



Red Hat JBoss Fuse 6.0

Console Reference

Quick access to the commands used to manage the broker

Red Hat JBoss Fuse 6.0 Console Reference

Quick access to the commands used to manage the broker

JBoss A-MQ Docs Team

Content Services

fuse-docs-support@redhat.com

Legal Notice

Copyright © 2013 Red Hat.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

The primary interface for managing a container is the command console. This reference provides an easy way to see the commands, their syntax, and options.

Table of Contents

CHAPTER 1. USING THE COMMAND CONSOLE	16
OVERVIEW	16
STARTING THE COMMAND CONSOLE	16
GETTING HELP	16
COMMAND COMPLETION	17
COMMAND GROUPS	17
SHORT VERSION	18
CHAPTER 2. SHELL CONSOLE COMMANDS	20
NAME	20
SYNOPSIS	20
ARGUMENTS	20
NAME	20
SYNOPSIS	20
ARGUMENTS	20
NAME	21
SYNOPSIS	21
ARGUMENTS	21
NAME	21
SYNOPSIS	21
ARGUMENTS	21
NAME	21
SYNOPSIS	21
ARGUMENTS	22
NAME	22
SYNOPSIS	22
ARGUMENTS	22
NAME	23
SYNOPSIS	23
ARGUMENTS	23
NAME	23
SYNOPSIS	23
ARGUMENTS	23
NAME	24
SYNOPSIS	24
ARGUMENTS	24
NAME	24
SYNOPSIS	24
ARGUMENTS	24
NAME	24
SYNOPSIS	25
ARGUMENTS	25
NAME	25
SYNOPSIS	25
ARGUMENTS	25
NAME	25
SYNOPSIS	25
ARGUMENTS	25
NAME	26
SYNOPSIS	26
ARGUMENTS	26

NAME	26
SYNOPSIS	26
ARGUMENTS	26
NAME	27
SYNOPSIS	27
ARGUMENTS	27
NAME	27
SYNOPSIS	27
ARGUMENTS	27
NAME	28
SYNOPSIS	28
ARGUMENTS	28
NAME	28
SYNOPSIS	28
ARGUMENTS	28
NAME	29
SYNOPSIS	29
ARGUMENTS	29
NAME	29
SYNOPSIS	29
ARGUMENTS	30
CHAPTER 3. ACTIVEMQ CONSOLE COMMANDS	31
NAME	31
SYNOPSIS	31
ARGUMENTS	31
MESSAGE FILTERS	32
EXAMPLES	32
NAME	33
SYNOPSIS	33
ARGUMENTS	33
NAME	33
SYNOPSIS	33
ARGUMENTS	34
NAME	34
SYNOPSIS	34
ARGUMENTS	34
EXAMPLES	35
NAME	35
SYNOPSIS	35
ARGUMENTS	35
EXAMPLES	36
CHAPTER 4. ADMIN CONSOLE COMMANDS	38
NAME	38
SYNOPSIS	38
ARGUMENTS	38
NAME	38
SYNOPSIS	38
ARGUMENTS	38
NAME	39
SYNOPSIS	39
ARGUMENTS	39

NAME	39
SYNOPSIS	39
ARGUMENTS	39
NAME	39
SYNOPSIS	40
ARGUMENTS	40
NAME	40
SYNOPSIS	40
ARGUMENTS	40
NAME	41
SYNOPSIS	41
ARGUMENTS	41
NAME	42
SYNOPSIS	42
ARGUMENTS	42
NAME	42
SYNOPSIS	42
ARGUMENTS	42
NAME	43
SYNOPSIS	43
ARGUMENTS	43
NAME	43
SYNOPSIS	43
ARGUMENTS	43
NAME	43
SYNOPSIS	43
ARGUMENTS	44
CHAPTER 5. CAMEL CONSOLE COMMANDS	45
NAME	45
SYNOPSIS	45
ARGUMENTS	45
NAME	45
SYNOPSIS	45
ARGUMENTS	45
NAME	45
SYNOPSIS	45
ARGUMENTS	45
NAME	46
SYNOPSIS	46
ARGUMENTS	46
NAME	46
SYNOPSIS	46
ARGUMENTS	46
NAME	46
SYNOPSIS	47
ARGUMENTS	47
NAME	47
SYNOPSIS	47
DESCRIPTION	47
ARGUMENTS	47
NAME	47
SYNOPSIS	47

ARGUMENTS	47
NAME	48
SYNOPSIS	48
ARGUMENTS	48
NAME	48
SYNOPSIS	48
ARGUMENTS	48
NAME	49
SYNOPSIS	49
ARGUMENTS	49
NAME	49
SYNOPSIS	49
ARGUMENTS	49
CHAPTER 6. CONFIG CONSOLE COMMANDS	50
NAME	50
SYNOPSIS	50
DETAILS	50
ARGUMENTS	50
NAME	51
SYNOPSIS	51
DETAILS	51
ARGUMENTS	51
NAME	51
SYNOPSIS	51
DETAILS	51
ARGUMENTS	52
NAME	52
SYNOPSIS	52
ARGUMENTS	52
NAME	52
SYNOPSIS	52
DETAILS	53
ARGUMENTS	53
NAME	53
SYNOPSIS	53
DETAILS	53
ARGUMENTS	53
NAME	54
SYNOPSIS	54
ARGUMENTS	54
NAME	54
SYNOPSIS	54
DETAILS	54
ARGUMENTS	54
NAME	55
SYNOPSIS	55
ARGUMENTS	55
CHAPTER 7. CXF CONSOLE COMMANDS	56
NAME	56
SYNOPSIS	56
ARGUMENTS	56

NAME	56
SYNOPSIS	56
ARGUMENTS	56
NAME	56
SYNOPSIS	56
ARGUMENTS	57
NAME	57
SYNOPSIS	57
ARGUMENTS	57
CHAPTER 8. DEV CONSOLE COMMANDS	58
NAME	58
SYNOPSIS	58
ARGUMENTS	58
NAME	58
SYNOPSIS	58
ARGUMENTS	58
NAME	58
SYNOPSIS	58
ARGUMENTS	59
NAME	59
SYNOPSIS	59
ARGUMENTS	59
NAME	59
SYNOPSIS	59
ARGUMENTS	59
NAME	60
SYNOPSIS	60
ARGUMENTS	60
NAME	60
SYNOPSIS	60
ARGUMENTS	60
NAME	61
SYNOPSIS	61
ARGUMENTS	61
NAME	61
SYNOPSIS	61
DESCRIPTION	61
ARGUMENTS	61
NAME	62
SYNOPSIS	62
ARGUMENTS	62
CHAPTER 9. FUSE APPLICATION BUNDLE(FAB) CONSOLE COMMANDS	64
NAME	64
SYNOPSIS	64
DESCRIPTION	64
ARGUMENTS	64
NAME	65
SYNOPSIS	65
ARGUMENTS	65
NAME	65
SYNOPSIS	65

DESCRIPTION	65
ARGUMENTS	66
NAME	66
SYNOPSIS	66
ARGUMENTS	66
NAME	66
SYNOPSIS	66
ARGUMENTS	66
NAME	67
SYNOPSIS	67
DESCRIPTION	67
ARGUMENTS	67
CHAPTER 10. FABRIC CONSOLE COMMANDS	68
NAME	68
SYNOPSIS	68
DESCRIPTION	68
ARGUMENTS	68
RELATED TOPICS	68
NAME	68
SYNOPSIS	68
ARGUMENTS	68
NAME	69
SYNOPSIS	69
DESCRIPTION	69
INSTALLING THE COMMAND IN A FABRIC	70
ARGUMENTS	71
NAME	72
SYNOPSIS	72
DESCRIPTION	72
ARGUMENTS	72
NAME	72
SYNOPSIS	72
DESCRIPTION	72
ARGUMENTS	73
NAME	73
SYNOPSIS	73
ARGUMENTS	73
NAME	73
SYNOPSIS	74
DESCRIPTION	74
ARGUMENTS	74
RELATED TOPICS	74
NAME	74
SYNOPSIS	74
EXAMPLES	75
ARGUMENTS	75
RELATED TOPICS	76
NAME	76
SYNOPSIS	76
DESCRIPTION	76
SHUTTING DOWN CHILD CONTAINERS	77
ARGUMENTS	78

RELATED TOPICS	78
NAME	79
SYNOPSIS	79
DESCRIPTION	79
ARGUMENTS	80
RELATED TOPICS	83
NAME	83
SYNOPSIS	83
DESCRIPTION	83
ARGUMENTS	84
RELATED TOPICS	85
NAME	85
SYNOPSIS	85
DESCRIPTION	85
ARGUMENTS	86
NAME	86
SYNOPSIS	86
ARGUMENTS	86
NAME	86
SYNOPSIS	86
ARGUMENTS	86
NAME	87
SYNOPSIS	87
ARGUMENTS	87
NAME	87
SYNOPSIS	87
ARGUMENTS	87
NAME	88
SYNOPSIS	88
DESCRIPTION	88
ARGUMENTS	88
RELATED TOPICS	88
NAME	88
SYNOPSIS	88
DESCRIPTION	88
MANUAL IP RESOLVER POLICY	90
ARGUMENTS	90
NAME	90
SYNOPSIS	90
DESCRIPTION	90
ARGUMENTS	90
NAME	91
SYNOPSIS	91
ARGUMENTS	91
NAME	91
SYNOPSIS	91
ARGUMENTS	91
NAME	92
SYNOPSIS	92
DESCRIPTION	92
ARGUMENTS	93
NAME	93
SYNOPSIS	93

DESCRIPTION	93
ARGUMENTS	93
EXAMPLES	95
RELATED TOPICS	96
NAME	96
SYNOPSIS	96
DESCRIPTION	96
ARGUMENTS	96
NAME	97
SYNOPSIS	97
DESCRIPTION	97
ARGUMENTS	97
NAME	97
SYNOPSIS	97
DESCRIPTION	97
ARGUMENTS	97
NAME	97
SYNOPSIS	97
DESCRIPTION	97
ARGUMENTS	97
NAME	97
SYNOPSIS	97
DESCRIPTION	97
ARGUMENTS	98
NAME	98
SYNOPSIS	98
DESCRIPTION	98
ARGUMENTS	98
NAME	99
SYNOPSIS	99
ARGUMENTS	99
NAME	100
SYNOPSIS	100
DESCRIPTION	100
ARGUMENTS	100
EXAMPLES	101
NAME	102
SYNOPSIS	102
ARGUMENTS	102
EXAMPLES	103
NAME	104
SYNOPSIS	104
ARGUMENTS	104
NAME	104
SYNOPSIS	104
DESCRIPTION	105
ARGUMENTS	105
NAME	105
SYNOPSIS	105
ARGUMENTS	105
NAME	106
SYNOPSIS	106
ARGUMENTS	106
NAME	106
SYNOPSIS	106
DESCRIPTION	108
ENCLOSING AN OPTION VALUE IN QUOTES	113

ARGUMENTS	114
NAME	115
SYNOPSIS	115
DESCRIPTION	115
ARGUMENTS	115
NAME	116
SYNOPSIS	116
ARGUMENTS	116
NAME	116
SYNOPSIS	116
DESCRIPTION	117
ARGUMENTS	117
NAME	117
SYNOPSIS	117
DESCRIPTION	117
ARGUMENTS	118
NAME	118
SYNOPSIS	118
DESCRIPTION	119
ARGUMENTS	119
RELATED TOPICS	119
NAME	120
SYNOPSIS	120
DESCRIPTION	120
ARGUMENTS	120
NAME	121
SYNOPSIS	121
DESCRIPTION	121
ARGUMENTS	121
NAME	122
SYNOPSIS	122
DESCRIPTION	122
ARGUMENTS	122
NAME	122
SYNOPSIS	122
DESCRIPTION	122
ARGUMENTS	122
NAME	122
SYNOPSIS	122
DESCRIPTION	122
ARGUMENTS	123
CHAPTER 11. FEATURES CONSOLE COMMANDS	124
NAME	124
SYNOPSIS	124
DESCRIPTION	124
ARGUMENTS	124
NAME	125
SYNOPSIS	125
DESCRIPTION	125
ARGUMENTS	125
NAME	126
SYNOPSIS	126
ARGUMENTS	126
NAME	127
SYNOPSIS	127
ARGUMENTS	127

NAME	127
SYNOPSIS	127
ARGUMENTS	127
NAME	128
SYNOPSIS	128
ARGUMENTS	128
NAME	128
SYNOPSIS	129
ARGUMENTS	129
NAME	129
SYNOPSIS	129
ARGUMENTS	129
NAME	130
SYNOPSIS	130
ARGUMENTS	130
NAME	130
SYNOPSIS	130
ARGUMENTS	131
NAME	131
SYNOPSIS	131
ARGUMENTS	131
CHAPTER 12. JAAS CONSOLE COMMANDS	132
NAME	134
SYNOPSIS	134
DETAILS	134
ARGUMENTS	134
NAME	134
SYNOPSIS	134
DETAILS	135
ARGUMENTS	135
EXAMPLES	135
NAME	136
SYNOPSIS	136
DETAILS	136
ARGUMENTS	137
NAME	137
SYNOPSIS	137
ARGUMENTS	137
NAME	137
SYNOPSIS	137
DETAILS	138
ARGUMENTS	138
NAME	138
SYNOPSIS	138
DETAILS	138
ARGUMENTS	139
NAME	139
SYNOPSIS	139
DETAILS	139
ARGUMENTS	139
NAME	140
SYNOPSIS	140

DETAILS	140
ARGUMENTS	140
NAME	141
SYNOPSIS	141
DETAILS	141
ARGUMENTS	141
NAME	142
SYNOPSIS	142
ARGUMENTS	142
CHAPTER 13. JBI CONSOLE COMMANDS	143
NAME	143
SYNOPSIS	143
ARGUMENTS	143
NAME	143
SYNOPSIS	143
ARGUMENTS	144
NAME	144
SYNOPSIS	144
ARGUMENTS	144
NAME	145
SYNOPSIS	145
ARGUMENTS	145
CHAPTER 14. LOG CONSOLE COMMANDS	146
NAME	146
SYNOPSIS	146
ARGUMENTS	146
NAME	146
SYNOPSIS	146
ARGUMENTS	146
NAME	147
SYNOPSIS	147
ARGUMENTS	147
NAME	147
SYNOPSIS	147
ARGUMENTS	148
NAME	148
SYNOPSIS	148
ARGUMENTS	148
NAME	149
SYNOPSIS	149
ARGUMENTS	149
CHAPTER 15. THE NMR:LIST COMMAND	150
NAME	150
SYNOPSIS	150
ARGUMENTS	150
CHAPTER 16. OBR CONSOLE COMMANDS	151
NAME	151
SYNOPSIS	151
ARGUMENTS	151
NAME	152

SYNOPSIS	152
ARGUMENTS	152
NAME	152
SYNOPSIS	152
ARGUMENTS	152
NAME	153
SYNOPSIS	153
ARGUMENTS	153
NAME	153
SYNOPSIS	153
ARGUMENTS	153
NAME	154
SYNOPSIS	154
ARGUMENTS	154
NAME	154
SYNOPSIS	154
ARGUMENTS	154
NAME	155
SYNOPSIS	155
ARGUMENTS	155
NAME	155
SYNOPSIS	155
ARGUMENTS	156
CHAPTER 17. OSGI CONSOLE COMMANDS	157
NAME	157
SYNOPSIS	157
ARGUMENTS	157
NAME	157
SYNOPSIS	158
ARGUMENTS	158
NAME	158
SYNOPSIS	158
ARGUMENTS	159
NAME	159
SYNOPSIS	159
ARGUMENTS	159
NAME	160
SYNOPSIS	160
ARGUMENTS	160
NAME	160
SYNOPSIS	160
ARGUMENTS	160
NAME	161
SYNOPSIS	161
ARGUMENTS	161
NAME	161
SYNOPSIS	161
ARGUMENTS	162
NAME	162
SYNOPSIS	162
ARGUMENTS	163
NAME	163

SYNOPSIS	163
ARGUMENTS	163
NAME	164
SYNOPSIS	164
ARGUMENTS	164
NAME	164
SYNOPSIS	164
ARGUMENTS	164
NAME	165
SYNOPSIS	165
ARGUMENTS	165
NAME	166
SYNOPSIS	166
ARGUMENTS	166
NAME	166
SYNOPSIS	166
ARGUMENTS	167
NAME	167
SYNOPSIS	167
ARGUMENTS	167
NAME	168
SYNOPSIS	168
ARGUMENTS	168
NAME	168
SYNOPSIS	168
ARGUMENTS	168
CHAPTER 18. PACKAGES CONSOLE COMMANDS	170
NAME	170
SYNOPSIS	170
ARGUMENTS	170
NAME	171
SYNOPSIS	171
ARGUMENTS	171
CHAPTER 19. PATCH CONSOLE COMMANDS	172
NAME	173
SYNOPSIS	173
ARGUMENTS	173
NAME	173
SYNOPSIS	173
ARGUMENTS	174
NAME	174
SYNOPSIS	174
ARGUMENTS	174
NAME	174
SYNOPSIS	175
ARGUMENTS	175
NAME	175
SYNOPSIS	175
ARGUMENTS	175
CHAPTER 20. SSH CONSOLE COMMANDS	176
NAME	176

SYNOPSIS	176
ARGUMENTS	176
NAME	177
SYNOPSIS	177
ARGUMENTS	177
CHAPTER 21. WEB CONSOLE COMMANDS	178
NAME	178
SYNOPSIS	178
ARGUMENTS	178
CHAPTER 22. THE WRAPPER:INSTALL COMMAND	179
NAME	179
SYNOPSIS	179
ARGUMENTS	179
CHAPTER 23. ZOOKEEPER CONSOLE COMMANDS	181
NAME	181
SYNOPSIS	181
DESCRIPTION	181
ARGUMENTS	183
NAME	184
SYNOPSIS	184
ARGUMENTS	184
NAME	185
SYNOPSIS	185
ARGUMENTS	185
NAME	185
SYNOPSIS	185
ARGUMENTS	186
NAME	186
SYNOPSIS	186
DESCRIPTION	186
ARGUMENTS	187

CHAPTER 1. USING THE COMMAND CONSOLE

OVERVIEW

The Red Hat JBoss Fuse command console is the central tool for both managing the JBoss Fuse environment and interacting with Fuse Fabric. When you start JBoss Fuse the console starts automatically.

The console provides commands that you can use to perform basic management of your JBoss Fuse environment, including deploying and configuring applications.

The console uses prefixes to group commands relating to the same functionality. For example, commands related to configuration are prefixed **config:**, and logging-related commands are prefixed **log:**.

STARTING THE COMMAND CONSOLE

To start JBoss Fuse, open a command prompt in the installation directory and enter:

Windows	<code>bin\fuse</code>
*NIX	<code>bin/fuse</code>

JBoss Fuse starts and the console is ready. You should see a prompt similar to this:

```

  _ _ _ _ _ _ _ _ _ _ | | _ \ | _ _ | | | |_) | _ _ _ _ _ | | _ _ _ _ _ _ _ _ |
 | _ < / _ \ / _ / _ | | _ | | | / _ | / _ \ | | _ | | |_) | ( ) \_ \_ \ | |
 | | | \_ \ _ / \_ / | _ / \_ / | _ / _ / | | \_ , _ | _ / \_ | JBoss Fuse
 (6.0.0.redhat-xxx)
 http://www.redhat.com/products/jbossenterprisemiddleware/fuse/ Hit '<tab>'
 for a list of available commands and '[cmd] --help' for help on a specific
 command. Hit '<ctrl-d>' or 'osgi:shutdown' to shutdown JBoss Fuse.
 JBossFuse:karaf@root

```

GETTING HELP

The console provides two levels of help:

- console help—lists all of the commands along with a brief summary of the commands function
- command help—provides a detailed description of a command and its arguments

To access the console help, enter the **help** command from the console prompt. It displays a grouped list of all the commands available in the console. Each command in the list is followed by a description, as shown in [Example 1.1, “Console Help”](#).

