Experts

Experts can be deployed on CEP servers (including the domain server) within the M6 Network. M6 is supplied with built-in experts that 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 M6 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.

13.2.2 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 the *AutoPilot® M6 User's Guide* for detailed configuration instructions.

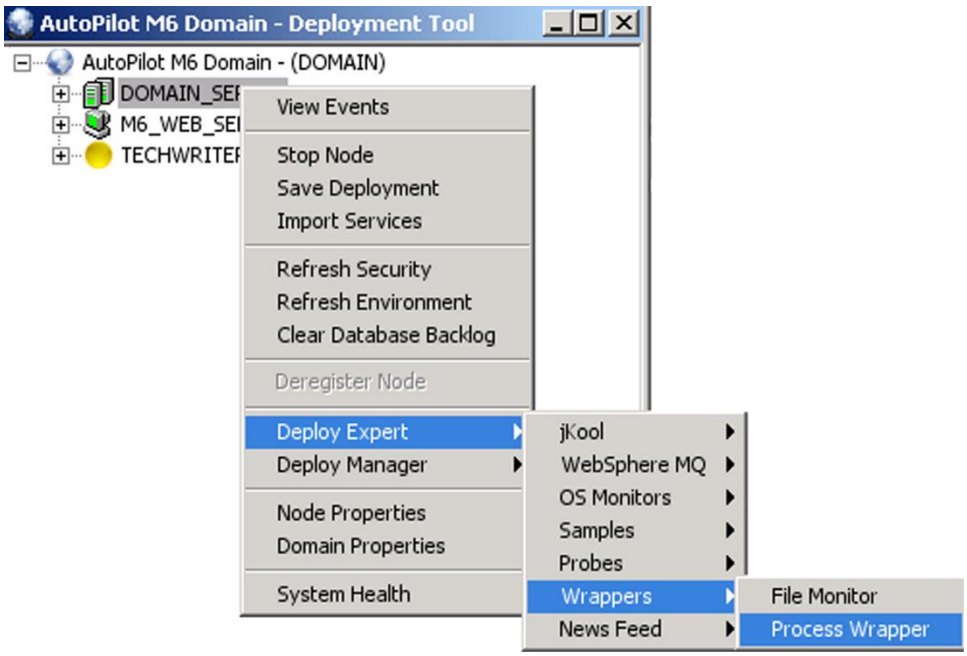


Figure 4-36. Deploying Process Wrapper Experts

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**.

