USING THE COMMAND LINE INTERFACE

The command line interface (CLI) allows interaction with the message broker by use of an interactive shell. You can manage broker actions, configure messages, add users and roles to files and enter other useful commands by using the CLI.

You can use the CLI to interact with the broker in a Bash or Zsh shell, or a custom **artemis** shell. The same commands are available in each shell. By default, the **artemis** shell has built-in auto-completion of commands and command parameters. You can add auto-completion of the CLI commands and command parameters to a Bash or Zsh shell also.

USING THE CLI IN AN **ARTEMIS** SHELL

The **artemis** shell interface provides auto-completion of commands and parameters that can be used with the **artemis** command. The shell also reuses the connection information provided, such as the broker URI and login credentials, for subsequent commands that you run in the same shell session.

Procedure

1. Switch to the user account you created during installation, for example:

   $ su - amq-broker

2. Use the **artemis** command to start the shell, for example:

   $ /var/opt/amq-broker/mybroker/bin/artemis

   If you want to supply the broker connection details when you start an **artemis** shell, use the **artemis shell** command. For example:

   $ /var/opt/amq-broker/mybroker/bin/artemis shell --user myuser --password mypassword --url tcp://localhost:61616

   The credentials and URI provided are reused for any subsequent command run in the shell that requires authentication with a broker.

3. Press Tab to display auto-completion information anywhere in the shell. For example:

   - To display a list of commands available in the **artemis** shell, press Tab at the shell prompt.

     To display sub-commands for a command, press Tab after the command. For example, type **check** and press Tab to display the sub-commands for the **check** command. The auto-completion information shows that the **check** command supports three sub-commands: **cluster**, **node** and **queue**.

   - To display auto-completion information for a sub-command, press Tab after the sub-command. For example, type **check node** and press Tab. If no further auto-completion information is displayed, type **--**, for example, **check node --**, and press Tab to display the parameters available for the command.

CONFIGURING AUTO-COMPLETION IN A BASH OR ZSH SHELL

You can configure auto-completion of CLI commands and command parameters in a Bash or Zsh shell.

Procedure

1. Switch to the user account you created during installation, for example:

   $ su - amq-broker

2. Use the following command to create a command auto-completion script called **auto-complete-artemis.sh**.

   $ /var/opt/amq-broker/mybroker/bin/artemis auto-complete

3. To run the **auto-complete-artemis.sh**, use the following command:

   $ source /var/opt/amq-broker/mybroker/bin/artemis/auto-complete-artemis.sh

4. To view auto-completion information for the **artemis** command, type **/var/opt/amq-broker/mybroker/artemis** and press Tab. A list of commands available for the **artemis** command is displayed.

5. To view auto-completion information for an available command, press Tab after the command. For example, type **/var/opt/amq-broker/mybroker/artemis check node** and press Tab. If no further auto-completion information is displayed, type **--**, for example, **check node --**, and press Tab to display the parameters available for the command.

STARTING BROKER INSTANCES

You can start a broker in the foreground by using the **artemis** script, as a Linux service, or as a Windows service.

Starting the broker instance

After the broker instance is created, you use the **artemis run** command to start it.

Procedure

1. Switch to the user account you created during installation.

   $ su - amq-broker

2. Use the **artemis run** command to start the broker instance.

   $ /var/opt/amq-broker/mybroker/bin/artemis run

```
        __  __  ____    ____            _
  /\   |  \/  |/ __ \  |  _ \          | |
 /  \  | \  / | |  | | | |_) |_ __ ___ | | _____ _ __
/ /\ \ | |\/| | |  | | |  _ <| '__/ _ \| |/ / _ \ '__|
/ ____ \| |  | | |__| | | |_) | | | (_) |   <  __/ |
/_/    \_\_|  |_|\___\_\ |____/|_|  \___/|_|\_\___|_|
```

Red Hat JBoss AMQ 7.2.1.GA

```
10:53:43,959 INFO  [org.apache.activemq.artemis.integration.bootstrap] AMQ101000: Starting ActiveMQ Artemis Server
10:53:44,076 INFO  [org.apache.activemq.artemis.core.server] AMQ221000: live Message Broker is starting with configuration Broker Configuration (clustered=false,journalDirectory=./data/journal,bindingsDirectory=./data/bindings,largeMessagesDirectory=./data/large-messages,pagingDirectory=./data/paging)
10:53:44,099 INFO [org.apache.activemq.artemis.core.server] AMQ221012: Using AIO Journal
...
```

The broker starts and displays log output with the following information:

- The location of the transaction logs and cluster configuration.
- The type of journal being used for message persistence (AIO in this case).
- The URI(s) that can accept client connections.
  By default, port 61616 can accept connections from any of the supported protocols (CORE, MQTT, AMQP, STOMP, HORNETQ, and OPENWIRE). There are separate, individual ports for each protocol as well.
- The web console is available at http://localhost:8161.
- The Jolokia service (JMX over REST) is available at http://localhost:8161/jolokia.