Example 1.1. Console Help

```

JBossFuse:karaf@root> help
COMMANDS activemq:browse activemq:bstat activemq:create-broker Creates a
broker instance. activemq:destroy-broker Destory a broker instance.

```

```
activemq:list activemq:purge activemq:query admin:change-opts Changes
the Java options of an existing container instance. admin:change-rmi-
registry-port Changes the RMI registry port (used by management layer)
of an existing container instance.
```

```
...
JBossFuse:karaf@root>
```

The help for each command includes the definition, the syntax, and the arguments and any options. To display the help for a command, type the command with the `--help` option. As shown in [Example 1.2, “Help for a Command”](#), entering `admin:start --help` displays the help for that command.

Example 1.2. Help for a Command

```
JBossFuse:karaf@root> admin:start --help
DESCRIPTION admin:start Starts an existing container instance. SYNTAX
admin:start [options] name ARGUMENTS name The name of the container
instance OPTIONS --help Display this help message -o, --java-opts Java
options when launching the instance
JBossFuse:karaf@root>
```

COMMAND COMPLETION

Pressing **Tab** at anytime provides you with a list of commands that can complete what you have already entered at the prompt. For example, if you enter `active` followed by **Tab**, a list similar to [Example 1.3, “Console Commands”](#) is shown.

Example 1.3. Console Commands

```
activemq:browse activemq:bstat activemq:create-broker activemq:destroy-
broker activemq:list activemq:purge activemq:query JBossA-MQ:karaf@root>
```

If you press **Tab** without entering anything at the prompt, the console lists all of the available commands.

COMMAND GROUPS

Commands are grouped under prefixes according to functionality. [Table 1.1, “Apache ActiveMQ Command Groups”](#) summarizes the command groups available in the console. Click on a command group name for more information.

Table 1.1. Apache ActiveMQ Command Groups

Command Group	Description
activemq	Views and manages brokers and messages.
admin	Creates, manages, and destroys containers.

Command Group	Description
camel	Manages Apache Camel contexts and routes
config	Manages configuration.
cxfr	Manages Apache CXF buses and endpoints.
dev	Utilities that are useful for a developer while testing bundles in the container.
fab	Manages the dependency resolution mechanism used by Fuse Application Bundles.
fabric	Performs provisioning and configuration using Fuse Fabric.
features	Performs provisioning based on Apache Karaf feature specs.
jaas	Manages the console's security settings.
jbi	Manage deployed JBI artifacts.
log	Displays and configures logging.
nmr	Lists NMR endpoints.
obr	Accesses the OSGi Bundle Repository (OBR).
osgi	Manages OSGi bundles.
packages	Lists imported and exported packages.
patch	Manages patches.
shell	Performs basic console functions
ssh	Creates and connects to a remote SSH server
web	Lists the WARs deployed in the container.
zk	Accesses and modifies entries in the Zookeeper registry.

SHORT VERSION

Many of the console commands allow you to omit the group prefix.

If the command is only in one command groups, you can omit the group prefix. For example, you can enter **bstat** in place of **activemq:bstat** because it only exists in the activemq command group.

If the command exists in multiple command groups, you can still drop the prefix and the console will default to using the version of the command from one of the following command groups:

- shell
- osgi
- admin

For example, **info** is equivalent to **shell:info**. If you wanted to use **osgi:info**, you need to enter the full command.

CHAPTER 2. SHELL CONSOLE COMMANDS

The shell command group provides a number of commands that provide basic console functions such as displaying system information and showing the contents of files.

Type **shell**: then press **Tab** at the prompt to view the commands in this group.

NAME

shell:cat, cat — displays the contents of a file or URL

SYNOPSIS

```
shell:cat [ -n ] [ --help ] { [ path ] | [ URL ] }
```

ARGUMENTS

Table 2.1, “[shell:cat Arguments](#)” describes the arguments for this command.

Table 2.1. shell:cat Arguments

Argument	Interpretation
-n	Display line numbers.
--help	Displays the online help for this command
<i>path</i>	The path(s) of the file to display, separated by whitespace (separated by - for STDIN)
<i>URL</i>	The URL(s) to display, separated by whitespace (separated by - for STDIN)

NAME

shell:clear, clear — clears the console buffer

SYNOPSIS

```
shell:clear [ --help ]
```

ARGUMENTS

Table 2.2, “[shell:clear Arguments](#)” describes the command's arguments.

Table 2.2. shell:clear Arguments

Argument	Interpretation
--help	Displays the online help for this command

NAME

shell:each, each — execute a closure on a list of arguments

SYNOPSIS

```
shell:each [ --help ] { values } { function }
```

ARGUMENTS

Table 2.3, “[shell:each Arguments](#)” describes the command's arguments.

Table 2.3. shell:each Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>values</i>	The collection of arguments to iterate over.
<i>function</i>	The function to execute.

NAME

shell:echo, echo — prints arguments to the standard output

SYNOPSIS

```
shell:echo [ --help ] [ -n ] { argument ... }
```

ARGUMENTS

Table 2.4, “[shell:echo Arguments](#)” describes the command's arguments.

Table 2.4. shell:echo Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>-n</code>	Do not print the trailing newline character.
<i>argument</i>	Specifies a space delimited list of arguments to print.

NAME

shell:exec, exec — executes system processes

SYNOPSIS

```
shell:exec [ --help ] { command }
```

ARGUMENTS

Table 2.5, “[shell:exec Arguments](#)” describes the command's arguments.

Table 2.5. `shell:exec` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>command</i>	Specifies the command, with arguments, to execute.

NAME

`shell:grep`, `grep` — displays lines matching a regular expression

SYNOPSIS

```
shell:grep [ --help ] [ -i ] | [ --ignore-case ] [ -w ] | [ --word-regexp ] [ -n ] | [ --line-number ] [ -x ] | [ --line-regexp ] [ -v ] | [ --invert-match ] { regex }
```

ARGUMENTS

Table 2.6, “[shell:grep Arguments](#)” describes the command's arguments.

Table 2.6. `shell:grep` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>-i</code> , <code>--ignore-case</code>	Ignore case distinctions in both the <i>regex</i> and the input files.
<code>-w</code> , <code>--word-regexp</code>	Select only lines containing matches that form whole words. A match qualifies if it meets one of the following conditions: <ul style="list-style-type: none"> • The matching string is at the beginning of the line. • The matching string is preceded by a non-word constituent character. • The matching string is at the end of the line. • The matching string is followed by a non-word constituent character.
<code>-n</code> , <code>--line-number</code>	Display the line number of the match within its input file.

Argument	Interpretation
-x, --line-regexp	Selects only those matches that exactly match the whole line.
-v, --invert-match	Select non-matching lines.
<i>regex</i>	Specifies the regular expression to match.

NAME

shell:head, head — displays the first lines of a file

SYNOPSIS

```
shell:head [ --help ] [ -n numLines ] [{ path } | [ URL ]]
```

ARGUMENTS

Table 2.7, “[shell:head Arguments](#)” describes the command's arguments.

Table 2.7. shell:head Arguments

Argument	Interpretation
--help	Displays the online help for this command.
-n	Specifies the number of lines to display. Default is 1.
<i>path</i>	The path(s) of the file to display, separated by whitespace (separated by - for STDIN)
<i>URL</i>	Specifies the URL(s) to display, separated by whitespace (separated by - for STDIN)

NAME

shell:history, history — prints the command history

SYNOPSIS

```
shell:history [ --help ]
```

ARGUMENTS

Table 2.8, “[shell:history Arguments](#)” describes the arguments for this command.

Table 2.8. shell:history Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

NAME

shell:if, if — executes an if/then/else block

SYNOPSIS

`shell:if` [`--help`] { *condition* } { *ifTrue* } [*ifFalse*]

ARGUMENTS

Table 2.9, “[shell:if Arguments](#)” describes the command's arguments.

Table 2.9. shell:if Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>condition</i>	Boolean condition.
<i>ifTrue</i>	Function to evaluate, if condition is true.
<i>ifFalse</i>	Function to evaluate, if condition is false.

NAME

shell:info, info — displays system information and statistics about the container

SYNOPSIS

`shell:info` [`--help`]

ARGUMENTS

Table 2.10, “[shell:info Arguments](#)” describes the command's arguments.

Table 2.10. shell:info Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this utility

NAME

shell:java, java — execute a Java application

SYNOPSIS

shell:java [--help] [[-m] | [--method] *methodName*] { *className* } [*arguments*]

ARGUMENTS

Table 2.11, “shell:java Arguments” describes the command's arguments.

Table 2.11. shell:java Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-m</code> , <code>--method</code>	Specifies the name of a method to invoke. The default is <code>main()</code> .
<i>className</i>	Specifies the name of the class to invoke.
<i>arguments</i>	Specifies the arguments to pass to the method of the given <i>className</i> .

NAME

shell:logout, logout — disconnects the shell from the current session

SYNOPSIS

shell:logout [--help]

ARGUMENTS

Table 2.12, “shell:logout Arguments” describes the command's arguments.

Table 2.12. shell:logout Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

NAME

shell:more, more — displays output as pages of a specified length

SYNOPSIS

shell:more [--help] [--lines *numLines*]

ARGUMENTS

Table 2.13, “shell:more Arguments” describes the command's arguments.

Table 2.13. shell:more Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>--lines</code>	Specifies the number of lines to display before pausing.

NAME

shell:new, new — creates a new Java object of the specified class

SYNOPSIS

```
shell:new [ --help ] { class } [ arg ...]
```

ARGUMENTS

Table 2.14, “[shell:new Arguments](#)” describes the command's arguments.

Table 2.14. shell:new Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>class</i>	The class of the object to create.
<i>args</i>	The constructor arguments.

NAME

shell:printf, printf — formats and prints the specified output

SYNOPSIS

```
shell:printf [ --help ] { format } { arguments }
```

ARGUMENTS

Table 2.15, “[shell:printf Arguments](#)” describes the command's arguments.

Table 2.15. shell:printf Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>format</i>	The output format pattern to use

Argument	Interpretation
<i>arguments</i>	The arguments for the given format pattern

NAME

shell:sleep, sleep — sleeps for a specified time, then wakes up

SYNOPSIS

```
shell:sleep [ --help ] [[ -s ] | [ --second ] ] { duration }
```

ARGUMENTS

Table 2.16, “[shell:sleep Arguments](#)” describes the command's arguments.

Table 2.16. shell:sleep Arguments

Argument	Interpretation
--help	Displays the online help for this command
-s, --second	Specify the duration in seconds (instead of milliseconds).
<i>duration</i>	The time to sleep in milliseconds (default) or in seconds (with the -s option).

NAME

shell:sort, sort — writes a sorted concatenation of the specified files to standard output

SYNOPSIS

```
shell:sort [ --help ] [[ -t ] | [ --field-separator ] sep] [[ -b ] | [ --ignore-leading-blanks ] ] [[ -f ] | [ --ignore-case ] ] [[ -r ] | [ --reverse ] ] [[ -k ] | [ --key ] keys] [[ -n ] | [ --numeric-sort ] ] [[ -u ] | [ --unique ] ] { file ... }
```

ARGUMENTS

Table 2.17, “[shell:sort Arguments](#)” describes the command's arguments.

Table 2.17. shell:sort Arguments

Argument	Interpretation
--help	Displays the online help for this command
-t, --field-separator	Specifies a character to use as a field separator. The default is whitespace.

Argument	Interpretation
-b, --ignore-leading-blanks	Ignores leading blanks.
-f, --ignore-case	Ignores case when sorting.
-r, --reverse	Reverses the result of the sort.
-k, --key	Specifies a space delimited list of fields to use for sorting.
-n, --numeric-set	Compares according to string numerical value.
-u, --unique	Outputs only the first of an equal run.
<i>files</i>	Specifies a space delimited list of files to sort.

NAME

shell:source, source — run a shell script

SYNOPSIS

```
shell:source [ --help ] { script } [ arguments ]
```

ARGUMENTS

Table 2.18, “[shell:source Arguments](#)” describes the command's arguments.

Table 2.18. shell:source Arguments

Argument	Interpretation
--help	Displays the online help for this command
<i>script</i>	A URI pointing to the script
<i>arguments</i>	Arguments to pass to the script

NAME

shell:tac, tac — captures the STDIN and returns it as a string and optionally writes the content to a file

SYNOPSIS

```
shell:tac [ --help ] [ -f fileName ]
```

ARGUMENTS

Table 2.19, “[shell: tac Arguments](#)” describes the command's arguments.

Table 2.19. shell: tac Arguments

Option	Interpretation
--help	Displays the online help for this command
-f	Specifies the name of the file into which the output is written.

NAME

shell:tail, tail — displays the last lines of a file

SYNOPSIS

```
shell:head [ --help ] [ -n lineNum ] [ -s seconds ] [ -f ] { [ path ] | [ URL ]...}
```

ARGUMENTS

Table 2.20, “[shell: tail Arguments](#)” describes the command's arguments.

Table 2.20. shell: tail Arguments

Argument	Interpretation
--help	Displays the online help for this command.
-n	Specifies the number of lines to display. The default is 1.
-s	Specifies the interval, in seconds, to sleep before checking for changes to display.
-f	Follow file changes.
<i>path</i>	A space delimited list of file paths to display.
<i>URL</i>	A space delimited list of file URLs to display.

NAME

shell:watch, watch — watches and refreshes the output of a command

SYNOPSIS

```
shell:watch [ --help ] [ [ -n ] | [ --interval ] seconds ] { command }
```

ARGUMENTS

Table 2.21, “[shell:watch Arguments](#)” describes the command's arguments.

Table 2.21. `shell:watch` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-n, --interval</code>	Specifies the interval, in seconds, between executions of the command. The default is 1.
<i>command</i>	Specifies the command to watch and refresh.

CHAPTER 3. ACTIVEMQ CONSOLE COMMANDS

The **activemq** commands allow you to view and manage the brokers and messages.

Type **activemq**: then press **Tab** at the prompt to view the available commands.

NAME

activemq:browse, **browse** — displays messages on a specified destination

SYNOPSIS

```
activemq:browse { --amqurl brokerURL } [ --msgsel { msgsel ... } ] [ --factory className ] [ --passwordFactory className ] [ --user username ] [ --password password ] [ --view { attr ... } ] [ [ -Vheader ] | [ -Vcustom ] | [ -Vbody ] ] [ --version ] [ [ --help ] | [ -h ] | [ -? ] ] destName
```

ARGUMENTS

Table 3.1, “**activemq:browse Arguments**” describes the command's arguments.

Table 3.1. **activemq:browse Arguments**

Argument	Interpretation
--amqurl <i>brokerURL</i>	Specifies the URL of the broker to which you are connecting.
--msgsel <i>msgsel1, msgsel2, ...</i>	Displays messages matched by the message selector.
--factory <i>className</i>	Load <i>className</i> as the <code>javax.jms.ConnectionFactory</code> to use for creating connections.
--passwordFactory <i>className</i>	Load <i>className</i> as the org.apache.activemq.console.command.PasswordFactory for retrieving the password from a keystore.
--user <i>username</i>	Username to use for JMS connections.
--password <i>password</i>	Password to use for JMS connections.
-Vheader	Shows all the standard JMS message headers.
-Vcustom	Shows all the custom fields added to each JMS message.
-Vbody	Shows the body of the message.
--view <i>attr1, attr1, ...</i>	Selects the specific attribute of the message to view.
--version	Displays the version information.

Argument	Interpretation
<code>-h, -?, --help</code>	Displays the online help for this command.

MESSAGE FILTERS

Message filters specified using the `--msgsel` option take the form *header=value*. [Table 3.2](#), “[Message Headers for Filtering](#)” lists the headers you can use to filter messages.

Table 3.2. Message Headers for Filtering

Name	Type
JMSCorrelationID	String
JMSDeliveryMode	1-Non-Persistent, 2-Persistent
JMSDestination	javax.jms.Destination
JMSExpiration	long
JMSMessageID	String
JMSPriority	int
JMSRedelivered	boolean
JMSReplyTo	javax.jms.Destination
JMSTimestamp	long
JMSType	String

EXAMPLES

The following command prints the JMS message header, custom message header, and message body of all the messages in the queue **TEST.FOO** on a broker:

```
JBossA-MQ:karaf@root>activemq:browse --amqurl tcp://localhost:61616
TEST.FOO
```

The following command displays the attributes from the body of the messages in the **TEST.FOO** queue:

```
JBossA-MQ:karaf@root>activemq:browse --amqurl tcp://localhost:61616 -Vbody
TEST.FOO
```

The following command displays any messages with an ID ending in **10**:

```
JBossA-MQ:karaf@root>activemq:browse --amqurl tcp://localhost:61616 --
msgsel JMSMessageID='*:10' TEST.FOO
```

The following command displays messages with a priority of **3**, enter:

```
JBossA-MQ:karaf@root>activemq:browse --amqurl tcp://localhost:61616 --
msgsel JMSPriority=3 TEST.FOO
```

The message selectors from the preceding two examples can be combined as follows:

```
JBossA-MQ:karaf@root>activemq:browse --amqurl tcp://localhost:61616 --
msgsel JMSMessageID='*:10',JMSPriority=3 TEST.FOO
```

NAME

activemq:bstat, bstat — summarizes the statistics for a broker

SYNOPSIS

```
activemq:bstat [ --jmxurl JMXUrl ] [ --pid PID ] [ -jmxuser userName ] [ -jmxpassword password ] [ -
jmxlocal ] [ --version ] [ [ --help ] | [ -h ] | [ -? ] ] { brokerName }
```

ARGUMENTS

Table 3.3, “[activemq:bstat Arguments](#)” describes the command's arguments.

Table 3.3. activemq:bstat Arguments

Argument	Description
<code>--jmxurl <i>URL</i></code>	Sets the JMX URL used to locate brokers.
<code>--pid <i>PID</i></code>	Set the pid to connect to (only on Sun JVM).
<code>--jmxuser <i>user</i></code>	Sets the JMX user, used for authentication.
<code>--jmxpassword <i>password</i></code>	Sets the JMX password, used for authentication.
<code>--jmxlocal</code>	Use the local JMX server instead of a remote server.
<code>--version</code>	Displays the version information.
<code>-h, -?, --help</code>	Displays the online help for this command.
<i>brokerName</i>	The name of the broker

NAME

activemq:list — lists all available brokers in the specified JMX context

SYNOPSIS

```
activemq:list [ --jmxurl JMXUrl] [ --pid PID] [ -jmxuser userName] [ -jmxpassword password] [ -jmxlocal ] [ --version ] [ --help ] [ -h ] [ -? ]
```

ARGUMENTS

Table 3.4, “[activemq:list Arguments](#)” describes the command's arguments.