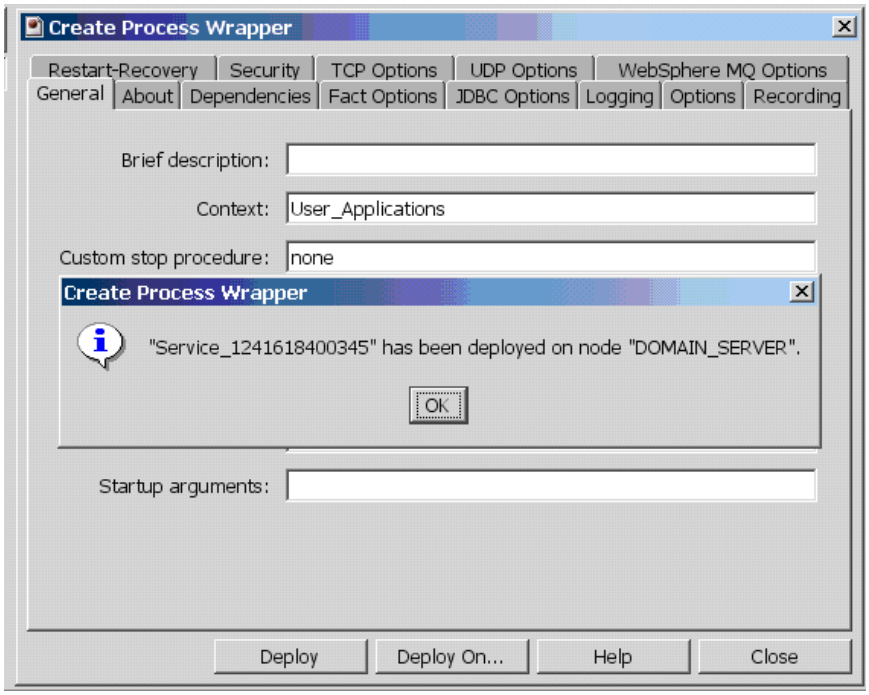


Figure 4-37. Deploying Wrappers

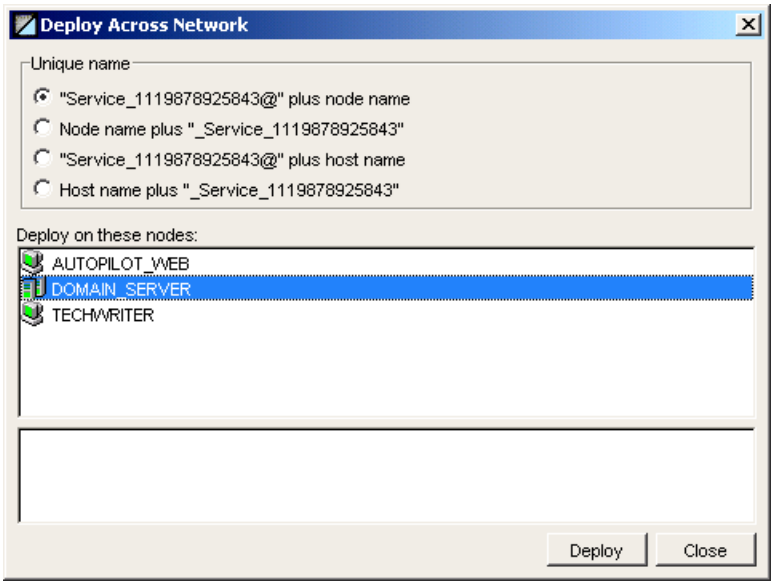


Figure 4-38. Deploy Wrapper on Multiple Nodes

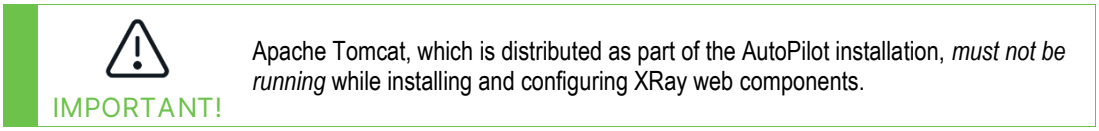
13.2.3 Check Logs and Connections

We recommend that you do the following before proceeding:

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

13.3Installing Web Server

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



13.3.1 Installing and Configuring Tomcat

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

13.3.1.1 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 ***tnt4j Logging***	Used to configure tnt4j log files (also	X	X

	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”
```

13.3.1.2 Configuring Tomcat Server.xml for XRay

This section contains examples for deploying the xray.war and jkool-server.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 ***Error! Reference source not found.*** for details.

13.3.1.2.1 Context Element Example: xray.war

The following example shows how to configure the Apache Tomcat server.xml file using the <Context> XML element for xray.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="/xray"
reloadable="true"
docBase="xray.war">

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

static:(tcp://localhost:61616,tcp://remotehost:61617?trace=false,vm://localbroker)?initialReconnectDelay=100 -->

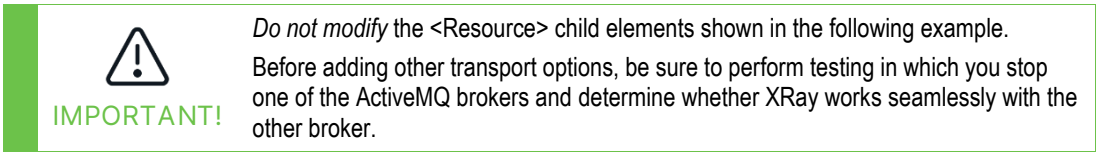
<Resource name="jms/FPConnectionFactory"
auth="Container"
type="org.apache.activemq.ActiveMQConnectionFactory"
factory="org.apache.activemq.jndi.JNDIReferenceFactory"

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 jKool 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 jKool 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 jKool 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 jKool 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>
```