Starting a broker as a Linux service

If the broker is installed on Linux, you can run it as a service.

Procedure

1. Create a new **amq-broker.service** file in the **/etc/systemd/system/** directory.
2. Copy the following text into the file.
   Modify the path and user fields according to the information provided during the broker instance creation. In the example below, the user **amq-broker** starts the broker service installed under the **/var/opt/amq-broker/mybroker/** directory.

```
[Unit]
Description=AMQ Broker
After=syslog.target network.target

[Service]
ExecStart=/var/opt/amq-broker/mybroker/bin/artemis run
Restart=on-failure
User=amq-broker
Group=amq-broker

# A workaround for Java signal handling
SuccessExitStatus=143

[Install]
WantedBy=multi-user.target
```

3. Open a terminal.
4. Enable the broker service using the following command:

sudo systemctl enable amq-broker

5. Run the broker service using the following command:

sudo systemctl start amq-broker

Starting a broker as a Windows service

If the broker is installed on Windows, you can run it as a service.

Procedure

1. Open a command prompt to enter the commands
2. Install the broker as a service with the following command:

<*broker_instance_dir*>\bin\artemis-service.exe install

3. Start the service by using the following command:

<*broker_instance_dir*>\bin\artemis-service.exe start

4. (Optional) Uninstall the service:

<*broker_instance_dir*>\bin\artemis-service.exe uninstall

STOPPING BROKER INSTANCES

Stop the broker instance manually or configure the broker to shutdown gracefully.

Stopping the broker instance

After creating the standalone broker and producing and consuming test messages, you can stop the broker instance.

This procedure manually stops the broker, which forcefully closes all client connections. In a production environment, you should configure the broker to stop gracefully so that client connections can be closed properly.

Procedure

- Use the **artemis stop** command to stop the broker instance:

$ /var/opt/amq-broker/mybroker/bin/artemis stop

2018-12-03 14:37:30,630 INFO [org.apache.activemq.artemis.core.server] AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.6.1.amq-720004-redhat-1 [b6c244ef-f1cb-11e8-a2d7-0800271b03bd] stopped, uptime 35 minutes
Server stopped!

Stopping a broker instance gracefully

A manual shutdown forcefully disconnects all clients after a **stop** command is entered. As an alternative, configure the broker to shut down gracefully by using the **graceful-shutdown-enabled** configuration element.

When **graceful-shutdown-enabled** is set to **true**, no new client connections are allowed after a **stop** command is entered. However, existing connections are allowed to close on the client-side before the shutdown process is started. The default value for **graceful-shutdown-enabled** is **false**.

Use the **graceful-shutdown-timeout** configuration element to set a length of time, in milliseconds, for clients to disconnect before connections are forcefully closed from the broker side. After all connections are closed, the shutdown process is started. One advantage of using **graceful-shutdown-timeout** is that it prevents client connections from delaying a shutdown. The default value for **graceful-shutdown- timeout** is **-1**, meaning the broker waits indefinitely for clients to disconnect.

The following procedure demonstrates how to configure a graceful shutdown that uses a timeout.

Procedure

1. Open the configuration file ***<broker_instance_dir>*\etc\broker.xml**.

2. Add the **graceful-shutdown-enabled** configuration element and set the value to **true**.

```
<configuration>
   <core>
      ...
      <graceful-
         shutdown-
         enabled> true
      </graceful-shutdown-enabled>
      ...
   </core>
</configuration>
```

Add the **graceful-shutdown-timeout** configuration element and set a value for the timeout in milliseconds. In the following example, client connections are forcefully closed 30 seconds (**30000** milliseconds) after the **stop** command is issued.

```
<configuration>
   <core>
      ...
      <graceful-
         shutdown-
         enabled> true
      </graceful-shutdown-enabled>
      <graceful-
```

```
            shutdown-
            timeout> 30000
         </graceful-shutdown-timeout>
         ...
      </core>
   </configuration>
```

CHECKING THE HEALTH OF BROKERS, QUEUES AND THE CLUSTER

AMQ Broker includes a command-line utility that enables you to perform various health checks on brokers and queues in your broker topology. If your brokers are clustered, you can also use the utility to check the health of your cluster topology.

Procedure

1. See the list of checks that you can run for a particular broker (that is, *node*) in your broker topology.

   ```
   $ <broker_instance_dir>/bin/artemis help check node
   ```

   You see output that describes the set of options that you can use with the **artemis check node** command.