Table 3.4. `activemq:list` Arguments

Argument	Interpretation
<code>--jmxurl <i>URL</i></code>	Sets the JMX URL to connect to
<code>--pid <i>PID</i></code>	Set the pid to connect to (only on Sun JVM).
<code>--jmxuser <i>user</i></code>	Sets the JMX user, used for authentication
<code>--jmxpassword <i>password</i></code>	Sets the JMX password, used for authentication
<code>--jmxlocal</code>	Specifies to use the local JMX server instead of a remote server
<code>--version</code>	Displays the version information
<code>-h, -?, --help</code>	Displays the online help for this command

NAME

`activemq:purge`, `purge` — purges messages from a destination

SYNOPSIS

```
activemq:purge [ --msgsel { msgsel ... } ] [ --pid PID] [ --jmxurl JMXUrl] [ -jmxuser userName] [ -jmxpassword password] [ -jmxlocal ] [ --version ] [ --help ] [ -h ] [ -? ] { destName }
```

ARGUMENTS

Table 3.5, “[activemq:purge Arguments](#)” describes the command's arguments.

Table 3.5. `activemq:purge` Arguments

Option	Interpretation
<code>--msgsel <i>msgsel1, msgsel2, ...</i></code>	Purges messages matched by the message selector. See the section called “Message filters” .
<code>--jmxurl <i>URL</i></code>	Sets the JMX URL used to locate the broker.
<code>--pid <i>PID</i></code>	Set the pid to connect to (only on Sun JVM).
<code>--jmxuser <i>user</i></code>	Sets the JMX user, used for authentication.

Option	Interpretation
<code>--jmxpassword <i>password</i></code>	Sets the JMX password, used for authentication.
<code>--jmxlocal</code>	Specifies to use the local JMX server instead of a remote server
<code>--version</code>	Displays the version information
<code>-h, -?, --help</code>	Displays the online help for this command
<i>destName</i>	The specified message destination(s)

EXAMPLES

The following command purges all the messages in the queue **TEST.FOO** on a broker:

```
JBossA-MQ:karaf@root>activemq:purge TEST.FOO
```

The following command purges any messages with an ID ending in **10**:

```
JBossA-MQ:karaf@root>activemq:purge --msgsel JMSMessageID='*:10' TEST.FOO
```

The following command purges messages with a priority of **3**, enter:

```
JBossA-MQ:karaf@root>activemq:purge --msgsel JMSPriority=3 TEST.FOO
```

The message selectors from the preceding two examples can be combined as follows:

```
JBossA-MQ:karaf@root>activemq:purge --msgsel
JMSMessageID='*:10',JMSPriority=3 TEST.FOO
```

NAME

`activemq:query`, `query` — queries the for broker information on specific objects

SYNOPSIS

```
activemq:query [-QMBeanType=name] [-xQMBeanType=name] [--objname query] [--xobjname
query] [--view { attr ...}] [--jmxurl JMXUrl] [--pid PID] [--jmxuser userName] [--jmxpassword password
] [--jmxlocal] [--version] [--help] | [-h] | [-? ]
```

ARGUMENTS

Table 3.6, “[activemq:query Arguments](#)” describes the command's arguments.

Table 3.6. activemq:query Arguments

Argument	Interpretation
----------	----------------

Argument	Interpretation
-Q <i>type=name</i>	Adds to the search list the specific object type matched by the defined object identifier.
-xQ <i>type=name</i>	Removes from the search list the specific object type matched by the object identifier.
--objname <i>query</i>	Adds to the search list objects matched by the query similar.
--xobjname <i>query</i>	Removes from the search list objects matched by the query.
--view <i>attr1,attr2,...</i>	Selects the specific attribute of the object to view. By default, all attributes are displayed.
--jmxurl <i>URL</i>	Sets the JMX URL to connect to.
--pid <i>PID</i>	Set the pid to connect to (only on Sun JVM).
--jmxuser <i>user</i>	Sets the JMX user, used for authentication
--jmxpassword <i>password</i>	Sets the JMX password, used for authentication
--jmxlocal	Specifies to use the local JMX server instead of a remote server
--version	Displays the version information
-h, -?, --help	Displays the online help for this command

EXAMPLES

The following command displays all attributes and object name information for all registered MBeans in the default JMX context:

```
JBossA-MQ:karaf@root>activemq:query
```

The following command displays all attributes and object name information of the destination topic **TEST.FOO**:

```
JBossA-MQ:karaf@root>activemq:query -QTopic=TEST.FOO
```

The following command displays all the brokers in a context whose name ends in **host**:

```
JBossA-MQ:karaf@root>activemq:query -QBroker=*host
```

The following command displays all attributes and object name information for all registered queues:

```
JBossA-MQ:karaf@root>activemq:query --objname=*Queue*
```



```
JBossA-MQ:karaf@root>activemq:query -QQueue=*`
```

The following command displays all attributes and object name information for all topics ending with **.FOO** except those that also begin with **ActiveMQ.Advisory.**:

```
JBossA-MQ:karaf@root>activemq:query -QTopic=*.FOO -  
xQTopic=ActiveMQ.Advisory.*`
```

CHAPTER 4. ADMIN CONSOLE COMMANDS

The **admin** commands allow you to create, manage and destroy container instances.

Type **admin:** then press **Tab** at the **FuseMQkaraf:karaf@root>** prompt to view the available commands.

NAME

admin:change-opts, **change-opts** — changes the Java options of an existing container

SYNOPSIS

```
admin:change-opts [ --help ] { name } { opts }
```

ARGUMENTS

Table 4.1, “**admin:change-opts Arguments**” describes the command's arguments.

Table 4.1. admin:change-opts Arguments

Argument	Interpretation
--help	Displays the online help for this command
<i>name</i>	The name of the container for which you want to change the Java options
<i>opts</i>	The Java options to change

NAME

admin:change-rmi-registry-port, **change-rmi-registry-port** — changes the RMI registry port used by the management layer of a container

SYNOPSIS

```
admin:change-rmi-registry-port [ --help ] { name } { port }
```

ARGUMENTS

Table 4.2, “**admin:change-rmi-registry-port Arguments**” describes the command's arguments.

Table 4.2. admin:change-rmi-registry-port Arguments

Argument	Interpretation
--help	Displays the online help for this command
<i>name</i>	The name of the container instance for which you want to change the port
<i>port</i>	The new RMI registry port

NAME

`admin:change-rmi-server-port`, `change-rmi-server-port` — changes the RMI server port used by the management layer of a container

SYNOPSIS

```
admin:change-rmi-server-port [ --help ] { name } { port }
```

ARGUMENTS

Table 4.3, “[admin:change-rmi-server-port Arguments](#)” describes the command's arguments.

Table 4.3. `admin:change-rmi-server-port Arguments`

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>name</i>	The name of the container instance for which you want to change the port
<i>port</i>	The new RMI server port

NAME

`admin:change-ssh-port`, `changessh-port` — changes the secure shell port of a container

SYNOPSIS

```
admin:change-ssh-port [ --help ] { name } { port }
```

ARGUMENTS

Table 4.4, “[admin:change-ssh-port Arguments](#)” describes the command's arguments.

Table 4.4. `admin:change-ssh-port Arguments`

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>name</i>	The name of the container instance for which you want to change the port
<i>port</i>	The new secure shell port

NAME

`admin:clone`, `clone` — clones an existing container instance

SYNOPSIS

```
admin:clone [ --help ] [ [-l ] | [ --location ] fileName ] [ [-o ] | [ --java-opts ] JVMOpts ] [ [-s ] | [ --ssh-port ] port ] [ [-rs ] | [ --rmi-server-port ] port ] [ [-r ] | [ -rr ] | [ --rmi-port ] | [ --rmi-registry-port ] port ] [ [-v ] | [ --verbose ] ] { name } { cloneName }
```

ARGUMENTS

Table 4.5, “[admin:clone Arguments](#)” describes the command's arguments.

Table 4.5. admin:clone Arguments

Argument	Interpretation
--help	Displays the online help for this command
-l, --location	Location of the cloned container instance in the file system.
-o, --java-opts	JVM options to use when launching the cloned instance.
-s, --ssh-port	Port number for remote secure shell connection.
-rs, --rmi-server-port	Port number for RMI server connection.
-r, -rr, --rmi-port, --rmi-registry-port	Port number for RMI registry connection.
-v, --verbose	Display actions performed by the command (disabled by default).
<i>name</i>	Name of the original container instance.
<i>cloneName</i>	Name of the cloned container instance.

NAME

admin:connect, connect — connects to an existing container

SYNOPSIS

```
admin:connect [ --help ] [ [-u ] | [ --username ] userName ] [ [-p ] | [ --password ] password ] { container } [ command ]
```

ARGUMENTS

Table 4.6, “[admin:connect Arguments](#)” describes the command's arguments.

Table 4.6. admin:connect Arguments

Argument	Interpretation
--help	Displays the online help for this command
-u, --username	The remote user name; the default is karaf
-p, --password	The remote user password; the default is karaf
container	The container to connect to
command	Command to execute on connecting

NAME

`admin:create`, `create` — creates a new child container

SYNOPSIS

```
admin:create [ --help ] [ -l ] [ --location ] filePath [ -furl ] [ --featureURL ] URL... [ -f ] [ --feature ] feature... [ -s ] [ --ssh-port ] SSHPort [ -rs ] [ --rmi-server-port ] RMIServPort [ -r ] [ -rr ] [ --rmi-registry-port ] [ --rmi-port ] RMIRegPort [ -o ] [ --java-opts ] javaOpts { name }
```

ARGUMENTS

Table 4.7, “`admin:create` Arguments” describes the command's arguments.

Table 4.7. `admin:create` Arguments

Argument	Interpretation
--help	Displays the online help for this command
-l, --location	The location of the child's data folders on the file system. By default, the child's data is added to the InstallDir/instances/name directory
-furl, --featureURL	Registers additional feature URLs with the child.
-f, --feature	Specifies additional features loaded by the child.
-s, --ssh-port	The port number for remote secure shell connection
-rs, --rmi-server-port	The port number for RMI server connection
-r, -rr, --rmi-registry-port, --rmi-port	The port number for RMI registry connection
-o, --java-opts	JVM options to use when launching the child

Argument	Interpretation
<i>name</i>	The name of the child

NAME

admin:destroy, destroy — destroys a child container

SYNOPSIS

```
admin:destroy [ --help ] { name }
```

ARGUMENTS

Table 4.8, “[admin:destroy Arguments](#)” describes the command's arguments.

Table 4.8. admin:destroy Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>name</i>	The name of the container to destroy

NAME

admin:list — list all of the child containers on the current host

SYNOPSIS

```
admin:list [ --help ] [ [-l] | [ --location ] filePath ] [ [-o] | [ --java-opts ] javaOpts ]
```

ARGUMENTS

Table 4.9, “[admin:list Arguments](#)” describes the command's arguments.

Table 4.9. admin:list Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-l, --location</code>	Displays the location of the container instances
<code>-o, --java-opts</code>	Displays the options used when launching the container's JVM

NAME

admin:rename, rename — renames a child container

SYNOPSIS

```
admin:rename [ --help ] { name } { new-name }
```

ARGUMENTS

Table 4.10, “[admin:rename Arguments](#)” describes the command's arguments.

Table 4.10. admin:rename Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>name</i>	Current name of the container
<i>new-name</i>	The new name for the container

NAME

admin:start — starts a child container

SYNOPSIS

```
admin:start [ --help ] [ -o ] [ --java-opts ] javaOpts { name }
```

ARGUMENTS

Table 4.11, “[admin:start Arguments](#)” describes the command's arguments.

Table 4.11. admin:start Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-o, --java-opts</code>	The Java options used when launching the container
<i>name</i>	The name of the container to start

NAME

admin:stop — stops a child container

SYNOPSIS

```
admin:stop [ --help ] { name }
```

ARGUMENTS

Table 4.12, “[admin:stop Arguments](#)” describes the command's arguments.

Table 4.12. admin:stop Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>name</i>	The name of the container to start

CHAPTER 5. CAMEL CONSOLE COMMANDS

The `camel` commands are used for managing Camel contexts and routes.

NAME

`camel:context-info` — display detailed information about the specified Camel context

SYNOPSIS

```
camel:context-info [ --help ] { contextName }
```

ARGUMENTS

This command takes the following arguments.

Table 5.1. camel:context-info Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>contextName</code>	The name of the Camel context.

NAME

`camel:context-list` — list all active Camel contexts

SYNOPSIS

```
camel:context-list [ --help ]
```

ARGUMENTS

This command takes the following arguments.

Table 5.2. camel:context-list Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

NAME

`camel:context-start` — start up the specified Camel context

SYNOPSIS

```
camel:context-start [ --help ] { contextName }
```

ARGUMENTS

This command takes the following arguments.

Table 5.3. camel:context -start Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>contextName</code>	The name of the Camel context.

NAME

camel:context-stop — stop the specified Camel context

SYNOPSIS

```
camel:context-stop [ --help ] { contextName }
```

ARGUMENTS

This command takes the following arguments.

Table 5.4. camel:context -stop Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>contextName</code>	The name of the Camel context.

NAME

camel:endpoint-list — lists all deployed Camel endpoints

SYNOPSIS

```
camel:endpoint-list [ --help ]
```

ARGUMENTS

This command takes the following arguments.

Table 5.5. camel:endpoint -list Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

NAME

camel:route-info — display detailed information about the specified Camel route

SYNOPSIS

```
camel:route-info [ --help ] { route } [ contextName ]
```

ARGUMENTS

This command takes the following arguments.

Table 5.6. camel:route-info Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>route</code>	The Camel route ID.
<code>contextName</code>	<i>(Optional)</i> The Camel context name.

NAME

camel:route-list — list the Camel routes

SYNOPSIS

```
camel:route-list [ --help ] [ contextName ]
```

DESCRIPTION

You can optionally restrict the listing to show only the routes belonging to the specified Camel context.

ARGUMENTS

This command takes the following arguments.

Table 5.7. camel:route-list Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>contextName</code>	<i>(Optional)</i> The Camel context name.

NAME

camel:route-resume — resume the specified Camel route (which was previously suspended)

SYNOPSIS

```
camel:route-resume [ --help ] { route } [ contextName ]
```

ARGUMENTS

This command takes the following arguments.

Table 5.8. camel:route-resume Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>route</code>	The Camel route ID.
<code>contextName</code>	<i>(Optional)</i> The Camel context name.

NAME

`camel:route-show` — display the Camel route definition in XML format

SYNOPSIS

```
camel:route-show [ --help ] { route } [ contextName ]
```

ARGUMENTS

This command takes the following arguments.

Table 5.9. camel:route-show Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>route</code>	The Camel route ID.
<code>contextName</code>	<i>(Optional)</i> The Camel context name.

NAME

`camel:route-start` — start the specified Camel route

SYNOPSIS

```
camel:route-start [ --help ] { route } [ contextName ]
```

ARGUMENTS

This command takes the following arguments.

Table 5.10. camel:route-start Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-n,--interval</code>	

Argument	Interpretation
<i>route</i>	The Camel route ID.

NAME

camel:route-stop — stop the specified Camel route

SYNOPSIS

camel:route-stop [--help] { *route* } [*contextName*]

ARGUMENTS

This command takes the following arguments.

Table 5.11. camel:route-stop Arguments

Argument	Interpretation
--help	Displays the online help for this command
<i>route</i>	The Camel route ID.
<i>contextName</i>	(Optional) The Camel context name.

NAME

camel:route-suspend — suspend the specified Camel route

SYNOPSIS

camel:route-suspend [--help] { *route* } [*contextName*]

ARGUMENTS

This command takes the following arguments.

Table 5.12. camel:route-suspend Arguments

Argument	Interpretation
--help	Displays the online help for this command
<i>route</i>	The Camel route ID.
<i>contextName</i>	(Optional) The Camel context name.

CHAPTER 6. CONFIG CONSOLE COMMANDS

The config commands are used for managing container configuration. The configuration data is edited in two stages. First the changes are queued until they are dynamically loaded into the container by executing the **config:update** command. A copy of the configuration is persisted to the file system in the container's **etc** folder.

When editing a configuration the commands are used as follows:

1. Start the editing session for the specified configuration.

config:edit

2. Edits, or creates, a configuration.

- **config:proplist**

Lists the properties in the configuration.

- **config:propappend**

Append a new property to the configuration.

- **config:propset**

Sets the value for a configuration property.

- **config:propdel**

Deletes a property from the configuration.

3. **config:update**

Saves the changes and updates the containers using the configuration.

You can abandon an editing session using **config:cancel**.

Type **config:** then press **Tab** at the prompt to view the available commands.

NAME

config:cancel — cancels the changes to the configuration being edited

SYNOPSIS

config:cancel [--help]

DETAILS

When editing a configuration, the changes are buffered until the editing session is closed. The **config:cancel** command clears the buffer without saving the changes and closes the editing session. You can see a list of the buffered changes using the **jaas:pending** command.

ARGUMENTS

[Table 6.1, “config:cancel Arguments”](#) describes the command's arguments.

Table 6.1. config:cancel Arguments

Option	Interpretation
--help	Displays the online help for this command

NAME

`config:delete`, `delete` — deletes a configuration from the container

SYNOPSIS

```
config:delete [ --help ] [[ -f ] | [ --use-file ]] [ --no-delete-cfg-file ] { pid }
```

DETAILS

When you delete a configuration, the change is made directly on the running container. Any properties set in the configuration are reverted to their default values and the behavior of the container will be immediate.

If you use the **--no-delete-cfg-file** argument, the original settings can be reloaded from the configuration file.

ARGUMENTS

Table 6.2, “[config:delete Arguments](#)” describes the command’s arguments.

Table 6.2. config:delete Arguments

Option	Interpretation
--help	Displays the online help for this command.
-f, --use-file	Use a filename instead of the PID to locate the configuration.
--no-delete-cfg-file	Does not delete the associated configuration file from the container’s etc folder.
<i>pid</i>	Specifies the configuration’s persistent identifier.

NAME

`config:edit`, `edit` — begins an editing session for a configuration. If the configuration does not exist a new configuration is created.

SYNOPSIS

```
config:edit [ --help ] [ --force ] [[ -f ] | [ --use-file ]] { pid }
```

DETAILS

The **config:edit** command is the first step in editing a container configuration. It opens the configuration so that calls to the **config:*** editing commands will update the selected configuration. The edits made by the **config:*** editing commands are placed in a buffer associated with the selected

configuration and not propagated to the container, or the file system, until the editing session is ended by the **config:update** command.

If you use the **config:edit** command before saving the changes to a configuration that is open for editing, the changes to the previously open configuration are abandoned. The pending edits cleared without being saved.

ARGUMENTS

Table 6.3, “**config:edit Arguments**” describes the command's arguments.

Table 6.3. config:edit Arguments

Option	Interpretation
--help	Displays the online help for this command
--force	Forces the editing of this configuration, even if another configuration was being edited
-f, --use-file	Use a filename instead of the PID to locate the configuration
<i>pid</i>	The persistent identifier of the configuration

NAME

config:list — lists the existing configurations for the container

SYNOPSIS

```
config:list [ --help ] [ query ]
```

ARGUMENTS

Table 6.4, “**config:list Arguments**” describes the command's arguments.