13.3.1.2.2Context Element Example: jkool-service.war

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



```
<Context path="/jkool-service"
 reloadable="true"
 docBase="jkool-service.war">

 <!-- Client is using ActiveMQ to communicate w/ jKool Service -->
 <!-- The brokerURL host name or IP address should be that of the jKool 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)?r
andomize=false&timeout=10000&nested.connectionTimeout=120000&nested.wireFormat.
maxInactivityDuration=300000&nested.wireFormat.maxInactivityDurationInitalDelay=60
000"/>

 <!-- ActiveMQ queue on which to send requests - physicalName must match definition on jKool
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 jKool 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 jKool 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
jKool 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>

13.3.1.2.3Context Element Example: Connector settings

For XRay and Navigator, 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/nastel/AutoPilotM6/ssl/your.jks" keystorePass="password"
         clientAuth="false" SSLProtocol="TLSv1.2"
    />
  -->
```

13.3.1.3 (Optional) Reviewing Context.xml

After XRay.war has been extracted, you can find context.xml located at \xray\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/apache-tomcat/conf directory. In this location, it will take precedence over the context.xml located in the xray.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 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/apache-tomcat/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=”samlsso.manager.config” value=”${CATALINA_HOME}/conf/xray-samlsso.xml”/>

13.3.1.3.1Context.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>${catalina.base}/conf/web.xml</WatchedResource>
<!--samlsso configuration file -->
<Parameter name="apwmq.samlsso.manager.config" value="/opt/nastel/AutoPilotM6/apache-
tomcat/navigator_SSO.xml"/>
<Parameter name="samlsso.manager.config" value="${CATALINA_HOME}/conf/xray-samlsso.xml"/>
<!-- Uncomment this to disable session persistence across Tomcat restarts -->
<!--
<Manager pathname="" />
-->
<Resources
cachingAllowed="true"
cacheMaxSize="100000"
/>
</Context>
```

13.3.2 Installing and Configuring the XRay User Interface

To install the XRay user interface, do the following:

1. Download the appropriate distribution file. You must choose the version of the file that matches that of the JKOOL_SERVICE Plug-in you installed (see Installing the XRay JKOOL_SERVICE Plug-in).
   - For XRay 1.4, the file name is: `jkool-ui-x.y.z[.r]`
   - For XRay 1.5, the file name is: `xray-ui-x.y.z`

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

To configure the XRay 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:
   - If you are installing XRay 1.4, Update all **bold** entries below by changing **`xray`** to **`jKool`** (with a capital 'K').
   - 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="/xray"
                      reloadable="true"
                      docBase="xray.war">

          <!-- Client is using ActiveMQ to communicate w/ jKool
Service -->
          <!-- The brokerURL host name or IP address should be
that of the jKool 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 jKool 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 jKool 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 jKool 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 jKool 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>
```

13.3.3 Installing and Configuring the XRay REST endpoint

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

To install the XRay REST Service, do the following:

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

To configure the XRay REST Service, do the following:

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

```
        <Context path="/jkool-service"
                          reloadable="true"
                          docBase="jkool-service.war">

            <!-- Client is using ActiveMQ to communicate w/ jKool
Service -->
            <!-- The brokerURL host name or IP address should be
that of the jKool 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 jKool 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 jKool 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 jKool 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 jKool 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.)

Appendix

The appendix contains four file samples that are referred to in Chapter 6.

jndi.properties (for ActiveMQ broker cluster config)

```
# Service is using ActiveMQ to communicate w/ Clients
java.naming.factory.initial = org.apache.activemq.jndi.ActiveMQInitialContextFactory

# Connection url to ActiveMQ broker
#java.naming.provider.url =
tcp://${jkool.service.conn.str:localhost:61616}?jms.prefetchPolicy.all=256&wireFormat.maxInactivityDuration=0

# Important !!! After updating Xray DP change java.naming.provider.url according to ActiveMQ static transport broker config

java.naming.provider.url =
failover:(tcp://<activeMQBRK1>:61616,tcp://<activeMQBRK2>:61616)?jms.prefetchPolicy.all=256&wireFormat.maxInactivityDuration=0