   ```
   NAME
         artemis check node - Check a node

   SYNOPSIS
         artemis check node [--backup] [--
               clientID <clientID>] [--diskUsage
               <diskUsage>]  [--fail-at-end]  [--
               live]
               [--memoryUsage <memoryUsage>] [--name <name>] [--
               password <password>] [--peers <peers>] [--protocol
               <protocol>] [--silent]

               [--timeout <timeout>] [--up] [--url <brokerURL>]
               [--user <user>] [--verbose]


   OPTIONS
               --backup
                  Check that the node has a backup

               --clientID <clientID>
                  ClientID to be associated with connection

               --diskUsage <diskUsage>
                  Disk usage percentage to check or -1 to use the max-disk-usage

               --fail-at-end
   ```

If a particular module check fails, continue the rest of the checks

--live
Check that the node has a live

--memoryUsage
<memoryUsage>
Memory usage
percentage to check

--name <name>
Name of the target to check

--password
<password>
Password used
to connect

--peers <peers>
Number of peers to check

--protocol <protocol>
Protocol used. Valid values are amqp or core. Default=core.

--silent
It will disable all the inputs, and it would make a best guess for any required input

--timeout <timeout>
Time to wait for the check execution, in milliseconds

--up
Check that the node is started, it is executed by default if there are no other checks

--url <brokerURL>
URL towards the broker. (default: tcp://localhost:61616)

--user <user>
User used to connect

--verbose
Adds more information on the execution

2. For example, check that the disk usage of the local broker is below the maximum disk usage configured for the broker.

```
$ <broker_instance_dir>/bin/artemis check node --url tcp://localhost:61616 --
diskUsage -1

Connection brokerURL =
tcp://localhost:61616 Running
NodeCheck
Checking that the disk usage is less then the max-disk-usage ... success
Checks run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.065 sec -
NodeCheck
```

In the preceding example, specifying a value of **-1** for the **--diskUsage** option means that the utility checks disk usage against the maximum disk usage configured for the broker. The maximum disk usage of a broker is configured using the **max-disk-usage** parameter in the **broker.xml** configuration file. The value specified for **max-disk-usage** represents the percentage of available physical disk space that the broker is allowed to consume.

3. See the list of checks that you can run for a particular queue in your broker topology.

```
$ <broker_instance_dir>/bin/artemis help check queue
```

You see output that describes the set of options that you can use with the **artemis check queue** command.

```
NAME
      artemis check queue - Check a queue

SYNOPSIS
      artemis check queue [--browse <browse>] [--
            clientID <clientID>] [--consume
            <consume>] [--fail-at-end] [--name
            <name>]
            [--password <password>] [--produce <produce>] [--
            protocol <protocol>]  [--silent] [--timeout <timeout>] [--
            up] [--url <brokerURL>]
            [--user <user>] [--verbose]


OPTIONS
      --browse <browse>
         Number of the messages to browse or -1 to check that
         the queue is browsable

      --clientID <clientID>
         ClientID to be associated with connection

      --consume <consume>
         Number of the messages to consume or -1 to check that the queue is
         consumable
```

```
--fail-at-end
   If a particular module check fails, continue the rest of the checks

--name <name>
   Name of the target to check

--password
   <password>
   Password used
   to connect

--produce <produce>
   Number of the messages to produce

--protocol <protocol>
   Protocol used. Valid values are amqp or core. Default=core.

--silent
   It will disable all the inputs, and it would make a best guess for any
   required input

--timeout <timeout>
   Time to wait for the check execution, in milliseconds

--up
   Check that the queue exists and is not paused, it is executed by
default if there are no other checks

--url <brokerURL>
   URL towards the broker. (default: tcp://localhost:61616)

--user <user>
   User used to connect

--verbose
   Adds more information on the execution
```

The utility can execute multiple options with a single command. For example, to check production, browsing, and consumption of 1000 messages on the default **helloworld** queue on the local broker, use the following command:

```
$ <broker_instance_dir>/bin/artemis check queue --name helloworld --
produce 1000 -- browse 1000 --consume 1000

Connection brokerURL =
tcp://localhost:61616 Running
QueueCheck
Checking that a producer can send 1000 messages to the queue helloworld
... success Checking that a consumer can browse 1000 messages from the
queue helloworld ... success Checking that a consumer can consume 1000
messages from the queue helloworld ... success
```

```
Checks run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.882 sec -
QueueCheck
```

In the preceding example, observe that you did not specify a broker URL when running the queue check. If you do not explicitly specify a URL, the utility uses a default value of **tcp://localhost:61616**.