Table 6.4. config:list Arguments

Argument	Interpretation
--help	Displays the online help for this command
<i>query</i>	An LDAP query

NAME

config:propappend, **propappend** — appends the given value to an existing property or creates the property with the specified name and value

SYNOPSIS

```
config:propappend [ --help ] [[ -b ] | [ --bypass-storage ] ] [[ -p PID ] | [ --pid PID ] ] { name } { value }
```


DETAILS

When you append a value to a property using the **config:propappend** command, the change is stored in the buffer and not propagated to the container until the editing session is closed.

If you use the **-p** argument to specify a PID, however, the change is made immediately.

ARGUMENTS

Table 6.5, “**config:propappend Arguments**” describes the command's arguments.

Table 6.5. config:propappend Arguments

Argument	Interpretation
--help	Displays the online help for this command.
-b, --bypass-storage	Do not write the change to the local file.
-p, --pid	Specifies the PID of the configuration in which to make the change. The default is to change the configuration currently open for editing.
<i>name</i>	Specifies the name of the property to change.
<i>value</i>	Specifies the value to append to the property.

NAME

config:propdel, propdel — deletes a property from the configuration being edited

SYNOPSIS

```
config:propdel [ --help ] [ [ -b ] | [ --bypass-storage ] ] [ [ -p PID ] | [ --pid PID ] ] { name }
```

DETAILS

When you delete a property using the **config:propdel** command, the change is stored in the buffer and not propagated to the container until the editing session is closed.

If you use the **-p** argument to specify a PID, however, the change is made immediately.

ARGUMENTS

Table 6.6, “**config:propdel Arguments**” describes the command's arguments.

Table 6.6. config:propdel Arguments

Argument	Interpretation
--help	Displays the online help for this command.
-b, --bypass-storage	Does not write the change to the local file.
-p, --pid	Specifies the PID of the configuration in which to make the change. The default is to change the configuration currently open for editing.

Argument	Interpretation
<i>name</i>	Specifies the name of the property to delete.

NAME

`config:proplist`, `proplist` — lists the properties in the configuration being edited

SYNOPSIS

`config:proplist` [`--help`] [[`-p` *PID*] | [`--pid` *PID*]]

ARGUMENTS

Table 6.7, “[config:proplist Arguments](#)” describes the command's arguments.

Table 6.7. `config:proplist` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-p</code> , <code>--pid</code>	The PID of the configuration in which to make the change

NAME

`config:propset`, `propset` — sets a property in the configuration being edited

SYNOPSIS

`config:propset` [`--help`] [[`-b`] | [`--bypass-storage`]] [[`-p` *PID*] | [`--pid` *PID*]] { *name* } { *value* }

DETAILS

When you set a property using the `config:propset` command, the change is stored in the buffer and not propagated to the container until the editing session is closed.

If you use the `-p` argument to specify a PID, however, the change is made immediately.

ARGUMENTS

Table 6.8, “[config:propset Arguments](#)” describes the command's arguments.

Table 6.8. `config:propset` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>-b</code> , <code>--bypass-storage</code>	Does not write the change to the local file.

Argument	Interpretation
-p, --pid	Specifies the PID of the configuration in which to make the change. The default is to change the configuration currently open for editing.
<i>name</i>	Specifies the name of the property to set.
<i>value</i>	Specifies the value to set for the property.

NAME

`config:update` — saves the changes made to the configuration being edited and propagates them to the container

SYNOPSIS

```
config:propset [ --help ] [ [ -b ] | [ --bypass-storage ] ]
```

ARGUMENTS

[Table 6.9, “`config:update` Arguments”](#) describes the command's arguments.

Table 6.9. `config:update` Arguments

Argument	Interpretation
--help	Displays the online help for this command
-b, --bypass-storage	Do not update the copy of the configuration on the file system

CHAPTER 7. CXF CONSOLE COMMANDS

The **cxf** commands are used for managing Apache CXF buses and endpoints. If these commands are not already loaded into the console, you can load them using the following console command:

```
JBossFuse:karaf@root> features:install cxf- osgi
```

NAME

cxf:list-busses — lists all Apache CXF buses

SYNOPSIS

```
cxf:list-busses [ --help ]
```

ARGUMENTS

This command takes the following arguments.

Table 7.1. cxf:list-busses Arguments

Argument	Interpretation
--help	Displays the online help for this command

NAME

cxf:list-endpoints — list all active endpoints belonging to the specified Apache CXF bus

SYNOPSIS

```
cxf:list-endpoints [ --help ] { busID }
```

ARGUMENTS

This command takes the following arguments.

Table 7.2. cxf:list-endpoints Arguments

Argument	Interpretation
--help	Displays the online help for this command
<i>busID</i>	The Apache CXF bus ID.

NAME

cxf:start-endpoint — start the specified Apache CXF endpoint belonging to the specified bus

SYNOPSIS

```
cxf:start-endpoint [ --help ] { busID } { endpointName }
```

ARGUMENTS

This command takes the following arguments.

Table 7.3. cxf:start-endpoint Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>busID</i>	The Apache CXF bus ID.
<i>endpointName</i>	The name of the Apache CXF endpoint.

NAME

`cxf:stop-endpoint` — stop the specified Apache CXF endpoint belonging to the specified bus

SYNOPSIS

```
cxf:stop-endpoint [ --help ] { busID } { endpointName }
```

ARGUMENTS

This command takes the following arguments.

Table 7.4. cxf:stop-endpoint Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>busID</i>	The Apache CXF bus ID.
<i>endpointName</i>	The name of the Apache CXF endpoint.

CHAPTER 8. DEV CONSOLE COMMANDS

The dev commands are a collection of utilities that are useful testing bundles in the container.

Type **dev**: then press **Tab** at the prompt to view the available commands.

NAME

dev:classloaders, classloaders — displays a list of leaking bundle classloaders

SYNOPSIS

dev: classloaders [--help]

ARGUMENTS

Table 8.1, “[dev:classloader Arguments](#)” describes the commands arguments.

Table 8.1. dev: classloader Arguments

Argument	Interpretation
--help	Displays the online help for this command

NAME

dev:create-dump, create-dump — creates a ZIP file containing diagnostic information

SYNOPSIS

dev: create-dump [--help] [[-d *dumpFolder*] | [--directory *dumpFolder*]] { *dumpName* }

ARGUMENTS

Table 8.2, “[dev:create-dump Arguments](#)” describes the commands arguments.

Table 8.2. dev: create-dump Arguments

Argument	Interpretation
--help	Displays the online help for this command
-d, --directory	Specifies the folder into which to store the dump
<i>dumpName</i>	Specifies the name for the dump file

NAME

dev:dynamic-import, dynamic-import — enables/disables dynamic imports for a bundle

SYNOPSIS

```
dev:dynamic-import [ --help ] { bundleID }
```

ARGUMENTS

Table 8.3, “[dev:dynamic-import Arguments](#)” describes the commands arguments.

Table 8.3. dev:dynamic-import Arguments

Argument	Interpretation
--help	Displays the online help for this command
<i>bundleID</i>	A bundle ID.

NAME

dev:framework, framework — enables/disables debugging for an OSGi framework

SYNOPSIS

```
dev:framework [ --help ] { [[ -debug ] | [ --enable-debug ] ] [ [-nodebug ] | [ --disable-debug ] ] } { framework }
```

ARGUMENTS

Table 8.4, “[dev:framework Arguments](#)” describes the commands arguments.

Table 8.4. dev:framework Arguments

Argument	Interpretation
--help	Displays the online help for this command
-nodebug, --disable-debug	Disable debugging for the OSGi framework.
-debug, --enable-debug	Enable debugging for the OSGi framework.
<i>framework</i>	Name of the OSGi framework

NAME

dev:print-stack-traces, print-stack-traces — enables/disables printing of full stack traces in the console when the execution of a command throws an exception

SYNOPSIS

```
dev:print-stack-traces [ --help ] [ false ]
```

ARGUMENTS

Table 8.5, “[dev:print-stack-traces Arguments](#)” describes the commands arguments.

Table 8.5. `dev:print-stack-traces` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>false</code>	Disables stack traces

NAME

`dev:restart` — restart the container

SYNOPSIS

`dev:restart` [`--help`] [`-c`] [`--clean`]

ARGUMENTS

Table 8.6, “`dev:restart` Arguments” describes the commands arguments.

Table 8.6. `dev:restart` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-c</code> , <code>--clean</code>	Force a clean (cold) restart by deleting the container's data directory.

NAME

`dev:show-tree`, `show-tree` — shows the tree of bundles based on the wiring information

SYNOPSIS

`dev:show-tree` [`--help`] { *bundleID* }

ARGUMENTS

Table 8.7, “`dev:show-tree` Arguments” describes the commands arguments.

Table 8.7. `dev:show-tree` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>bundleID</i>	A bundle ID.

NAME

dev:threads, threads — shows the threads in the JVM

SYNOPSIS

```
dev:threads [ --help ] [ -f ] [ --flat ]
```

ARGUMENTS

Table 8.8, “[dev:threads Arguments](#)” describes the commands arguments.

Table 8.8. dev:threads Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-f, --flat</code>	Do not show the threads in a tree

NAME

dev:wait-for-service, wait-for-service — wait for the specified OSGi service

SYNOPSIS

```
dev:wait-for-service [ --help ] [ -t ] [ --timeout ] timeout [ -e ] [ --exception ] {
serviceClassOrFilter }
```

DESCRIPTION

This command is useful when you are developing a console script and you want to wait for a specific OSGi service to start up, before proceeding with the execution of the script.

For example, the various command sets installed in the console (**shell:***, **admin:***, **features:***, and so on) are represented by OSGi services of type, **org.apache.karaf.shell.console.SubShell**.

If you want to check that a sub-shell service is available, you could enter the following console command:

```
karaf@root> dev:wait-for-service -t 1000
org.apache.karaf.shell.console.SubShell
true
```

This form of the command is not very useful in this case, because there are many different instances of the **SubShell** service installed in the container. To be more specific, you can define an LDAP filter, which specifies one or more service property values. For example, you can wait specifically for the **osgi** sub-shell service by entering a command like the following:

```
karaf@root> dev:wait-for-service -t 1000 &
(objectClass=org.apache.karaf.shell.console.SubShell)(name=osgi)
true
```

ARGUMENTS

Table 8.9, “[dev:wait-for-service Arguments](#)” describes the commands arguments.

Table 8.9. dev:wait-for-service Arguments

Argument	Interpretation
--help	Displays the online help for this command
-t, --timeout	Timeout (specified in milliseconds: negative to not wait at all, zero to wait forever). Default is forever.
-e, --exception	Throw an exception if the wait command times out (the service is not found). Default is false.
<i>serviceClassOrFilter</i>	Specifies the OSGi service either by the service's class name or by an LDAP-style filter (which is applied to the OSGi service's properties).

NAME

`dev:watch`, `watch` — watches and automatically updates bundles

SYNOPSIS

`dev:watch` [`--help`] [[`--start`] | [`--stop`]] [`-i interval`] [`--list`] [`--remove`] { *bundles ...* }

ARGUMENTS

Table 8.10, “`dev:watch` Arguments” describes the commands arguments.

Table 8.10. `dev:watch` Arguments

Argument	Interpretation
--help	Displays the online help for this command
--stop	Stop watching the specified bundles
--start	Start watching the specified bundles
-i	Specifies the interval, in milliseconds, to check the bundles.
--list	List the bundles being watched.
--remove	Remove the specified bundles from the watch list.
<i>bundles...</i>	Specifies a whitespace delimited list of bundle URLs or bundle IDs.



IMPORTANT

Only Maven URLs and Maven snapshots will be updated automatically. So, if you run

```
JBossA-MQ:karaf@root> dev:watch *
```

You are monitoring all bundles that have a location matching `mvn:*` that have `-SNAPSHOT` in their URL.

CHAPTER 9. FUSE APPLICATION BUNDLE(FAB) CONSOLE COMMANDS

The **fab** commands are used for managing Fuse Application Bundle(FAB)s.

There is no dedicated install command for FABs. To install a FAB, use the **osgi:install** command combined with the **fab:** URL prefix. For example, to install the FAB **mvn:org.fusesource.example/camel-example/1.0** use the following console command:

```
JBossA-MQ:karaf@root> osgi:install fab:mvn:org.fusesource.example/camel-example/1.0
```

NAME

fab:headers — displays the headers of a FAB

SYNOPSIS

```
fab:headers [ --help ] [ --indent style ] { URL }
```

DESCRIPTION

Displays the header entries from the **META-INF/MANIFEST.MF** file embedded in the FAB JAR file. This is *not* the same thing as the bundle headers returned by the **osgi:headers** command, because the **osgi:headers** command shows the effective headers *after* the FAB is converted into an OSGi bundle. For example, a typical FAB might have headers like the following:

```
JBossA-MQ:karaf@root> fab:headers
mvn:org.fusesource.examples/cbr/7.0.0.fuse-beta-042

Manifest-Version = 1.0 Archiver-Version = Plexus Archiver Built-By =
username Build-Jdk = 1.6.0_29 Created-By = Apache Maven
```

After the FAB is deployed, the corresponding OSGi bundle could have headers like the following (given that the bundle ID of the deployed FAB is 228):

```
JBossA-MQ:karaf@root> osgi:headers 228

org.fusesource.examples.cbr (228) -----
Manifest-Version = 1 Bnd-LastModified = 1334306872960 Archiver-Version =
Plexus Archiver Tool = Bnd-1.43.0 Originally-Created-By = Apache Maven
FAB-URL = mvn:org.fusesource.examples/cbr/7.0.0.fuse-beta-042 Generated-
By-FAB-From = mvn:org.fusesource.examples/cbr/7.0.0.fuse-beta-042 Built-By
= username FAB-Id = org.fusesource.examples:cbr:7.0.0.fuse-beta-042:jar
Build-Jdk = 1.6.0_29 Created-By = 1.6.0_29 (Apple Inc.) Bundle-Name =
org.fusesource.examples.cbr Bundle-SymbolicName =
org.fusesource.examples.cbr Bundle-Version = 7.0.0.fuse-beta-042 Bundle-
ManifestVersion = 2 Export-Package = OSGI-INF.blueprint, OSGI-INF
```

ARGUMENTS

[Table 9.1, “fab:headers Arguments”](#) describes the commands arguments.

Table 9.1. fab:headers Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>--indent</code>	Specify the indent style. Valid values are 1 , 2 , or 3 . The default is -1 .
<i>URL</i>	The URL of the FAB

NAME

`fab:info` — display information about a FAB, including the list of shared and unshared dependencies, and the list of features installed as part of the FAB resolution process

SYNOPSIS

```
fab:info [ --help ] { bundleID }
```

ARGUMENTS

Table 9.2, “`fab:info` Arguments” describes the commands arguments.

Table 9.2. `fab:info` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>bundleID</i>	The bundle ID of the FAB (for deployed FABs) or the URL location of the FAB (for FABs that are not yet deployed).

NAME

`fab:start` — starts the specified FAB

SYNOPSIS

```
fab:start [ --help ] [ --timeout millis ] { bundleID }
```

DESCRIPTION

Depending on how a FAB is configured, it can be associated with multiple dependent bundles. When a FAB is initially installed in the container, the transitive dependencies of the FAB are determined by scanning the embedded POM file, `META-INF/maven/GroupID/ArtifactID/pom.xml`. Any transitive dependencies that are shared (for example, by being marked as provided or because the dependency is already packaged as an OSGi bundle), are deployed as separate OSGi bundles in the container.

When you start the FAB using `fab:start`, the runtime attempts to start *all* of the corresponding bundles, starting with the leaves of the dependency tree and working its way up the tree to the FAB's bundle. In particular, this implies that any OSGi services, blueprint XML files, and Spring XML files in the

dependent OSGi bundles are activated in the appropriate order.

ARGUMENTS

Table 9.3, “[fab: start Arguments](#)” describes the commands arguments.

Table 9.3. fab: start Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>--timeout</code>	Specifies, in milliseconds, how long to wait for the FAB bundle to start up. Default is 30000 .
<i>bundleID</i>	The bundle ID of the FAB.

NAME

`fab:stop` — stops the specified FAB bundle together with its shared transitive dependencies, except for those dependencies that are being used by other bundles.

SYNOPSIS

```
fab:stop [ --help ] { bundleID }
```

ARGUMENTS

Table 9.4, “[fab: stop Arguments](#)” describes the commands arguments.

Table 9.4. fab: stop Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>bundleID</i>	The bundle ID of the FAB.

NAME

`fab:tree`, `tree` — displays the dependency tree of a FAB

SYNOPSIS

```
fab:tree [ --help ] { bundleID }
```

ARGUMENTS

Table 9.5, “[fab: tree Arguments](#)” describes the commands arguments.

Table 9.5. fab: tree Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>bundleID</code>	Specifies the bundle ID, URL, or filename of the FAB.

NAME

`fab:uninstall` — uninstall the specified FAB and all of its transitive dependencies, except for those dependencies that are being used by other bundles

SYNOPSIS

```
fab:uninstall [ --help ] { bundleID }
```

DESCRIPTION

Depending on how a FAB is configured, it can be associated with multiple dependent bundles. When a FAB is initially installed in the container, the transitive dependencies of the FAB are determined by scanning the embedded POM file, `META-INF/maven/GroupID/ArtifactID/pom.xml`. Any transitive dependencies that are shared (for example, by being marked as provided or because the dependency is already packaged as an OSGi bundle), are deployed as separate OSGi bundles in the container.

When you uninstall the FAB using `fab:uninstall`, the runtime attempts to uninstall all of the corresponding OSGi bundles, *except* for any bundles that are still being used by other applications in the container.

ARGUMENTS

[Table 9.6, “`fab:uninstall` Arguments”](#) describes the commands arguments.

Table 9.6. `fab:uninstall` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>bundleID</code>	The bundle ID of the FAB.

CHAPTER 10. FABRIC CONSOLE COMMANDS

10/03/12

Removed `container-change-profile` and replaced it with `container-add-profile` and `container-remove-profile`

This chapter describes `fabric` console commands.

NAME

`fabric:cluster-list`, `cluster-list` — lists the members of a message broker cluster

SYNOPSIS

```
fabric:cluster-list [ --help ] [ Path ]
```

DESCRIPTION

This command lists all message brokers in the fabric. It allows you to see which brokers are grouped into clusters. The resulting list will enable you to see which brokers are participating in a particular cluster.

ARGUMENTS

Table 10.1, “`fabric:cluster-list` Arguments” describes the command's arguments.

Table 10.1. `fabric:cluster-list` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>Path</i>	Specifies the path of the registry node to list. If not specified, all clusters will be listed.

RELATED TOPICS

[fabric:mq-create](#)

NAME

`fabric:cloud-firewall-edit` — manage a cloud container's firewall

SYNOPSIS

```
fabric:cloud-firewall-edit [ --help ] [ --owner owner ] [ --option key=value ]
```

ARGUMENTS

Table 10.2, “`fabric:cloud-firewall-edit` Arguments” describes the command's arguments.

Table 10.2. `fabric:cloud-firewall-edit` Arguments

Argument	Interpretation
<code>--port</code>	The target IP port.

Argument	Interpretation
<code>--flush</code>	Flush all rules.
<code>--revoke</code>	Revoke the rule for the specified port. This blocks access to the specified IP port.
<code>--target-container</code>	The target container name.
<code>--source-container</code>	The source container, which has access granted or revoked.
<code>--target-node-id</code>	The target node ID.
<code>--source-cidr</code>	The source CIDR, which has access granted or revoked.
<code>--provider</code>	The cloud provider name.
<code>--help</code>	Displays the online help for this command.