# Fatpipes4J JMS connection factory
connectionFactoryNames = FPConnectionFactory, connectionFactory

# Queues
queue.admin-requests = jkool.service.admin.requests
queue.client-requests = jkool.service.requests
queue.jkadminreq-to-query = jksvc.to.jkadmreq.to.query
queue.jkql-to-exe = jksvc.to.jkql.to.exec
queue.jkreq-to-query = jksvc.to.jkreq.to.query
queue.jkreq-to-sub-grid = jksvc.jkreq.to.subgrid
queue.ml-predict-req = jksvc.ml.predict.req
queue.ml-predict-resp = jksvc.ml.predict.resp
queue.ml-query-req = jksvc.ml.query.req
queue.ml-query-resp = jksvc.ml.query.resp
queue.ml-train-req = jksvc.ml.train.req
queue.ml-train-resp = jksvc.ml.train.resp
queue.raw-tnt4j-data = jksvc.stream.data
queue.result-to-router = jksvc.result.to.router
queue.status-results = jksvc.status.results
queue.sub-data-to-real-time = jksvc.sub.data.to.realtime
queue.sub-req-to-real-time = jksvc.sub.req.to.realtime
queue.sub-result-to-handler = jksvc.sub.res.to.handler
queue.sub-status-data = jksvc.sub.status.data
queue.to-activity-analyzer = jksvc.to.act.analyzer
queue.to-activity-cleanser = jksvc.to.act.cleanser
queue.to-activity-stitcher = jksvc.to.act.stitcher
queue.to-analytic-grid = jksvc.to.analytic.grid
```

```
queue.to-compute-handler = jksvc.to.compute.handler
queue.to-compute-stager = jksvc.to.compute.stager
queue.to-def-exporter = jksvc.to.def.exporter
queue.to-jkql-scheduler = jksvc.to.jkql.scheduler
queue.to-log-writer = jksvc.to.log.writer
queue.to-model-trainer = jksvc.to.model.trainer
queue.to-sched-jkql-handler = jksvc.to.sched.jkql.handler
queue.to-sched-jkql-res-handler = jksvc.to.sched.jkql.res.handler
queue.to-script-handler = jksvc.to.script.handler
queue.to-source-updater = jksvc.to.source.updater
queue.to-stitching-stager = jksvc.to.stitching.stager
queue.to-update-stager = jksvc.to.update.stager
queue.trig-data-to-real-time = jksvc.trig.data.to.realtime
queue.trig-def-to-real-time = jksvc.trig.def.to.realtime
queue.trig-req-to-handler = jksvc.trig.req.to.handler
queue.trig-result-to-handler = jksvc.trig.res.to.handler
queue.trig-status-data = jksvc.trig.status.data
queue.trig-status-result = jksvc.trig.status.results
queue.update-requests = jkool.service.update.requests

#Topics
topic.item-update-requests = jksvc.item.upd.requests
topic.published-metrics = jksvc.metrics
topic.status-requests = jkool.service.status.requests
topic.subscription-metrics = jksvc.sub.metrics
topic.to-item-updater = jksvc.to.item.updater
topic.trigger-metrics = jksvc.trigger.metrics
```

server.xml (for ActiveMQ broker cluster config)

```
<?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.
-->
<!-- Note: A "Server" is not itself a "Container", so you may not
define subcomponents such as "Valves" at this level.
```

```
Documentation at /docs/config/server.html
-->
<Server port="8005" shutdown="1qazxsw23wsxzsawq32132321qwwedsa234e">
<Listener className="org.apache.catalina.startup.VersionLoggerListener" />
<!-- Security listener. Documentation at /docs/config/listeners.html
<Listener className="org.apache.catalina.security.SecurityListener" />
-->
<!-- APR library loader. Documentation at /docs/apr.html -->
<Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />
<!-- Prevent memory leaks due to use of particular java/javax APIs-->
<Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
<Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
<Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />

<!-- Global JNDI resources
Documentation at /docs/jndi-resources-howto.html
-->
<GlobalNamingResources>
<!-- Editable user database that can also be used by
UserDatabaseRealm to authenticate users
-->
<Resource name="UserDatabase" auth="Container"
type="org.apache.catalina.UserDatabase"
description="User database that can be updated and saved"
factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
pathname="conf/tomcat-users.xml" />
</GlobalNamingResources>