If your brokers are clustered, see the list of checks that you can run for the cluster topology.

```
$ <broker_instance_dir>/bin/artemis help check cluster
```

You see output that describes the set of options that you can use with the **artemis check cluster** command.

```
NAME
        artemis check cluster - Verify if all the nodes on the cluster match the same
        topology and
time
        configuration.


SYNOPSIS
        artemis check cluster [--acceptor=<acceptor>]
                [--clientID=<clientID>]  [--password=<password>]
                [--protocol=<protocol>] [--silent] [--
                url=<brokerURL>] [--user=<user>] [--
                variance=<variance>] [--verbose]

OPTIONS
            --acceptor=<acceptor>
              Name used to find the default connection URL on the acceptor list. If an
        acceptor with that name cannot
              be found, the command looks for a connector with the same name.

            --clientID <clientID>
               ClientID to be associated with connection.

            --password <password>
               Password used to connect.

            --protocol <protocol>
               Protocol used. Valid values are amqp or core. Default=core.

            --silent
               Disables all the input options and make a best guess for any required
               input.

            --url <brokerURL>
               URL of the broker (default: tcp://localhost:61616).
```

--user <user>
User used to connect.

--variance <variance>
Allowed time variance in milliseconds after which check is deemed to have failed (default=1000).

--verbose
Adds more information on the execution.

COMMAND LINE TOOLS

AMQ Broker includes a set of command line interface (CLI) tools, so you can manage your messaging journal. The table below lists the name for each tool and its corresponding description.

Tool	Description
address	Addresses tool groups (create/delete/update/show) (example **./artemis address create**).
browser	Browses messages on an instance.
consumer	Consumes messages on an instance.
data	Prints reports about journal records and compacts the data.
decode	Imports the internal journal format from encode.
encode	Shows an internal format of the journal encoded to String.
exp	Exports the message data using a special and independent XML format.
help	Displays help information.
imp	Imports the journal to a running broker using the output provided by **exp**.
kill	Kills a broker instance started with --allow-kill.
mask	Masks a password and prints it out.
perf-journal	Calculates the journal-buffer timeout you should use with the current data folder.
queue	Queues tool groups (create/delete/update/stat) (example **./artemis queue create**).

run	Runs the broker instance.
stop	Stops the broker instance.
user	Default file-based user managament (add/rm/list/reset) (example **./artemis user list**)

For a full list of commands available for each tool, use the **help** parameter followed by the tool's name. For instance, in the example below, the CLI output lists all the commands available to the **data** tool after the user enters the command **./artemis help data**.

```
$ ./artemis

help data

NAME
      artemis data - data tools group
      (print|imp|exp|encode|decode|compact) (example
      ./artemis data print)

SYNOPSIS
      artemis data
      artemis data compact [--broker
            <brokerConfig>] [--verbose]  [--paging
            <paging>] [--journal <journal>]
            [--large-messages <largeMessges>] [--bindings
      <binding>] artemis data decode [--broker
      <brokerConfig>] [--suffix <suffix>]
            [--verbose] [--paging <paging>] [--prefix <prefix>] [--
            file-size <size>] [--directory <directory>] --input
            <input> [--journal <journal>]
            [--large-messages <largeMessges>] [--bindings
      <binding>] artemis data encode [--directory <directory>]
      [--broker <brokerConfig>]
            [--suffix <suffix>] [--verbose] [--paging <paging>] [--
            prefix <prefix>]  [--file-size <size>] [--journal
            <journal>]
            [--large-messages <largeMessges>] [--bindings
      <binding>] artemis data exp [--broker
      <brokerConfig>] [--verbose]
            [--paging <paging>] [--journal <journal>]
            [--large-messages <largeMessges>] [--bindings
      <binding>] artemis data imp [--host <host>] [--
      verbose] [--port <port>]
            [--password <password>] [--transaction] --input <input> [-
      -user <user>] artemis data print [--broker <brokerConfig>] [-
      -verbose]

            [--paging <paging>] [--journal <journal>]
            [--large-messages <largeMessges>] [--bindings <binding>]
```

```
COMMANDS
        With no arguments, Display help information

        print
           Print data records information (WARNING: don't use
           while a production server is running)

        ...
```

You can use the **help** parameter for more information on how to execute each of the commands. For example, the CLI lists more information about the **data print** command after the user enters the

./artemis help data print.

```
$ ./artemis

help data

print

NAME
        artemis data print - Print data records information
        (WARNING: don't use while a production server is running)

SYNOPSIS
        artemis data print [--bindings <binding>] [--
              journal <journal>] [--paging <paging>]

 OPTIONS
        --bindings <binding>
           The folder used for bindings (default ../data/bindings)

        --journal <journal>
           The folder used for messages journal (default ../data/journal)

        --paging <paging>
           The folder used for paging (default ../data/paging)
```