NAME

`fabric:cloud-service-add` — initialize a cloud provider (which can be used for provisioning containers in the cloud)

SYNOPSIS

```
fabric:cloud-service-add [ --help ] [ --provider providerName ] [ --name name ] [ --api APIName ]
[ --endpoint URL ] [ --identity accessKeyID ] [ --credential secretAccessKey ] [ --owner owner ] [ --option
key=value ] [ --async-registration ]
```

DESCRIPTION

This command runs asynchronously. That is, although the command returns immediately, it runs a thread in the background, which completes the initialization of the cloud provider. You can use `fabric:cloud-service-list` to discover when the initialization has completed.

There are two different styles of usage for this command:

- *Commercial cloud provider*—if you are using a commercial cloud provider, JClouds provides prepackaged modules that encapsulate the basic connection details for the provider. The prepackaged modules are available to install as Karaf features (named `jclouds-ProviderName`) and encapsulate such details as the endpoint URI, cloud API, and so on.

For example, to install an Amazon Web Services (AWS) EC2 cloud provider, you can perform the following steps (assuming you are working in a standalone container):

1. Install the basic set of fabric cloud commands:

```
karaf@root> features:install fabric-jclouds
```

2. Install the JClouds module specifically for AWS EC2:

```
karaf@root> features:install jclouds-aws-ec2
```

3. Add the AWS EC2 provider, specifying the login credentials for your EC2 account:

```
karaf@root> fabric:cloud-service-add --provider aws-ec2 --
identity AccessKeyID
--credential SecretAccessKey
```

4. You are now ready to start creating compute instances on the **aws-ec2** cloud service, using the **fabric:container-create-cloud** command.
- *Private cloud service*—if you are hosting your compute instances on a private cloud service, you must specify the connection details more explicitly, by supplying the **--api** and **--endpoint** options. In this case, you must also define a name for the cloud service, by supplying the **--name** option.

For example, to define a connection to a private cloud service that uses the **openstack-nova** API through the endpoint, **http://172.16.0.1:4000/v2.0/**, you can perform the following steps (assuming you are working in a standalone container):

1. Install the basic set of fabric cloud commands:

```
karaf@root> features:install fabric-jclouds
```

2. Install the JClouds module for the **openstack-nova** API:

```
karaf@root> features:install jclouds-api-openstack-nova
```

3. Add the private cloud service, specifying the login credentials, API, and endpoint URL:

```
karaf@root> fabric:cloud-service-add --name myOpenStack --api
openstack-nova
--endpoint http://172.16.0.1:4000/v2.0/ --identity AccessKeyID --
credential SecretAccessKey
```



NOTE

You can provide additional customisation of the connection by setting options through the **--option** flag (which can appear multiple times in the command).

4. You are now ready to start creating compute instances on the **myOpenStack** cloud service, using the **fabric:container-create-cloud** command.

INSTALLING THE COMMAND IN A FABRIC

To access this command from a fabric container, you must have installed the **fabric-jclouds** feature. To install the **fabric-jclouds** feature, deploy the **cloud** profile into the current container, using the **fabric:container-change-profile** command.

For example, if the console is currently logged on to the **root** container of the Fabric, you could add the **cloud** profile as follows:

```
JBossA-MQ:karaf@root> fabric:container-list
[id] [version] [alive] [profiles]
[provision status]
root* 1.0 true fabric, fabric-
ensemble-0000-1 success
JBossA-MQ:karaf@root> fabric:container-change-profile root fabric fabric-
ensemble-0000-1 cloud
JBossA-MQ:karaf@root> fabric:container-list
[id] [version] [alive] [profiles]
[provision status]
root* 1.0 true fabric, fabric-
ensemble-0000-1, cloud success
```

ARGUMENTS

Table 10.3, “[fabric:cloud-service-add Arguments](#)” describes the command's arguments.

Table 10.3. fabric:cloud-service-add Arguments

Argument	Interpretation
--help	Displays the online help for this command.
--provider	The name of a commercial cloud provider (for example, aws-ec2 or rackspace).
--name	The JClouds service context name, which identifies the cloud service uniquely. Defaults to the provider name (as specified by the --provider option).
--api	Specifies the cloud API (for example, ec2 , openstack-nova , or cloudstack).
--endpoint	Specifies the cloud service's endpoint URL.
--identity	The identity used to access the cloud service.
--credential	The credential used to access the cloud service.
--owner	Specifies the EC2 AMI owner, which enables you to use private images (AWS EC2 only).
--option	Provider-specific properties. For example: --option jclouds.regions=us-east-1 . If you want to specify more than one option, specify this option multiple times.
--async-registration	Do not wait for the provider registration (that is, complete the registration in a background thread).

NAME

`fabric:cloud-service-list` — list the configured cloud providers

SYNOPSIS

`fabric:cloud-service-list` [`--help`]

DESCRIPTION

For each configured cloud provider, displays the provider name, type (**compute** or **blobstore**), and registration (**local**, for a standalone container, or **fabric**, for a Fabric Container).

To access this command, the current container must belong to a Fabric and you must have installed the **fabric-jclouds** feature. To install the **fabric-jclouds** feature, deploy the **cloud** profile into the current container, using the **fabric:container-change-profile** command.

For example, if the console is currently logged on to the **root** container of the Fabric, you could add the **cloud** profile as follows:

```
JBossA-MQ:karaf@root> fabric:container-list
[id] [version] [alive] [profiles] [provision status] root* 1.0 true
fabric, fabric-ensemble-0000-1 success
JBossA-MQ:karaf@root> fabric:container-change-profile root fabric fabric-
ensemble-0000-1 cloud
JBossA-MQ:karaf@root> fabric:container-list
[id] [version] [alive] [profiles] [provision status] root* 1.0 true
fabric, fabric-ensemble-0000-1, cloud success
```

ARGUMENTS

Table 10.4, “[fabric:cloud-service-list Arguments](#)” describes the command's arguments.

Table 10.4. `fabric:cloud-service-list` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.

NAME

`fabric:cloud-service-remove` — removes the specified cloud provider

SYNOPSIS

`fabric:cloud-service-remove` [`--help`] { *Name* }

DESCRIPTION

To access this command, the current container must belong to a Fabric and you must have installed the **fabric-jclouds** feature. To install the **fabric-jclouds** feature, deploy the **cloud** profile into the current container, using the **fabric:container-change-profile** command.

For example, if the console is currently logged on to the **root** container of the Fabric, you could add the **cloud** profile as follows:

```
JBossA-MQ:karaf@root> fabric:container-list
```

```

[id]                                [version] [alive] [profiles]
[provision status]
root*                               1.0          true    fabric, fabric-
ensemble-0000-1 success
JBossA-MQ:karaf@root> fabric:container-change-profile root fabric fabric-
ensemble-0000-1 cloud
JBossA-MQ:karaf@root> fabric:container-list
[id]                                [version] [alive] [profiles]
[provision status]
root*                               1.0          true    fabric, fabric-
ensemble-0000-1, cloud success

```

ARGUMENTS

Table 10.5, “[fabric:cloud-service-remove Arguments](#)” describes the command's arguments.

Table 10.5. fabric:cloud-service-remove Arguments

Argument	Interpretation
--help	Displays the online help for this command.
Name	The JClouds service context name, which represents the cloud provider.

NAME

fabric:container-add-profile, container-add-profile — Adds the specified list of profiles to a container

SYNOPSIS

```
fabric:container-add-profile [ --help ] { Name } { Profiles }
```

ARGUMENTS

Table 10.6, “[fabric:container-add-profile Arguments](#)” describes the command's arguments.

Table 10.6. fabric:container-add-profile Arguments

Argument	Interpretation
--help	Displays the online help for this command
Name	Specifies the name of the container.
Profiles	Specifies the list of profiles to add to the container.

NAME

fabric:container-connect, container-connect — connects to a remote Fabric Container and execute the specified command

SYNOPSIS

```
fabric:container-connect [ --help ] [ [-u ] | [ --username ]User ] [ [-p ] | [ --password ]Password ] { ContainerName } [ Command ]
```

DESCRIPTION

This command allows you to connect to any container in the current fabric and execute a command. For example, to execute the `osgi:list` command on the `root2` container, you could enter a console command like [Example 10.1, “Executing a Command in a Remote Container”](#).

Example 10.1. Executing a Command in a Remote Container

```
JBossA-MQ:karaf@root> fabric:container-connect -u YourName -p YourPass
root2 osgi:list
```

This command uses fabric JAAS security to log into the container, so the username and password are managed by the container's JAAS realm.

ARGUMENTS

[Table 10.7, “fabric:container-connect Arguments”](#) describes the command's arguments.

Table 10.7. fabric:container-connect Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-u</code> , <code>--username</code>	Specifies the username for logging on to the remote container. The default is admin .
<code>-p</code> , <code>--password</code>	SPecifies the password for logging on to the remote container. The default is admin .
<i>ContainerName</i>	Specifies the name of the remote container.
<i>Command</i>	Specifies the console command to execute on the remote container.

RELATED TOPICS

[Chapter 12, JAAS Console Commands](#)

NAME

`fabric:container-create`, `container-create` — creates one or more Fabric Containers

SYNOPSIS

```
fabric:container-create [ --help ] [ [ --parent ParentID ] | [ --url URL ] ] [ --proxy-uri ProxyURI ] [ --ensemble-server ] [ --profile ProfileID ] [ --resolver policy ] [ --version Version ] [ --jvm-opts JvmOpts ] { Name } [ Number ]
```

EXAMPLES

This command is a generic container create command. It combines the functionality of the **fabric:container-create-child**, **fabric:container-create-cloud**, and **fabric:container-create-ssh** commands. The type of container that is created, depends on the specified URL.

Child container

To create a child container, specify a URL in the following format:

```
child://ParentName
```

Where *ParentName* is the name of the child's parent container.

Cloud container

To create a cloud container, specify a URL in the following format:

```
jclouds://ProviderId?
imageId=ImageID&locationId=LocationID&group=Group&user=User
```

For a detailed explanation of the options appearing in this URL, see [fabric:container-create-cloud](#).

SSH container

To create an SSH container with username and password credentials, specify a URL in the following format:

```
ssh://User:Password@Host:Port
```

Where *User* and *Password* are the credentials for logging in to the machine at *Host:Port*, through the SSH protocol.

To create an SSH container with username and private key credentials, specify a URL in the following format:

```
ssh://User@Host:Port?privateKeyFile=KeyPath
```

Where *KeyPath* is the pathname of the private key file on the local filesystem.

ARGUMENTS

Table 10.8, “[fabric:container-create Arguments](#)” describes the command's arguments.

Table 10.8. fabric:container-create Arguments

Argument	Interpretation
--help	Displays the online help for this command
--parent	Specifies the parent container's ID.
--url	Specifies the URL of the new container.

Argument	Interpretation
<code>--proxy-uri</code>	Specifies the Maven proxy URI to use.
<code>--ensemble-server</code>	Specifies if the new container should be a Fabric Server.
<code>--profile</code>	Specifies a list of profiles to deploy into the new container.
<code>--resolver</code>	Specifies how the container will report its address to other containers. Valid values are localip , localhostname , publicip , publichostname , manualip . For more information see fabric:container-resolver-set .
<code>--version</code>	Specifies the version of the profiles used by the new container. Defaults to the current default version.
<code>--jvm-opts</code>	Specifies options to pass to the container's JVM.
<i>Name</i>	Specifies the name of the new container. When creating multiple containers, the name serves as a prefix.
<i>Number</i>	Specifies the number of containers that should be created.

RELATED TOPICS

fabric:container-create-child
fabric:container-create-cloud
fabric:container-create-ssh
fabric:container-resolver-list

NAME

`fabric:container-create-child` — create one or more child containers

SYNOPSIS

```
fabric:container-create-child [ --help ] [ --ensemble-server ] [ --profile profileID ] [ --version version ] [ --jvm-opts jvmOpts ] [ --resolver policy ] { parent } { name } [ number ]
```

DESCRIPTION

Child containers have the following characteristics:

- Each child container has a parent, so that the child containers form a hierarchy, with the root container as the ultimate ancestor.
- The child starts in a new JVM instance (JVM options can be passed to the new JVM through the `--jvm-opts` command option).
- A complete set of data directories are created for the child instance, under the `ESBInstallDir/instances/ChildName` directory. The `ESBInstallDir/system` directory is shared with the root container.

For example, if you have already created a new fabric (for example, by invoking `fabric:create`), you could add some child containers to the root container by entering the following command:

```
karaf@root> fabric:container-create-child root child 3
```

This command creates three new children under the `root` container. To check that the containers have been successfully created, invoke the `fabric:container-list` command, as follows:

```
karaf@root> fabric:container-list
[id]                [version] [alive] [profiles]
[provision status]
root                1.0       true   fabric, fabric-
ensemble-0000-1
  child1            1.0       true   default
success
  child2            1.0       true   default
success
  child3            1.0       true   default
success
```

As you can see, the command creates three new child containers, `child1`, `child2`, and `child3`, with the `default` profile. These containers are ordinary (non-ensemble) containers, running fabric agents (ZooKeeper clients).

If you do not explicitly specify any profile (or profiles) for the new child containers, each of the child containers is created with the OSGi bundles required for a minimal Apache Karaf container and all of the profiles and bundles specified by the `default` profile.

To associate multiple profiles with a new child container, you can specify the `--profile` option multiple times. For example, if you want to deploy your own application profile, `myApp`, together with the `esb` profile, you would use a command like the following:

```
fabric:container-create-child --profile esb --profile myApp root
childMyApp
```

SHUTTING DOWN CHILD CONTAINERS

After you create new child containers, the children run as separate processes, independently of the parent. Consequently, when you shut down the parent container, *the child processes continue to run in the background*. If you want to shut down the children, you must explicitly invoke the `fabric:container-stop` command. For example, if a root container has three children—`child1`, `child2`, and `child3`—you can issue the following commands in the root container console to shut down all of the containers:

```
karaf@root> fabric:container-stop child1
karaf@root> fabric:container-stop child2
```

```
karaf@root> fabric:container-stop child3
karaf@root> shutdown -f
```

ARGUMENTS

Table 10.9, “[fabric:container-create-child Arguments](#)” describes the command's arguments.

Table 10.9. fabric:container-create-child Arguments

Argument	Interpretation
--help	Displays the online help for this command
--ensemble-server	Whether the new container should be a Fabric Server.
--profile	A profile ID to associate with the new container. To associate multiple profiles with the container, specify this flag multiple times on the command line—for example, --profile foo --profile bar . If no profile is specified, the container is associated with the default profile.
--version	Specifies the version of the new container (the version must be created in advance using fabric:version-create). Defaults to the current default version (use version-list to find the current default).
--jvm-opts	Specify options to pass to the container's JVM.
--resolver	Specifies how the container will report its address to other containers. Valid values are localip , localhostname , publicip , publichostname , manualip . For more information see fabric:container-resolver-set .
Parent	<i>(Required)</i> The parent container ID.
Name	<i>(Required)</i> The name of the container to create. When creating multiple containers, it serves as a prefix
Number	The number of containers that should be created.

RELATED TOPICS

For more details about resolver policies, see:

[fabric:container-resolver-list](#)

[fabric:container-resolver-set](#)

fabric:create

NAME

fabric:container-create-cloud — creates one or more new containers on the cloud

SYNOPSIS

```

fabric:container-create-cloud [ --help ] [ --name contextName ] [ --provider cloudProvider ]
[ --api cloudAPI ] [ --identity cloudIdentity ] [ --credential loginCredential ] [ --imageId imageID ]
[ --os-family osFamily ] [ --os-version osVersion ] [ --hardwareId hardwareID ] [ --instanceType instanceType ]
[ --locationId location ] [ --user userAcc ] [ --password userPass ] [ --public-key-file file ]
[ --owner owner ] [ --group group ] [ --proxy-uri URI ] [ --ensemble-server ] [ --new-user jaasUser ]
[ --new-user-password jaasUserPass ] [ --new-user-role jaasUserRole ] [ --zookeeper-password zooPass ]
[ --resolver policy ] [ --min-port minPort ] [ --max-port maxPort ] [ --profile profileID ]
[ --version version ] [ --jvm-opts jvmOpts ] [ --add-option key=value ] [ --no-admin-access ]
{ Name } [ Number ]

```

DESCRIPTION

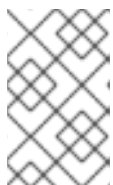
To access this command, you must have installed the **fabric-jclouds** feature. To install the **fabric-jclouds** feature, enter the following console command:

```
features:install fabric-jclouds
```

The **fabric:container-create-cloud** command provisions the container as follows:

1. Creates a new node on the cloud provider. The node is created using a JClouds compute service: either by lookup in the service registry (using the provider ID as a property) or by instantiating a new node, by specifying the identity and credential of the provider.
2. Connects to the created node, using the authentication metadata returned upon the node creation (this is usually a username and private key, where the username can be overridden by the **--user** option). After it connects to the node, it executes a script, which downloads the fabric distribution from the Maven proxy and untars the distribution.

By default, the script uses the oldest Maven proxy server in the current ensemble (every ensemble server has a Maven proxy server deployed in it). You can optionally override the default Maven proxy by specifying the **--proxy-uri** option. The script would then use the specified Maven proxy server to download the container runtime.



NOTE

The ability to override the Maven proxy is important in certain cases (for example, in a cloud deployment) where the remote host might not be able to access the default Maven proxy server.

3. Starts up the newly installed container (or containers) and installs the specified fabric profile (or profiles).
4. When creating multiple containers using this command (by adding the **Number** argument), multiple nodes will be created and a root container will be installed on each node.

By default, the newly created cloud containers belong to the current fabric (that is, the same fabric as the container from which you invoked the command). It is possible, however, to create a container on the compute instance that acts as the seed for a completely new fabric, separate from the current one. To create a new fabric on the compute instance, invoke the **fabric:container-create-cloud** command with the **--ensemble-server** flag, which makes the newly created container (or containers) an ensemble server, with its own fabric registry agent. The newly created ensemble server on the cloud *does not join the current ensemble*: it belongs to an independent ensemble (a new fabric).

ARGUMENTS

Table 10.10, “**fabric:container-create-cloud Arguments**” describes the command’s arguments.

Table 10.10. fabric:container-create-cloud Arguments

Argument	Interpretation
--help	Displays the online help for this command
--name	<i>(Required)</i> JClouds service context name.
--provider	JClouds provider name.
--api	The cloud API name.
--identity	The identity used to access the cloud service.
--credential	The credential used to access the cloud service.
--imageId	The image ID to use for the new node(s). Alternatively, the image can be specified indirectly using the --os-family and --os-version options. Defaults to an instance of the latest version of Ubuntu.
--os-family	Specify the image by requesting a particular kind of operating system—for example, ubuntu or redhat . To see which O/S families are available, type Tab while entering this option. Defaults to ubuntu .
--os-version	Specifies the version of the O/S family. The version number need not be exact (it will be rounded up to the latest available patch version). Defaults to the latest version available.
--hardwareId	Kind of hardware to use.
--instanceType	Type of instance required.
--locationId	The location used to create the new node(s).