<!-- A "Service" is a collection of one or more "Connectors" that share
a single "Container" Note: A "Service" is not itself a "Container",
so you may not define subcomponents such as "Valves" at this level.
Documentation at /docs/config/service.html
-->
<Service name="Catalina">

<!--The connectors can use a shared executor, you can define one or more named thread pools-->
<!--
<Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
maxThreads="150" minSpareThreads="4"/>
-->


<!-- A "Connector" represents an endpoint by which requests are received
and responses are returned. Documentation at :
Java HTTP Connector: /docs/config/http.html
Java AJP Connector: /docs/config/ajp.html
APR (HTTP/AJP) Connector: /docs/apr.html
Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
-->
```

```
<Connector port="8887" protocol="HTTP/1.1"
server="Nastel XRay Server"
connectionTimeout="20000"
maxThreads="250"
proxyName="xraytest.nastel.com"
redirectPort="8443"
scheme="https"
secure="true"
proxyPort="443"
URIEncoding="UTF-8"
/> <!-- A "Connector" using the shared thread pool-->
<Connector
protocol="org.apache.coyote.http11.Http11NioProtocol"
port="9443"
maxThreads="200"
proxyName="xraytest.nastel.com"
redirectPort="8443"
ProxyPort="443"
scheme="https" secure="true" SSLEnabled="true"
keystoreFile="/opt/nastel/AutoPilotM6/ssl/wild/wild_nastel.jks" keystorePass="zaits123"
clientAuth="false"
/>




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


<Connector
protocol="org.apache.coyote.http11.Http11NioProtocol"
port="8443" maxThreads="200"
scheme="https" secure="true" SSLEnabled="true"
keystoreFile="/opt/nastel/AutoPilotM6/ssl/your.jks" keystorePass="password"
clientAuth="false" SSLProtocol="TLSv1.2"
/>

-->

<!-- A "Connector" using the shared thread pool-->
<!--
<Connector executor="tomcatThreadPool"
port="8080" protocol="HTTP/1.1"
connectionTimeout="20000"
redirectPort="8443" />
-->
<!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443
This connector uses the NIO implementation. The default
```

```
SSLImplementation will depend on the presence of the APR/native
library and the useOpenSSL attribute of the
AprLifecycleListener.
Either JSSE or OpenSSL style configuration may be used regardless of
the SSLImplementation selected. JSSE style configuration is used below.
-->
<!--
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
maxThreads="150" SSLEnabled="true">
<SSLHostConfig>
<Certificate certificateKeystoreFile="conf/localhost-rsa.jks"
type="RSA" />
</SSLHostConfig>
</Connector>
-->
<!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443 with HTTP/2
This connector uses the APR/native implementation which always uses
OpenSSL for TLS.
Either JSSE or OpenSSL style configuration may be used. OpenSSL style
configuration is used below.
-->
<!--
<Connector port="8443" protocol="org.apache.coyote.http11.Http11AprProtocol"
maxThreads="150" SSLEnabled="true" >
<UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
<SSLHostConfig>
<Certificate certificateKeyFile="conf/localhost-rsa-key.pem"
certificateFile="conf/localhost-rsa-cert.pem"
certificateChainFile="conf/localhost-rsa-chain.pem"
type="RSA" />
</SSLHostConfig>
</Connector>
-->

<!-- Define an AJP 1.3 Connector on port 8009
<Connector port="8009" connectionTimeout="20000" redirectPort="8443"
maxThreads="150" minSpareThreads="25" secretRequired="false" address="0.0.0.0"
enableLookups="false" acceptCount="10" debug="0" URIEncoding="UTF-8"
protocol="org.apache.coyote.ajp.AjpNioProtocol" proxyname="proxy.com" proxyPort="443"
scheme="https"

-->
<Connector
port="8009"
connectionTimeout="20000"
protocol="AJP/1.3"
redirectPort="8443"
secretRequired="false"
address="0.0.0.0"
```

```
maxThreads="150"
enableLookups="false"
proxyPort="443"
scheme="https"



/>
<!--
proxyname="xraytest.nastel.com"
<Connector protocol="AJP/1.3"
address="::1"
port="8009"
redirectPort="8443" />
-->

<!-- An Engine represents the entry point (within Catalina) that processes
every request. The Engine implementation for Tomcat stand alone
analyzes the HTTP headers included with the request, and passes them
on to the appropriate Host (virtual host).
Documentation at /docs/config/engine.html -->

<!-- You should set jvmRoute to support load-balancing via AJP ie :
<Engine name="Catalina" defaultHost="localhost" jvmRoute="jvm1">
-->
<Engine name="Catalina" defaultHost="localhost">

<!--For clustering, please take a look at documentation at:
/docs/cluster-howto.html (simple how to)
/docs/config/cluster.html (reference documentation) -->
<!--
<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
-->

<!-- Use the LockOutRealm to prevent attempts to guess user passwords
via a brute-force attack -->
<Realm className="org.apache.catalina.realm.LockOutRealm">
<!-- This Realm uses the UserDatabase configured in the global JNDI
resources under the key "UserDatabase". Any edits
that are performed against this UserDatabase are immediately
available for use by the Realm. -->
<Realm className="org.apache.catalina.realm.UserDatabaseRealm"
resourceName="UserDatabase"/>
</Realm>

<Host name="localhost" appBase="webapps"
unpackWARs="true" autoDeploy="true">
<Context path="/xray"
reloadable="true"
```

```
docBase="xray.war">

<!-- Client is using ActiveMQ to communicate w/ jKool Service -->
<!-- The brokerURL host name or IP address should be that of the jKool Service (which is
hosting ActiveMQ broker)
reference ActiveMQ documentation for Failover Transport Reference
https://activemq.apache.org/failover-transport-reference -->
<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)?randomi
ze=false&amp;timeout=10000&amp;nested.connectionTimeout=120000&amp;nested.wireForm
at.maxInactivityDuration=300000&amp;nested.wireFormat.maxInactivityDurationInitalDelay=6
0000"/>

<!-- ActiveMQ queue on which to send requests - physicalName must match definition on jKool
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
jKool 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
jKool 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
jKool 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>
```

```
<Context path="/jkool-service"
reloadable="true"
docBase="jkool-service.war">

<!-- Client is using ActiveMQ to communicate w/ jKool Service -->
<!-- The brokerURL host name or IP address should be that of the jKool 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)?randomi
ze=false&amp;timeout=10000&amp;nested.connectionTimeout=120000&amp;nested.wireForm
at.maxInactivityDuration=300000&amp;nested.wireFormat.maxInactivityDurationInitalDelay=6
0000"/>

<!-- ActiveMQ queue on which to send requests - physicalName must match definition on jKool
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
jKool 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
jKool 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
jKool 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>
```

```
<!-- SingleSignOn valve, share authentication between web applications
Documentation at: /docs/config/valve.html -->
<!--
<Valve className="org.apache.catalina.authenticator.SingleSignOn" />
-->

<!-- Access log processes all example.
Documentation at: /docs/config/valve.html
Note: The pattern used is equivalent to using pattern="common" -->
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
prefix="localhost_access_log" suffix=".txt"
pattern="%h %l %u %t &amp;quot;%r&amp;quot; %s %b" />

</Host>
</Engine>
</Service>
</Server>
```

activemq.xml (config on brk1)

```
<!--
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.
-->
<!-- START SNIPPET: example -->
<beans
xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://activemq.apache.org/schema/core http://activemq.apache.org/schema/core/activemq-
core.xsd">

<!-- Allows us to use system properties as variables in this configuration file -->
<bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
<property name="locations">
```

```
<value>file:${activemq.conf}/credentials.properties</value>
</property>
</bean>

<!--
The <broker> element is used to configure the ActiveMQ broker.
-->
<broker xmlns="http://activemq.apache.org/schema/core" brokerName="JKMessageBus"
dataDirectory="${activemq.data}" schedulePeriodForDestinationPurge="300000">