Argument	Interpretation
--user	Specifies the O/S user account to run on the new nodes. If the user account does not already exist on the new nodes, it will automatically be created. Defaults to the username that matches the current user.
--password	Specifies the password associated with the O/S user account defined by the --user option.
--public-key-file	An option to specify a public key file to copy to the created node. Copying a public key file to a node can be used for SSH access using public key authentication. If no key file is specified, Fabric attempts to auto-detect the user's public key and, if found, this key will be used by default.
--owner	Optional owner of images; only really used for EC2, and will be deprecated in future.
--group	Group tag to use on the new node(s). Defaults to fabric .
--proxy-uri	URL of the Maven proxy server used to download the container runtime.
--ensemble-server	Whether the new container should be a Fabric Server (effectively creates a new fabric).
--new-user	Used in combination with the --ensemble-server option to ensure that at least one user exists in the JAAS realm of the Zookeeper login module for the new fabric (otherwise it would be impossible to connect to the newly created Fabric Server). When using this option, you <i>must</i> also specify a password using the --new-user-password option.
--new-user-password	Used in combination with the --new-user option and the --ensemble-server option to specify the new user's password. No default value.
--new-user-role	Used in combination with the --new-user option and the --ensemble-server to specify the new user's role. Default is admin .

Argument	Interpretation
--zookeeper-password	<p>Used in combination with the --ensemble-server option. Specifies the Zookeeper password, which is used to access the Zookeeper nodes under the /fabric/ path. Defaults to the password of the current session user.</p> <p>If you subsequently try to join the current container to the newly-created Fabric Server (ensemble server) using the fabric:join command, you will be prompted to enter the Zookeeper password.</p>
--resolver	<p>Specifies how the container will report its address to other containers. Valid values are localip, localhostname, publicip, publichostname, manualip. For more information see fabric:container-resolver-set.</p>
--min-port	<p>Specifies the minimum port number of the allowed IP port range. Default is 0.</p>
--max-port	<p>Specifies the maximum port number of the allowed IP port range. Default is 65535.</p>
--profile	<p>A list of profile IDs to associate with the new container.</p>
--version	<p>Specifies the version of the new container (the version must be created in advance using fabric:version-create). Defaults to the current default version (use version-list to find the current default).</p>
--jvm-opts	<p>Specify options to pass to the container's JVM.</p>
--add-option	<p>Specifies generic JCloud properties or provider-specific properties. For example, when using Amazon with Amazon VPC to create a container inside a VPN, you can specify --option subnetId=yoursubnetId to define the VPC subnet where you want the node to be created. If you want to specify more than one option, specify this option multiple times.</p>
--no-admin-access	<p>Disables admin access, as it might not be feasible on all images.</p>
Name	<p><i>(Required)</i> The name of the container to create. When creating multiple containers, it serves as a prefix.</p>

Argument	Interpretation
<i>Number</i>	The number of containers that should be created.

RELATED TOPICS

See the other Fabric cloud commands:

- [fabric:cloud-provider-add](#)
- [fabric:cloud-provider-list](#)
- [fabric:cloud-provider-remove](#)

For more details about resolver policies, see:

- [fabric:container-resolver-list](#)
- [fabric:container-resolver-set](#)
- [fabric:create](#)

NAME

`fabric:container-create-ssh` — creates one or more new containers through SSH

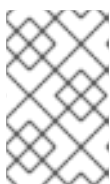
SYNOPSIS

```
fabric:container-create-ssh [ --help ] [ --host host ] [ --path path ] [ --user user ] [ --password password ] [ --private-key keyPath ] [ --port port ] [ --ssh-retries retries ] [ --proxy-uri URI ] [ --ensemble-server ] [ --profile profileID ] [ --version version ] [ --jvm-opts jvmOpts ] [ --resolver policy ] { Name } [ Number ]
```

DESCRIPTION

Specifically, this command provisions the container as follows:

1. Logs into the specified SSH host, using either the provided username and password *or* using the provided username and private key.
2. Runs a script on the remote host that that downloads the container runtime to the remote host. The runtime files are downloaded through a Maven proxy server. By default, the script uses the oldest Maven proxy server in the current ensemble (every Fabric Server has a Maven proxy server deployed in it). You can optionally override the default Maven proxy by specifying the `--proxy-uri` option. The script would then use the specified Maven proxy server to download the container runtime.



NOTE

The ability to override the Maven proxy is important in certain cases (for example, in a cloud deployment) where the remote host might not be able to access the default Maven proxy server.

- Starts up the newly installed container (or containers) and installs the specified fabric profile (or profiles).

By default, the newly created containers belong to the current fabric (that is, the same fabric as the container from which you invoked the command). It is possible, however, to create a container on the remote host that acts as the seed for a completely new fabric, separate from the current one. To create a new fabric on the remote host, invoke the **fabric:container-create-ssh** command with the **--ensemble-server** flag, which makes the newly created container (or containers) a Fuse Server. The newly created Fuse Server on the remote host *does not join the current ensemble*: it belongs to an independent ensemble (a new fabric).

ARGUMENTS

Table 10.11, “**fabric:container-create-ssh Arguments**” describes the command's arguments.

Table 10.11. fabric:container-create-ssh Arguments

Argument	Interpretation
--help	Displays the online help for this command
--host	<i>(Required)</i> Host name to SSH into.
--path	Path on the remote filesystem where the container is to be installed.
--user	<i>(Required)</i> User name for login.
--password	Password for login. If the password is omitted, private key authentication is used instead.
--private-key	Specifies the path to the private key on the local file system. The default is <code>~/.ssh/id_rsa</code> on *NIX platforms or <code>C:\Documents and Settings\UserName\.ssh\id_rsa</code> on Windows.
--port	The IP port number for the SSH connection.
--ssh-retries	Maximum number or times to retry SSH connection.
--proxy-uri	URL of the Maven proxy server used to download the container runtime.
--ensemble-server	Whether the new container should be a Fabric Server.
--profile	A list of profile IDs to associate with the new container.

Argument	Interpretation
--version	Specifies the version of the new container (the version must be created in advance using fabric:version-create). Defaults to the current default version (use version-list to find the current default).
--jvm-opts	Specify options to pass to the container's JVM.
--resolver	Specifies how the container will report its address to other containers. Valid values are localip , localhostname , publicip , publichostname , manualip . For more information see fabric:container-resolver-set .
Name	<i>(Required)</i> The name of the container to create. When creating multiple containers, it serves as a prefix.
Number	The number of containers that should be created.

RELATED TOPICS

For more details about resolver policies, see:

fabric:container-resolver-list
fabric:container-resolver-set
fabric:create

NAME

`fabric:container-delete`, `container-delete` — stops and deletes a Fuse Container

SYNOPSIS

```
fabric:container-delete [ --help ] [ [-r] | [ --recursive ] ] { Name }
```

DESCRIPTION

Deleting a Fuse Container deletes all of the files associated with the container from the host.

If the container has children, the default behavior of the command is to leave the children in place. You can force the deletion of the children using the **-r** option.



NOTE

If the container to be deleted is a Fabric Server, you must first remove it from the ensemble using **fabric:ensemble-remove**.

ARGUMENTS

Table 10.12, “[fabric:container-delete Arguments](#)” describes the command's arguments.

Table 10.12. `fabric:container-delete` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>-r, --recursive</code>	Recursively stops and deletes all child containers.
<i>Name</i>	Specifies the name of the container to delete.

NAME

`fabric:container-domains`, `container-domains` — lists a container's JMX domains

SYNOPSIS

```
fabric:container-domains [ --help ] { Name }
```

ARGUMENTS

Table 10.13, “[fabric:container-domains Arguments](#)” describes the command's arguments.

Table 10.13. `fabric:container-domains` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<i>Name</i>	Specifies the name of the container.

NAME

`fabric:container-info`, `container-info` — displays information about the specified container

SYNOPSIS

```
fabric:container-info [ --help ] [ ContainerName ]
```

ARGUMENTS

Table 10.14, “[fabric:container-info Arguments](#)” describes the command's arguments.

Table 10.14. `fabric:container-info` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

Argument	Interpretation
<i>ContainerName</i>	Specifies the name of the container for which information is provided. If not specified, the container name defaults to root .

NAME

fabric:container-list, container-list — lists the containers in a fabric

SYNOPSIS

fabric:container-list [--help] [--version *Version*] [[-v] | [--verbose]] [[*ID*] | [*profile*]]

ARGUMENTS

Table 10.15, “[fabric:container-list Arguments](#)” describes the command's arguments.

Table 10.15. **fabric:container-list Arguments**

Argument	Interpretation
--help	Displays the online help for this command
--version	Specifies a profile version to use as filter.
-v, --verbose	Display verbose output.
<i>ID</i>	Specifies a container ID to use in filtering the output.
<i>profile</i>	Specifies a profile to use in filtering the output. When a profile is specified only the containers with the profile are listed.

NAME

fabric:container-remove-profile, container-remove-profile — removes the specified list of profiles from the container

SYNOPSIS

fabric:container-remove-profile [--help] { *Name* } { *Profiles* }

ARGUMENTS

[fabric:container-remove-profile](#), describes the command's arguments.

Table 10.16. **fabric:container-remove-profile Arguments**

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>Name</i>	Specifies the name of the container.
<i>Profiles</i>	Specifies the list of profiles to remove from the container.

NAME

`fabric:container-resolver-list` — show the resolver policies for the specified containers

SYNOPSIS

`fabric:container-resolver-list` [`--help`] [*containers*]

DESCRIPTION

For all containers in the fabric, list the resolver policy and the following variants of the host address: local IP address, local hostname, public IP address, public hostname, and manually specified IP address. The host addresses are found by looking them up in the Fabric Registry for each container. This information is stored in the Fabric Registry at the time when the container is created. In most cases, only the local IP address and the local hostname are known. The public IP address and public hostname are generally available only for cloud containers.

ARGUMENTS

[Table 10.17, “`fabric:container-resolver-list` Arguments”](#) describes the command's arguments.

Table 10.17. `fabric:container-resolver-list` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>containers</i>	List of containers for which the resolver policy is displayed. Shows all containers by default.

RELATED TOPICS

[fabric:container-resolver-set](#)

NAME

`fabric:container-resolver-set` — specifies how the container reports its address to other containers

SYNOPSIS

`fabric:container-resolver-set` [`--help`] [`--container` *name*] [`--all`] { *Resolver* }

DESCRIPTION

Apply the specified resolver policy to the specified container or containers, where the resolver policy can take one of the following values:

```
localip
localhostname
publicip
publichostname
manualip
```

The **localip** and **localhostname** resolver policies are suitable for accessing a container in a LAN. The **publicip** and **publichostname** resolver policies are suitable for accessing a container in a WAN (Internet), but they are typically only available for cloud containers. In the case of a the cloud, **localip** and **localhostname** can be used for container-to-container connections within the cloud, but for container-to-container connections from outside the cloud, you must use **publicip** or **publichostname**.

Fabric manages host addresses as follows:

- When you create a new container, fabric tries to discover as much as it can about the container's host address and stores this information in the following fields in the fabric registry: **localip** (local IP address); **localhostname** (local hostname); **publicip** (public IP address); **publichostname** (public hostname).

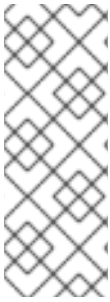
For example, if you create a new container using the **fabric:container-create-ssh** command and specify the local IP address to the **--host** option, fabric attempts to perform a reverse lookup to obtain the corresponding local hostname and then stores both the local IP address *and* the local hostname in the Fabric Registry.

If you create a new container in the cloud, the metadata sent by the cloud provider typically includes a complete set of host addresses: **localip**, **localhostname**, **publicip**, and **publichostname**.

- Every container in the fabric has its own *resolver policy*, which determines what kind of host address is returned to another container that wants to connect to it. The container's resolver policy is set in one of the following ways:
 - (*Default*) By inheriting the resolver policy from the global resolver policy (specified at the time the fabric is created)
 - By specifying the resolver policy explicitly at the time the container is created (through the **--resolver** option).
 - By invoking the **fabric:container-resolver-set** command.
- The container's resolver policy is applied whenever fabric looks up the container's host address, irrespective of what protocol is involved. In particular, the resolver policy determines the form of the host address used in the following URLs:
 - Fabric Ensemble URL,
 - SSH URL (console client port),
 - Maven proxy URL,
 - JMX URL.

For example, if your fabric includes a container called **SSH1** (originally created using the

`fabric:container-create-ssh` command) and the `SSH1` container is configured with the `localip` resolver policy, any container that tries to connect to `SSH1` will automatically receive the local IP address of `SSH1` when it looks up the Fabric Registry.



NOTE

A container's resolver policy only affects the host address returned when *other* containers want to connect to it. The container's own policy has no effect on how the container resolves the host addresses of the other containers. In other words, if containers `X`, `Y`, and `Z` want to connect to container `SSH1`, the form of host address they get is determined by `SSH1`'s resolver policy. But if `SSH1` wants to connect to container `X`, it is container `X`'s resolver policy that is used.

MANUAL IP RESOLVER POLICY

The `manualip` resolver policy is a special case. If none of the standard resolver policies are suitable for your network set-up, you can manually specify a container's host address by setting the following key in the Fabric Registry:

```
/fabric/registry/containers/config/ContainerName/manualip
```

ARGUMENTS

Table 10.18, “`fabric:container-resolver-set` Arguments” describes the command's arguments.

Table 10.18. `fabric:container-resolver-set` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>--container</code>	Apply the resolver policy to the specified container.
<code>--all</code>	Apply the resolver policy to all containers in the fabric.
<i>Resolver</i>	<i>(Required)</i> The resolver policy to set on the specified container(s). Possible values are: <code>localip</code> , <code>localhostname</code> , <code>publicip</code> , <code>publichostname</code> , <code>manualip</code> .

NAME

`fabric:container-rollback` — roll back the specified containers to an older version

SYNOPSIS

```
fabric:container-rollback [ --help ] [ --all ] { Version } [ ContainerList ]
```

DESCRIPTION

For an example of how this command is used, see [fabric:container-upgrade](#).

ARGUMENTS

Table 10.19, “[fabric:container-rollback Arguments](#)” describes the command's arguments.

Table 10.19. fabric:container-rollback Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>--all</code>	Roll back all containers.
<i>Version</i>	<i>(Required)</i> The version to roll back to.
<i>ContainerList</i>	The list of containers to roll back. An empty list implies the current container.

NAME

fabric:container-start, container-start — start the specified container

SYNOPSIS

```
fabric:container-start [ --help ] { name }
```

ARGUMENTS

Table 10.20, “[fabric:container-start Arguments](#)” describes the command's arguments.

Table 10.20. fabric:container-start Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>Name</i>	Specifies the name of the container.

NAME

fabric:container-stop, container-stop — shuts down the specified container

SYNOPSIS

```
fabric:container-stop [ --help ] { Name }
```

ARGUMENTS

Table 10.21, “[fabric:container-stop Arguments](#)” describes the command's arguments.

Table 10.21. fabric:container-stop Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>Name</i>	<i>(Required)</i> The name of the container.

NAME

`fabric:container-upgrade` — upgrade the specified containers to a new version

SYNOPSIS

```
fabric:container-upgrade [ --help ] [ --all ] { Version } [ ContainerList ]
```

DESCRIPTION

This command is typically used in combination with the `fabric:profile-edit` command to guarantee atomicity of profile modifications. That is, if multiple edits need to be made to a profile, you can use `fabric:container-upgrade` to roll out all of the changes in one step.

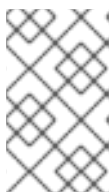
For example, consider the container, `child1`, which is currently assigned to version 1.0 and has the `sample` profile deployed inside it. If you need to make multiple changes to the `sample` profile, you can roll out these changes atomically, as follows:

1. Create a new version, 1.1, to hold the pending changes, as follows:

```
karaf@root> fabric:version-create
Created version: 1.1 as copy of: 1.0
```

2. Now start editing the new version of the sample profile, remembering to specify `1.1`, so that the modifications are applied to version 1.1 of `sample`. For example, to add the `camel-quartz` feature to the sample profile, enter the following command:

```
fabric:profile-edit --features camel-quartz sample 1.1
```



NOTE

Instead of adding the option `1.1` to every edit command, you could change the default version to 1.1 by entering the command, `fabric:version-set-default 1.1`.

3. When you have finished editing the `sample` profile and you are ready to let the changes take effect on the container, `child1`, you can roll out the changes by upgrading the `child1` container to version 1.1, as follows:

```
fabric:container-upgrade 1.1 child1
```

4. If you are not happy with the changes you made, you can easily roll back to the old version of the `sample` profile, using the `fabric:container-rollback` command, as follows:

```
fabric:container-rollback 1.0 child1
```


ARGUMENTS

Table 10.22, “**fabric:container-upgrade Arguments**” describes the command's arguments.

Table 10.22. fabric:container-upgrade Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>--all</code>	Upgrade all containers.
<i>Version</i>	(Required) The version to upgrade to.
<i>ContainerList</i>	The list of containers to upgrade. An empty list implies the current container.

NAME

`fabric:create` — creates a new fabric and imports fabric profiles

SYNOPSIS

```
fabric:create [ --help ] [ --clean ] [ --no-import ] [ --import-dir dir ] [[ -v ] | [ --verbose ]] [[ -t ] | [ --time ] millis] [[ -n ] | [ --non-managed ]] [[ -p ] | [ --profile ] profile] [ --new-user username ] [ --new-user-password password ] [ --new-user-role role ] [ --zookeeper-password zooPassword ] [ --generate-zookeeper-password ] [[ -g ] | [ --global-resolver ] policy] [[ -r ] | [ --resolver ] policy] [[ -m ] | [ --manual-ip ] ipAddress] [ --min-port port ] [ --max-port port ] [ ContainerList ]
```

DESCRIPTION

This command is used to create a new fabric. It can also be used to change the Fabric Servers in an existing fabric. Converting the current container into a fabric has two important side effects:

- The contents of a container should now be managed using *fabric profiles*. Do not try to deploy bundles and features directly in a fabric container.
- The default JAAS realm is superseded by the Zookeeper login module, which stores user data in the Zookeeper registry. As the fabric is created it initializes the user data by importing all of the user data that it finds in the `etc/users.properties` file. If the `users.properties` file is empty, you can specify a new user explicitly using the `--new-user` and `--new-user-password` options (at least one user *must* be defined).

If you want to create your own import directory with custom profile data, it is recommended that you proceed as follows:

1. Create a fabric that imports the sample profiles (for example, using `fabric:create`).
2. Modify the sample profiles using the `fabric:profile-create`, `fabric:profile-delete`, and `fabric:edit` commands.
3. Export the modified profiles using the `fabric:export` command.

ARGUMENTS

Table 10.23, “**fabric:create Arguments**” describes the command's arguments.