<destinationPolicy>
<policyMap>
<policyEntries>
<policyEntry topic=">" >
<!-- The constantPendingMessageLimitStrategy is used to prevent
slow topic consumers to block producers and affect other consumers
by limiting the number of messages that are retained
For more information, see:

http://activemq.apache.org/slow-consumer-handling.html

-->
<pendingMessageLimitStrategy>
<constantPendingMessageLimitStrategy limit="1000"/>
</pendingMessageLimitStrategy>
</policyEntry>

<!-- Delete jKool client result queues that are no longer used (inactive for 2 minutes)
-->
<policyEntry queue="jkool.client.>" gcInactiveDestinations="true"
inactiveTimoutBeforeGC="120000" sendAdvisoryIfNoConsumers="true"/>
</policyEntries>
</policyMap>
</destinationPolicy>


<!--
The managementContext is used to configure how ActiveMQ is exposed in
JMX. By default, ActiveMQ uses the MBean server that is started by
the JVM. For more information, see:

http://activemq.apache.org/jmx.html
-->
<managementContext>
<managementContext createConnector="false"/>
</managementContext>


<!--
```

```
The store and forward broker networks ActiveMQ will listen to.
We'll leave it empty as duplex network will be configured by another broker

-->

<networkConnectors>
</networkConnectors>


<!--
Configure message persistence for the broker. The default persistence
mechanism is the KahaDB store (identified by the kahaDB tag).
For more information, see:

http://activemq.apache.org/persistence.html
-->
<persistenceAdapter>
<kahaDB directory="${activemq.data}/kahadb"/>
</persistenceAdapter>


<!--
The systemUsage controls the maximum amount of space the broker will
use before disabling caching and/or slowing down producers. For more information, see:
http://activemq.apache.org/producer-flow-control.html
-->
<systemUsage>
<systemUsage>
<memoryUsage>
<memoryUsage percentOfJvmHeap="70" />
</memoryUsage>
<storeUsage>
<storeUsage limit="100 gb"/>
</storeUsage>
<tempUsage>
<tempUsage limit="50 gb"/>
</tempUsage>
</systemUsage>
</systemUsage>

<!--
The transport connectors expose ActiveMQ over a given protocol to
clients and other brokers. For more information, see:

http://activemq.apache.org/configuring-transports.html
-->
<transportConnectors>
<!-- DOS protection, limit concurrent connections to 1000 and frame size to 100MB -->
```

```
<transportConnector name="openwire"
uri="tcp://0.0.0.0:61616?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857
600&amp;wireFormat.maxInactivityDuration=0"/>
<transportConnector name="amqp"
uri="amqp://0.0.0.0:5672?maximumConnections=1000&amp;wireFormat.maxFrameSize=10485
7600"/>
<transportConnector name="stomp"
uri="stomp://0.0.0.0:61613?maximumConnections=1000&amp;wireFormat.maxFrameSize=104
857600"/>
<transportConnector name="mqtt"
uri="mqtt://0.0.0.0:1883?maximumConnections=1000&amp;wireFormat.maxFrameSize=10485
7600"/>
<transportConnector name="ws"
uri="ws://0.0.0.0:61614?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857
600"/>
</transportConnectors>

<!-- destroy the spring context on shutdown to stop jetty -->
<shutdownHooks>
<bean xmlns="http://www.springframework.org/schema/beans"
class="org.apache.activemq.hooks.SpringContextHook" />
</shutdownHooks>

</broker>

<!--
Enable web consoles, REST and Ajax APIs and demos
The web consoles requires by default login, you can disable this in the jetty.xml file

Take a look at ${ACTIVEMQ_HOME}/conf/jetty.xml for more details
-->
<import resource="jetty.xml"/>

</beans>
<!-- END SNIPPET: example -->
```

activemq.xml (config on brk2)

```
<!--
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.
-->
<!-- START SNIPPET: example -->
<beans
xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://activemq.apache.org/schema/core http://activemq.apache.org/schema/core/activemq-
core.xsd">

<!-- Allows us to use system properties as variables in this configuration file -->
<bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
<property name="locations">
<value>file:${activemq.conf}/credentials.properties</value>
</property>
</bean>

<!--
The <broker> element is used to configure the ActiveMQ broker.
-->
<broker xmlns="http://activemq.apache.org/schema/core" brokerName="JKMessageBus"
dataDirectory="${activemq.data}" schedulePeriodForDestinationPurge="300000">