Table 10.23. fabric:create Arguments

Argument	Interpretation
--help	Displays the online help for this command
--clean	Clean local zookeeper cluster and configurations.
--no-import	Disable the import of the sample registry data.
--import-dir	Directory of files to import into the newly created ensemble.
-v, --verbose	Flag to enable verbose output of files being imported.
-t, --time	How long to wait (milliseconds) for the ensemble to start up, before trying to import the default data.
-n, --non-managed	Specifies that the container remains unmanaged.
-p, --profile	Specifies the profile to use for the ensemble containers in the new fabric.
--new-user	<p>Create a new user in the new fabric's JAAS realm. Because the fabric:create command automatically imports user data from the etc/users.properties file, you would only need to specify this option, if the etc/users.properties file contains no valid user entries.</p> <p>When using this option, you <i>must</i> also specify a password using the --new-user-password option.</p>
--new-user-password	Used in combination with the --new-user option to specify the new user's password. No default value.
--new-user-role	Used in combination with the --new-user option to specify the new user's role. Default is admin .
--zookeeper-password	<p>Specifies the Zookeeper password, which is used to access the Zookeeper nodes under the /fabric/ path. Defaults to the password of the current session user.</p> <p>Subsequently, because the Zookeeper password is cached in the current session, you normally do not need to provide it when executing fabric commands. You can display the Zookeeper password at any time using the fabric:ensemble-password command.</p>

Argument	Interpretation
--generate-zookeeper-password	Directs Fabric to generate a random Zookeeper password. Subsequently, you can display the Zookeeper password using the fabric:ensemble-password command.
-g, --global-resolver	Specifies the global resolver policy, which becomes the default resolver policy applied to all new containers created in this fabric. Possible values are: localip, localhostname, publicip, publichostname, manualip . The default is localhostname .
-r, --resolver	Specifies the local resolver policy. Possible values are: localip, localhostname, publicip, publichostname, manualip . The default is localhostname .
-m, --manual-ip	If you select the manualip resolver policy (using either the --resolver or --global-resolver options), specifies the IP address to use for the resolver.
--min-port	Specifies the minimum port number of the allowed IP port range. Default is 0 .
--max-port	Specifies the maximum port number of the allowed IP port range. Default is 65535 .
ContainerList	The list of containers to include in the ensemble. An empty list implies the current container.

EXAMPLES

Create a fabric and import sample profiles from the **ESBInstallDir/fabric/import** directory, as follows:

```
fabric:create --clean
```

Create a fabric *without* imported profiles, as follows:

```
fabric:create --clean --no-import
```

Create a fabric and import profiles from the custom import directory, *CustomImportDir*, as follows:

```
fabric:create --clean --import-dir CustomImportDir
```

Re-create a fabric such that the containers, **reg1**, **reg2**, and **reg3**, are now included in the registry ensemble (an ensemble must consist of an odd number of containers):

```
fabric:create reg1 reg2 reg3
```

In this case, the contents of the Zookeeper registry are preserved and the ensemble is expanded to include the specified containers.

RELATED TOPICS

For more details about resolver policies, see:

- [fabric:container-resolver-list](#).
- [fabric:container-resolver-set](#).

NAME

`fabric:ensemble-add` — extend the current Fabric Ensemble by converting the specified containers into Fuse Servers

SYNOPSIS

```
fabric:ensemble-add [ --help ] { ContainerList }
```

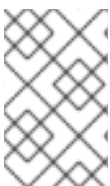
DESCRIPTION

Because the total number of containers in an ensemble must always be odd, you should add an even number of containers.

For example, consider a fabric consisting of three containers—**root1**, **root2**, and **root3**—where **root1** is an Fuse Server and **root2** and **root3** are ordinary Fabric Containers. You can now add **root2** and **root3** to the current ensemble by entering the following console command:

```
fabric:ensemble-add root2 root3
```

Normally, it makes sense to have at most one Fabric Server running on each host, so that the specified containers are actually running on remote hosts (hence, it usually does not make sense to add child containers to an ensemble). You do not need to provide any information about where the containers are running, however, because fabric already knows the location of the containers in the fabric.



NOTE

Because the Fabric Ensemble is the key component of Fuse Fabric, changing the ensemble is a critical operation. All data will be preserved and copied to the new Fuse Servers before switching.

ARGUMENTS

Table 10.24, “[fabric:ensemble-add Arguments](#)” describes the command's arguments.

Table 10.24. `fabric:ensemble-add` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>ContainerList</i>	The list of containers to add.

NAME

`fabric:ensemble-list` — lists the Fuse Servers in the current Fabric Ensemble

SYNOPSIS

```
fabric:ensemble-list [ --help ]
```

DESCRIPTION

For a complete listing of *all* the containers in the fabric, use `fabric:container-list` instead.

ARGUMENTS

Table 10.25, “`fabric:ensemble-list` Arguments” describes the command's arguments.

Table 10.25. `fabric:ensemble-list` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

NAME

`fabric:ensemble-password` — display the ensemble password

SYNOPSIS

```
fabric:ensemble-password [ --help ]
```

DESCRIPTION

The ensemble password protects access to the Zookeeper nodes under the `/fabric/` path, which contains critical configuration data for the fabric. To ensure integrity of the fabric configuration data, you should modify the fabric configuration exclusively using the `fabric:*` console commands.

ARGUMENTS

Table 10.26, “`fabric:ensemble-password` Arguments” describes the command's arguments.

Table 10.26. `fabric:ensemble-password` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

NAME

`fabric:ensemble-remove` — remove the specified containers from the current ensemble

SYNOPSIS

```
fabric:ensemble-remove [ --help ] { ContainerList }
```

DESCRIPTION

Re-create the current ensemble, excluding the specified containers from the ensemble. All containers are switched to this new ensemble.

**NOTE**

Because the Fabric Ensemble is the key component of Fuse Fabric, changing the ensemble is a critical operation. All data will be preserved and copied to the new ensemble before switching.

ARGUMENTS

Table 10.27, “**fabric:ensemble-remove Arguments**” describes the command's arguments.

Table 10.27. fabric:ensemble-remove Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>ContainerList</i>	The list of containers to remove. Must be an even number of containers.

NAME

`fabric:export` — export the contents of the Fabric Registry to the specified directory in the filesystem

SYNOPSIS

```
fabric:export [ --help ] [ -d|--delete ] [ -p|--path path ] [ -f|--regex regex ] [ -rf|--reverse-regex regex ] [ -t|--trim ] [ --dry-run ] { target }
```

DESCRIPTION

The output of this command is compatible with the import options of the other **fabric** commands. The regular expression options, **-f** and **-rf**, provide you with considerable flexibility at specifying which parts of the Fabric Registry to export. For example, to export every version of the **default** profile's data, you could use a command like the following:

```
fabric:export -f /fabric/configs/versions/[0-9\\.]*/profiles/default/.*
```

Where a double-backslash, `\\`, is required to escape the period, `.`, so that the period gets interpreted as a character literal.

ARGUMENTS

Table 10.28, “**fabric:export Arguments**” describes the commands arguments.

Table 10.28. fabric:export Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-d, --delete</code>	Delete the existing contents of the target directory before exporting. Caution: Performs a recursive delete!

Argument	Interpretation
-p, --path	Top-level znode to export. Default is <code>/</code> .
-f, --regex	Specifies a regular expression that matches the znode paths you want to <i>include</i> in the export. For multiple include expressions, specify this option multiple times. The regular expression syntax is defined by the <code>java.util.regex</code> package.
-rf, --reverse-regex	Specifies a regular expression that matches the znode paths you want to <i>exclude</i> from the export. For multiple exclude expressions, specify this option multiple times. The regular expression syntax is defined by the <code>java.util.regex</code> package.
-t, --trim	Trims the first timestamp comment line in properties files starting with the <code>#</code> character.
--dry-run	Log the actions that would be performed during an export, but do not actually perform the export.
target	Path of the directory to export to. Default is <code>./export</code> .

NAME

`fabric:import` — import data either from a filesystem or from a properties file into the Fabric Registry

SYNOPSIS

```
fabric:import [ --help ] [ -fs|--filesystem ] [ -props|--properties URL ] [ -t|--target path ] [ -d|--delete ] [ -f|--regex regex ] [ -rf|--reverse-regex regex ] [ -v|--verbose ] [ --dry-run ] { source }
```

ARGUMENTS

Table 10.29, “`fab:start` Arguments” describes the commands arguments.

Table 10.29. `fab:start` Arguments

Argument	Interpretation
--help	Displays the online help for this command
-fs, --filesystem	Indicates that the <i>source</i> argument is a directory on the filesystem. Defaults to true .
-props, --properties	Indicates that the <i>source</i> argument is a properties file. Defaults to false .

Argument	Interpretation
-t, --target	Path of the znode that the data is imported into. Default is <code>/</code> .
-d, --delete	Delete any paths that are not in the tree being imported. Ignored when importing a properties file. Caution: Using this option could permanently delete all or part of the Fabric Registry.
-f, --regex	Specifies a regular expression that matches the znode paths you want to <i>include</i> in the import. For multiple include expressions, specify this option multiple times. The regular expression syntax is defined by the <code>java.util.regex</code> package.
-rf, --reverse-regex	Specifies a regular expression that matches the znode paths you want to <i>exclude</i> from the import. For multiple exclude expressions, specify this option multiple times. The regular expression syntax is defined by the <code>java.util.regex</code> package.
-v, --verbose	Verbose log of files being imported.
--dry-run	Log the actions that would be performed during an import, but do not actually perform the import.
source	Location of a filesystem (if <code>--filesystem</code> is specified) or a properties file (if <code>--properties</code> is specified). Defaults to <code>./import</code> .

NAME

`fabric:join` — join a container to an existing fabric

SYNOPSIS

```
fabric:join [ --help ] [[ -f ] | [ --force ] ] [[ -p ] | [ --profile Profile ] ] [[ -n ] | [ --non-managed ] ] [ --zookeeper-password zooPassword ] [[ -r ] | [ --resolver policy ] ] [[ -m ] | [ --manual-ip ipAddress ] ] [ --min-port port ] [ --max-port port ] URL [ ContainerName ]
```

DESCRIPTION

The `fabric:join` command can be used in either of the following scenarios:

- You have an existing fabric, A, and you want to join a standalone container to fabric A.
- You have two separate fabrics, A and B, and you want to transfer a container from fabric B to fabric A.

ARGUMENTS

[Table 10.30, “`fabric:join` Arguments”](#) describes the command's arguments.

Table 10.30. fabric:join Arguments

Argument	Interpretation
--help	Displays the online help for this command.
-f, --force	Forces the provided container name to be used.
-p, --profile	Specifies the profile to associate with the container after it joins the fabric. The fabric profile, which installs the Fabric Agent, is automatically assigned to all managed containers.
-n, --non-managed	Registers the container with the fabric's ensemble, but does not install a Fabric Agent into the container. The container's configuration is not managed by the fabric and continues to behave like a standalone container except that it can be discovered through the fabric's ensemble.
--zookeeper-password	The ensemble password for the fabric that you are trying to join. If you do not specify this option, you will be prompted to enter the password.
-r, --resolver	Specifies the local resolver policy. Possible values are: localip , localhostname , publicip , publichostname , manualip . The default is localhostname .
-m, --manual-ip	If you select the manualip resolver policy (using the --resolver option), specifies the IP address to use for the resolver.
--min-port	Specifies the minimum port number of the allowed IP port range. Default is 0 .
--max-port	Specifies the maximum port number of the allowed IP port range. Default is 65535 .
URL	Specifies the URL of one of the Fabric Servers, specified in the format Host[:Port] . The Port value defaults to 2181 .
ContainerName	Specifies a unique name for the container to use when joining the fabric. By default, the value of the karaf.name property from the etc/system.properties file is used.

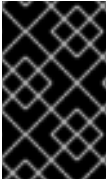
EXAMPLES

The following command will add a standalone container to a fabric as a managed container:

■

```
fabric:join myhostA ishmael
```

Where **myhostA** is the hostname of a Fabric Server (you must connect to a Fabric Server, not an ordinary fabric container) and the container is assigned the name **ishmael**. You will be prompted to enter the fabric's Zookeeper password.



IMPORTANT

If the container being added to a fabric is assigned the same name as a container that is already a part of the fabric, the original container will be reset to have the same settings as the new container.



WARNING

If no container name is specified as part of the command, the command will use the value of the **karaf.name** property from the **etc/system.properties** file. The default setting for this property is **root**. To avoid conflicts, you should either specify a container name or change the value of the **karaf.name** property.

To make sure that the container starts up with a specific profile, you use the **-p** argument as follows:

```
fabric:join -p whaler myhostA ishmael
```

The container **ishmael** is assigned the profile, **whaler**, when it joins the fabric.

If you want to be able to configure the container manually, but take advantage of the fabric's discovery features, you can add the container as a non-managed container using the following command:

```
fabric:join -n myhostA ishmael
```

NAME

fabric:mq-create — create a new broker profile

SYNOPSIS

```
fabric:mq-create [ --help ] [ --group groupName ] [ --networks brokerGroup, ... ] [ --networks-username user ] [ --networks-password password ] [ --create-container containerID, ... ] [ --assign-container containerID, ... ] [ --config configFile ] [ --data dataDir ] [ --jmx-user jmxUser ] [ --jmx-password jmxPassword ] [ --jvm-opts jvmOpts ] [ --version version ] { name }
```

ARGUMENTS

[Table 10.31, “fabric:mq-create Arguments”](#) describes the command's arguments.

Table 10.31. fabric:mq-create Arguments

Argument	Description
--help	Displays the online help for this command.

Argument	Description
<code>--group <i>groupName</i></code>	Specifies the name of the group to which brokers using this profile are assigned. By default brokers are assigned to the default group.
<code>--networks <i>brokerGroup,...</i></code>	Specifies a comma separated list of broker groups to which brokers using this profile will establish network connections to form a network of brokers.
<code>--networks-username</code>	Specifies the username part of the credentials that are used to connect to the broker networks specified by the <code>--networks</code> option.
<code>--networks-password</code>	Specifies the password part of the credentials that are used to connect to the broker networks specified by the <code>--networks</code> option.
<code>--create-container <i>containerID,...</i></code>	Specifies a comma separated list of child containers to create using the new profile. The new containers will be children of the container from which the command is executed.
<code>--assign-container <i>containerID,...</i></code>	Specifies a comma separated list of containers to which the new profile will be deployed.
<code>--config <i>configFile</i></code>	Specifies the ensemble path of the XML configuration template used by the profile. The path will have the syntax <code>/fabric/configs/versions/version/profiles/profile/config.xml</code> .
<code>--data <i>dataDir</i></code>	Specifies the path, relative to the container, for storing the persistence data for a broker using the profile.
<code>--jmx-user</code>	The JMX username for logging on to the parent's JMX port.
<code>--jmx-password</code>	The JMX password for logging on to the parent's JMX port.
<code>--jvm-opts</code>	Specify options to pass to the container's JVM.
<code>--version <i>version</i></code>	Specifies the version into which the profile is stored. Defaults to the current default version.
<i>name</i>	Specifies the name of the new broker profile.

EXAMPLES

To create a new broker profile with the name **myBrokerProfile** that uses the XML template file **myConfigTemplate.xml** use the command:

```
fabric:mq-create --config /fabric/configs/versions/1.0/profiles/mq-
base/myConfigTemplate.xml myBrokerProfile
```

To create a new broker profile and create a new container using the new profile use the command:

```
fabric:mq-create --config /fabric/configs/versions/1.0/profiles/mq-
base/myConfigTemplate.xml --create-container broker1 myBrokerProfile
```

To create a new broker profile and associate it with an existing container use the command:

```
fabric:mq-create --config /fabric/configs/versions/1.0/profiles/mq-
base/myConfigTemplate.xml --assign-container container1 myBrokerProfile
```

NAME

`fabric:profile-change-parents` — replace the profile's parents with the specified list of parents (where the parents are specified as a space-separated list)

SYNOPSIS

```
fabric:profile-change-parents [ --help ] [ --version version ] { Name } { ParentList }
```

ARGUMENTS

Table 10.32, “[fabric:profile-change-parents Arguments](#)” describes the command's arguments.

Table 10.32. `fabric:profile-change-parents` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>--version</code>	The profile version. Defaults to the current default version (use <code>version-list</code> to find the current default).
<i>Name</i>	(Required) Name of the profile.
<i>ParentList</i>	(Required) The list of new parent profiles.

NAME

`fabric:profile-create` — create a new profile with the specified name and version

SYNOPSIS

```
fabric:profile-create [ --help ] [ --version version ] [ --parents parentList ] { Name }
```

DESCRIPTION

The new profile is created *only for the version you specify* (or the current default version). If you want to create a profile for every version, you must invoke **fabric:profile-create** separately for each version (use **fabric:version-list** to list all versions).

The newly created profile is initially empty, apart from the settings inherited from the parent profiles. To add settings to the new profile, use the **fabric:profile-edit** command.

For example, to add the new profile, **test**, which has the current default version and inherits from the parent profiles, **mq** and **camel**, enter the following console command:

```
fabric:profile-create --parents mq --parents camel test
```

ARGUMENTS

Table 10.33, “**fabric:profile-create Arguments**” describes the command's arguments.

Table 10.33. fabric:profile-create Arguments

Argument	Interpretation
--help	Displays the online help for this command
--version	The profile version. Defaults to the current default version (use version-list to find the current default).
--parents	Optionally specifies one or multiple parent profiles. To specify multiple parent profiles, specify this flag multiple times on the command line—for example, --parents foo --parents bar .
<i>Name</i>	<i>(Required)</i> Name of the new profile.

NAME

fabric:profile-delete — delete the specified version of the specified profile (where the version defaults to the current default version)

SYNOPSIS

```
fabric:profile-delete [ --help ] [ --version version ] { Profile }
```

ARGUMENTS

Table 10.34, “**fabric:profile-delete Arguments**” describes the command's arguments.

Table 10.34. fabric:profile-delete Arguments

Argument	Interpretation
--help	Displays the online help for this command

Argument	Interpretation
--version	The profile version to delete. Defaults to the current default version (use version-list to find the current default).
Profile	<i>(Required)</i> Name of the profile to delete.

NAME

fabric:profile-display — displays information about the specified version of the specified profile (where the version defaults to the current default version)

SYNOPSIS

```
fabric:profile-display [ --help ] [ --version version ] [ -o ] [ --overlay ] {
    Profile
}
```

ARGUMENTS

[Table 10.35, “fabric:profile-display Arguments”](#) describes the command's arguments.

Table 10.35. fabric:profile-display Arguments

Argument	Interpretation
--help	Displays the online help for this command
--version	Select a specific profile version. Defaults to the current default version (use version-list to find the current default).
-o, --overlay	Enable overlay. Shows the effective profile settings, taking into account the settings inherited from parent profiles.
Profile	<i>(Required)</i> The name of the profile.

NAME

fabric:profile-edit – edits the specified version of the specified profile (where the version defaults to the current default version)

SYNOPSIS

```

fabric:profile-edit [
  --help
] [[
  -p
] | [
  --pid
]PID] [[
  -r
] | [
  --repositories
] | [
  -f
] | [
  --features
] | [
  -b
] | [
  --bundles
] | [
  --fabs
] | [
  -c
] | [
  --config
] | [
  -s
] | [
  --system
] | [
  -o
] | [
  --overrides
]] [[
  --set
] | [
  --delete
]] [[
  --append
] | [
  --remove
]] [
  --import-pid
] [[
  --delimiter
]delim] [[
  --resource
]ResourceName] {
  Profile
} [
  Version
]

```

DESCRIPTION

In the specified profile, you can edit different kinds of settings, as follows:

- *Feature repository locations*—to add a feature repository to the profile, enter a command in the following format:

```
fabric:profile-edit --repositories RepoList Profile [Version]
```

For example, to add the fuse-fabric feature repository to the profile, enter a command like the following:

```
fabric:profile-edit --repositories mvn:org.fusesource.fabric/fuse-fabric/6.0.0.redhat-024/xml/features Profile [Version]
```

To delete repositories, enter a command of the following form:

```
fabric:profile-edit --delete --repositories RepoList Profile [Version]
```

To edit repository locations directly, using a visual text editor, enter the following command:

```
fabric:profile-edit Profile [Version]
```

The visual editor opens, showing the contents of the specified profile's agent properties. To edit the repository settings, add, modify, or delete lines of the following form:

```
repository.ID=URL
```

Where *ID* is an arbitrary unique identifier and *URL* gives the location of a single feature repository. Only one repository URL can be specified on each line.