<destinationPolicy>
<policyMap>
<policyEntries>
<policyEntry topic=">" >
<!-- The constantPendingMessageLimitStrategy is used to prevent
slow topic consumers to block producers and affect other consumers
by limiting the number of messages that are retained
For more information, see:

http://activemq.apache.org/slow-consumer-handling.html

-->
<pendingMessageLimitStrategy>
<constantPendingMessageLimitStrategy limit="1000"/>
</pendingMessageLimitStrategy>
</policyEntry>

<!-- Delete jKool client result queues that are no longer used (inactive for 2 minutes)
-->
<policyEntry queue="jkool.client.>" gcInactiveDestinations="true"
inactiveTimoutBeforeGC="120000" sendAdvisoryIfNoConsumers="true"/>
</policyEntries>
```

```
</policyMap>
</destinationPolicy>


<!--
The managementContext is used to configure how ActiveMQ is exposed in
JMX. By default, ActiveMQ uses the MBean server that is started by
the JVM. For more information, see:

http://activemq.apache.org/jmx.html
-->
<managementContext>
<managementContext createConnector="false"/>
</managementContext>

<!--
The store and forward broker networks ActiveMQ will listen to
Create a duplex connector to the first broker
For two node ActiveMQ cluster config need to add following in the <networkConnectors> for
only one of the brokers, either broker1 or broker2, but not both. And host/ipad is the host/ipad of
other broker.
-->

<networkConnectors>
<networkConnector uri="static:(tcp://<activeMQBRK1>:61616)" duplex="true"/>
</networkConnectors>


<!--
Configure message persistence for the broker. The default persistence
mechanism is the KahaDB store (identified by the kahaDB tag).
For more information, see:

http://activemq.apache.org/persistence.html
-->
<persistenceAdapter>
<kahaDB directory="${activemq.data}/kahadb"/>
</persistenceAdapter>


<!--
The systemUsage controls the maximum amount of space the broker will
use before disabling caching and/or slowing down producers. For more information, see:
http://activemq.apache.org/producer-flow-control.html
-->
<systemUsage>
<systemUsage>
<memoryUsage>
<memoryUsage percentOfJvmHeap="70" />
```

```
</memoryUsage>
<storeUsage>
<storeUsage limit="100 gb"/>
</storeUsage>
<tempUsage>
<tempUsage limit="50 gb"/>
</tempUsage>
</systemUsage>
</systemUsage>

<!--
The transport connectors expose ActiveMQ over a given protocol to
clients and other brokers. For more information, see:

http://activemq.apache.org/configuring-transports.html
-->
<transportConnectors>
<!-- DOS protection, limit concurrent connections to 1000 and frame size to 100MB -->
<transportConnector name="openwire"
uri="tcp://0.0.0.0:61616?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857
600&amp;wireFormat.maxInactivityDuration=0"/>
<transportConnector name="amqp"
uri="amqp://0.0.0.0:5672?maximumConnections=1000&amp;wireFormat.maxFrameSize=10485
7600"/>
<transportConnector name="stomp"
uri="stomp://0.0.0.0:61613?maximumConnections=1000&amp;wireFormat.maxFrameSize=104
857600"/>
<transportConnector name="mqtt"
uri="mqtt://0.0.0.0:1883?maximumConnections=1000&amp;wireFormat.maxFrameSize=10485
7600"/>
<transportConnector name="ws"
uri="ws://0.0.0.0:61614?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857
600"/>
</transportConnectors>

<!-- destroy the spring context on shutdown to stop jetty -->
<shutdownHooks>
<bean xmlns="http://www.springframework.org/schema/beans"
class="org.apache.activemq.hooks.SpringContextHook" />
</shutdownHooks>

</broker>

<!--
Enable web consoles, REST and Ajax APIs and demos
The web consoles requires by default login, you can disable this in the jetty.xml file

Take a look at ${ACTIVEMQ_HOME}/conf/jetty.xml for more details
-->
```

```
<import resource="jetty.xml"/>

</beans>
<!-- END SNIPPET: example -->
```