- *Features to install*—to add features to the profile, enter a command in the following format:


```
fabric:profile-edit --features FeatureList Profile [Version]
```

Where *FeatureList* is a comma-separated list of features. For example, to add the camel-jetty and the camel-quartz features to the default version of the sample profile, enter a command like the following:

```
fabric:profile-edit --features camel-jetty,camel-quartz sample
```

To delete features, enter a command of the following form:

```
fabric:profile-edit --delete --features FeatureList Profile [Version]
```

To edit features directly, using the visual text editor, enter the following command:

```
fabric:profile-edit Profile [Version]
```

The visual editor opens, showing the contents of the specified profile's agent properties. To edit the features, add, modify, or delete lines of the following form:

```
feature.ID=FeatureName
```

Where *ID* is an arbitrary unique identifier and *FeatureName* is the name of a feature.

- *Bundles to install*—to add bundles to the profile, enter a command in the following format:

```
fabric:profile-edit --bundles BundleList Profile [Version]
```

For example, to add camel-quartz bundle to the sample profile, enter a command like the following:

-

```
fabric:profile-edit --bundles mvn:org.apache.camel/camel-  
quartz/2.10.0.redhat-60024 sample
```

To delete bundles, enter a command of the following form:

```
fabric:profile-edit --delete --bundles BundleList Profile [Version]
```

To edit bundles directly, using the visual text editor, enter the following command:

```
fabric:profile-edit Profile [Version]
```

The visual editor opens, showing the contents of the specified profile's agent properties. To edit the bundles, add, modify, or delete lines of the following form:

```
bundle.ID=URL
```

Where *ID* is an arbitrary unique identifier and *URL* specifies the bundle's location.

- *Configuration settings for the OSGi Config Admin service*—to modify or create a configuration setting from the OSGi Config Admin service, enter a command in the following format:

```
fabric:profile-edit --pid PID/Property=Value Profile [Version]
```

Where *PID* is a persistent ID, which is used in the context of the OSGi Config Admin service to identify a collection of related properties. For example, to change the value of the secure HTTPS port used by the Jetty server in the sample profile, you could edit the `org.osgi.service.http.port.secure` property from the `org.ops4j.pax.web` PID using a command like the following:

```
fabric:profile-edit --pid  
org.ops4j.pax.web/org.osgi.service.http.port.secure=8553 sample
```

To delete a property, enter a command of the following form:

```
fabric:profile-edit --delete --pid PID/Property Profile [Version]
```

To edit OSGi Config Admin settings directly, using the visual text editor, enter the following command:

```
fabric:profile-edit --pid PID Profile [Version]
```

The visual editor opens, showing the contents of the specified profile's *PID.properties* file (which is actually stored in the ZooKeeper registry). To edit the properties, add, modify, or delete lines of the following form:

```
Property=Value
```

-

Property settings from etc/config.properties—to modify or create a Java system property in the container's *etc/config.properties* file (which affects the container), enter a command in the following format:

```
fabric:profile-edit --config Property=Value Profile [Version]
```

For example, to change the value of the *karaf.startlevel.bundle* Java system property in *config.properties*, you would enter a command like the following:

```
fabric:profile-edit --config karaf.startlevel.bundle=80 Profile [Version]
```

To delete a Java system property from *config.properties*, enter a command of the following form:

```
fabric:profile-edit --delete --config Property Profile [Version]
```

To edit the Java system properties directly, using the visual text editor, enter the following command:

```
fabric:profile-edit Profile [Version]
```

The visual editor opens, showing the contents of the specified profile's agent properties. To edit the Java system properties (analogous to `etc/config.properties`), add, modify, or delete lines of the following form:

```
config.Property=Value
```

- *Property settings from etc/system.properties*—to modify or create a Java system property in the container's `etc/system.properties` file (which affects bundles deployed in the container), enter a command in the following format:

```
fabric:profile-edit --system Property=Value Profile [Version]
```

For example, to change the default port for the OSGi HTTP service, you would enter a command like the following:

```
fabric:profile-edit --system org.osgi.service.http.port=8181 Profile [Version]
```

If the system property, *Property*, is already set at the JVM level (for example, through the `--jvm-opts` option to the `fabric:container-create` command), the preceding `fabric:profile-edit` command *will not override the JVM level setting*. If you want to override the JVM level setting, you must indicate this explicitly by adding the `karaf.override` prefix to the property name, *Property*—for example:

```
fabric:profile-edit --system karaf.override.Property=Value Profile [Version]
```

To delete a Java system property from `system.properties`, enter a command of the following form:

```
fabric:profile-edit --delete --system Property Profile [Version]
```

To edit the Java system properties directly, using the visual text editor, enter the following command:

```
fabric:profile-edit Profile [Version]
```

The visual editor opens, showing the contents of the specified profile's agent properties. To edit the Java system properties (analogous to `etc/system.properties`), add, modify, or delete lines of the following form:

```
system.Property=Value
```

If you want to ensure that this setting overrides any JVM level setting, set the system property as follows:

```
system.karaf.override.Property=Value
```

IMPORTANT

Any modifications you make to a profile using `fabric:profile-edit` are *immediately* propagated to the containers that use that profile. This is not the recommended way to edit profiles, however: if you change multiple settings in the profile, you could potentially put the affected containers into an inconsistent state. To guarantee atomicity, it is better to use the `fabric:profile-edit` command in combination with the `fabric:container-upgrade` command—see [fabric:container-upgrade](#).

ENCLOSING AN OPTION VALUE IN QUOTES

The Karaf shell strips double quotes from an option by default. Hence, to enclose an option value in double quotes, it is necessary to enclose the whole setting in double quotes and to escape the quotes around the option value. For example, to define the system property setting, `http.nonProxyHosts="myserver1|myserver2"`, on the default profile, you would use the following command:

```
fabric:profile-edit --system
"karaf.override.http.nonProxyHosts=\"myserver1|myserver2\"" default
```

where the `karaf.override` prefix is prepended to the property name,

because `http.nonProxyHosts` is already set at the JVM level and needs to be overridden.

ARGUMENTS

Table 10.36, “[fabric:profile-edit Arguments](#)” describes the command's arguments.

Table 10.36. `fabric:profile-edit` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-p, --pid</code>	Edit an OSGi configuration property, specified in the format <i>PID/Property</i> .
<code>-r, --repositories</code>	Edit the list of feature repositories.
<code>-f, --features</code>	Edit features, specifying a comma-separated list of features to add (or delete).
<code>-b, --bundles</code>	Edit bundles, specifying a comma-separated list of bundles to add (or delete).
<code>-f, --fabs</code>	Edit Fuse Application Bundles (FABs), specifying a comma-separated list of FABs to add (or delete).
<code>-c, --config</code>	Edit the Java system properties that affect the container (analogous to editing <code>etc/config.properties</code> in a root container).
<code>-s, --system</code>	Edit the Java system properties that affect installed bundles (analogous to editing <code>etc/system.properties</code> in a root container).
<code>-o, --overrides</code>	Edit the bundle overrides, specifying a comma-separated list of bundles to add (or delete). A bundle override can be used to override the bundle version installed by a feature. For example, if a feature installs version 1.0.0 of a particular bundle, you could use a bundle override to install version 1.0.1 of the bundle instead.
<code>--set</code>	Set or create values (selected by default).
<code>--delete</code>	Delete values.

Argument	Interpretation
--append	When editing list values, append the specified value to the list. Can only be used in combination with the --config , --system , and --pid options.
--remove	When editing list values, remove the specified value from the list. Can only be used in combination with the --config , --system , and --pid options.
--delimiter	Specifies the delimiter to use in combination with the --append and --remove options. Default is <code>,</code> (comma).
--resource	When editing with the visual text editor, specifies the name of the resource to edit.
-i, --import-pid	Imports the PIDs that are edited, from local OSGi Config Admin.
Profile	<i>(Required)</i> Name of the profile to edit.
Version	Version of the profile to edit. Defaults to the current default version (use version-list to find the current default).

NAME

fabric:profile-list – lists all profiles that belong to the specified version (where the version defaults to the current default version)

SYNOPSIS

```
fabric:profile-list [
  --help
] [
  --version version
] [
  --hidden
]
```

DESCRIPTION**ARGUMENTS**

[Table 10.37, “fabric:profile-list Arguments”](#) describes the command's arguments.

Table 10.37. fabric:profile-list Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>--version</code>	Specifies the version of the profiles to list. Defaults to the current default version (use <code>version-list</code> to find the current default).
<code>--hidden</code>	Shows hidden profiles.

NAME

`fabric:require-profile-delete` – deletes requirements on the specified profile

SYNOPSIS

```
fabric:require-profile-delete [
    --help
] {
    Profile
}
```

ARGUMENTS

[Table 10.38, “fabric:require-profile-delete Arguments”](#) describes the command's arguments.

Table 10.38. fabric:require-profile-delete Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>Profile</i>	A profile ID.

NAME

`fabric:require-profile-list` – lists all profile requirements in the current fabric

SYNOPSIS


```
fabric:require-profile-list [
  --help
]
```

DESCRIPTION

For example, if both the `example-camel` profile and the `example-cxf` profile have requirements set, you could see output like the following:

```
karaf@root> fabric:require-profile-list
[profile]                [# minimum]    [# maximum]
[depends on]
example-camel            2              4
example-cxf              2              4
```

ARGUMENTS

[Table 10.39, “fabric:require-profile-list Arguments”](#) describes the command's arguments.

Table 10.39. fabric:require-profile-list Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

NAME

`fabric:require-profile-set` – associates requirements with the specified profile

SYNOPSIS

```
fabric:require-profile-set [
  --help
] [
  --minimum MinInstance
] [
  --maximum MaxInstance
] [
  --dependsOn Dependency
] {
  Profile
}
```

DESCRIPTION

Requirements associated with a profile are used to assess the health of

the current fabric. Profile requirements are entirely passive. For example, if the number of running instances of a profile is less than the minimum or greater than the maximum, monitoring tools can be configured to indicate a problem or to trigger an alert. Otherwise, the requirements have no effect on the fabric.

In Fuse IDE a green/red bar indicates what proportion of the required profile instances are currently running in the fabric.

For example, to require a range of 2 to 4 running instances of the example-camel profile, you would enter the following command:

```
karaf@root> require-profile-set --minimum 2 --maximum 4 example-camel
```

ARGUMENTS

Table 10.40, “fabric:require-profile-set Arguments” describes the command's arguments.

Table 10.40. fabric:require-profile-set Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>--minimum</code>	The minimum number of instances of this profile expected to be running in the fabric.
<code>--maximum</code>	The maximum number of instances of this profile expected to be running in the fabric.
<code>--dependsOn</code>	The profile IDs that must be provisioned before this profile.
<i>Profile</i>	A profile ID.

NAME

`fabric:status` – displays the current status of the fabric, based on the configured profile requirements

SYNOPSIS

```
fabric:status [
  --help
]
```

DESCRIPTION

This command summarizes the health of the fabric, based on requirements previously configured by the `fabric:require-profile-set` command. For example, if you configured the `example-camel` profile to require a minimum of two instances and a maximum of four instances, and there is currently only one instance running, the `example-camel` profile would get a health rating of 50%.

The `fabric:status` command produces output like the following:

```
karaf@root> fabric:status
[profile]           [instances]   [health]
cloud               1             100%
example-camel       0             0%
example-cxf         0             0%
fabric              1             100%
fabric-ensemble-0000-1 1             100%
```

ARGUMENTS

[Table 10.41, “fabric:status Arguments”](#) describes the command's arguments.

Table 10.41. `fabric:status` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

RELATED TOPICS

For more details, see:

- [fabric:require-profile-set](#)
- [fabric:require-profile-list](#)

- [fabric:require-profile-delete](#)

NAME

fabric:version-create – create a new version

SYNOPSIS

```
fabric:version-create [
  --help
] [
  --parent parentVersion
] {
  Version
}
```

DESCRIPTION

Create a new version, by default copying all of the profiles *from the current latest version* into the new version. You can specify which version to copy the profiles from using the `--parent` option. If no version is specified, the command creates a new minor version by default. For example:

```
karaf@root> fabric:version-list
[version]      [default] [# containers]
1.0            true      1
karaf@root> fabric:version-create
Created version: 1.1 as copy of: 1.0
karaf@root> fabric:version-list
[version]      [default] [# containers]
1.0            true      1
1.1            false     0
```

ARGUMENTS

[Table 10.42, “fabric:version-create Arguments”](#) describes the command's arguments.

Table 10.42. fabric:version-create Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>--default</code>	Set the created version to be the new default version.

Argument	Interpretation
<code>--parent</code>	The parent version. By default, use the latest version as the parent.
<i>Version</i>	The new version to create. If not specified, defaults to the next minor version.

NAME

`fabric:version-delete` – delete the specified version

SYNOPSIS

```
fabric:version-delete [
  --help
] {
  Version
}
```

DESCRIPTION

Delete the specified version.

**WARNING**

This command also deletes all of the profile data associated with the deleted version.

ARGUMENTS

[Table 10.43, “fabric:version-delete Arguments”](#) describes the command's arguments.

Table 10.43. fabric:version-delete Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>Version</i>	<i>(Required)</i> The version to delete.

NAME

fabric:version-list – lists the existing versions

SYNOPSIS

```
fabric:version-list [
    --help
]
```

DESCRIPTION

For example:

```
karaf@root> fabric:version-list
[version]      [default] [# containers]
1.0            true      1
1.1            false     0
```

ARGUMENTS

[Table 10.44, “fabric:version-list Arguments”](#) describes the command's arguments.

Table 10.44. fabric:version-list Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

NAME

fabric:version-set-default – set the new default version (must be one of the existing versions)

SYNOPSIS

```
fabric:version-set-default [
    --help
] {
    Version
}
```

DESCRIPTION

Many of the fabric console commands work with a default version. For example, when you create a new profile with `fabric:profile-create`, the new

profile is created in the default version by default. The `fabric:version-set-default` changes the default version that is used by these commands.

ARGUMENTS

Table 10.45, “`fabric:version-set-default` Arguments” describes the command's arguments.

Table 10.45. `fabric:version-set-default` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>Version</i>	<i>(Required)</i> Version number to use as the new default version.

CHAPTER 11. FEATURES CONSOLE COMMANDS

The features commands allow you to provision entire applications using the Apache Karaf features facility. Features allow you to provision a collection of bundles using a single name.

Type `features:` then press `Tab` at the `karaf>` prompt to view the available commands.

NAME

`features:addurl`, `addurl` – registers one or more URLs to feature repositories with the container

SYNOPSIS

```
features:addurl [
  --help
] [[
  -i
] | [
  --install-all
]] {
  urls
}
```

DESCRIPTION

Each feature repository defines one or more features, and each feature is made up of a collection of bundles that work together to provide some functionality. When a feature is loaded, the container loads any required bundles that are not already present into the container and activates them.

ARGUMENTS

[Table 11.1, “features:addurl Arguments”](#) describes the command's arguments.

Table 11.1. features:addurl Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-i</code> , <code>--install-all</code>	Install all of the features in the specified feature repository URLs.

Argument	Interpretation
<i>urls</i>	One or more repository URLs separated by whitespaces.

NAME

`features:chooseurl`, `chooseurl` – registers the feature repository URL for a well known project

SYNOPSIS

```
features:chooseurl [
  --help
] {
  project
} {
  version
}
```

DESCRIPTION

uses a number of features implemented by well-known projects. To simplify the process of adding their feature repositories, the `chooseurl` command allows you to add a feature repository without knowing its Maven URL. The list of projects supported by `chooseurl` is configured by the `org.apache.karaf.features.repos` PID.

ARGUMENTS

[Table 11.2, “features:chooseurl Arguments”](#) describes the command's arguments.

Table 11.2. features:chooseurl Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>feature</i>	Specifies the project name for the feature repository to add.
<i>version</i>	Specifies the version of the project's feature repository to add.

NAME

features:info – show information about the specified feature with the optionally specified version

SYNOPSIS

```
features:info [
  --help
] [[
  -c
] | [
  --configuration
]] [[
  -b
] | [
  --bundle
]] [[
  -t
] | [
  --tree
]] [[
  -d
] | [
  --dependency
]] {
  featureName
} {
  version
}
```

ARGUMENTS

[Table 11.3, “features:info Arguments”](#) describes the command's arguments.

Table 11.3. features:info Arguments

Argument	Interpretation
--help	Displays the online help for this command
-c, --configuration	Display configuration information.
-b, --bundle	Display bundle information.
-t, --tree	Display feature tree.
-d, --dependency	Display dependency information.
<i>command</i>	

NAME

features:install – installs a feature

SYNOPSIS

```
features:install [
  --help
] {
  name
} [
  version
]
```

ARGUMENTS

[Table 11.4, “features:install Arguments”](#) describes the command's arguments.

Table 11.4. features:install Arguments

Argument	Interpretation
--help	Displays the online help for this command
<i>name</i>	The name of the feature to install
<i>version</i>	The version of the feature

NAME

features:list – Lists all existing features available from the defined repositories

SYNOPSIS

```
features:list [
  --help
] [[
  -i
] | [
  --installed
]]
```

ARGUMENTS

[Table 11.5, “features:list Arguments”](#) describes the command's arguments.

Table 11.5. features:list Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-i, --installed</code>	Displays the list of all installed features

NAME

`features:listurl` – lists the features repository URLs

SYNOPSIS

```
features:listurl [
  --help
] [[
  -v
] | [
  --validate
]] [[
  -vo
] | [
  --verbose
]]
```

ARGUMENTS

[Table 11.6, “features:listurl Arguments”](#) describes the command's arguments.

Table 11.6. features:listurl Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-v, --validate</code>	Validate current version of descriptors.
<code>-vo, --verbose</code>	Shows validation output.

NAME

`features:listVersions`, `listVersions` – lists all versions of a feature available from the current feature repositories

SYNOPSIS

```
features:listVersions [
  --help
] {
  feature
}
```

ARGUMENTS

[Table 11.7, “features:listVersions Arguments”](#) describes the command's arguments.

Table 11.7. `features:listVersions` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>feature</i>	Name of a feature.

NAME

`features:refreshUrl` – reloads the list of available features from the repositories

SYNOPSIS

```
features:refreshUrl [
  --help
] {
  urls
}
```

ARGUMENTS

[Table 11.8, “features:refreshUrl Arguments”](#) describes the command's arguments.

Table 11.8. `features:refreshUrl` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

Argument	Interpretation
<i>urls</i>	The repository URLs to reload (leave empty for all).

NAME

features:removeUrl – removes the specified list of repository URLs from the features service

SYNOPSIS

```
features:removeUrl [
  --help
] {
  urls
}
```

ARGUMENTS

[Table 11.9, “features:removeUrl Arguments”](#) describes the command's arguments.

Table 11.9. features:removeUrl Arguments

Argument	Interpretation
--help	Displays the online help for this command
-n,--interval	
<i>urls</i>	One or more repository URLs separated by whitespace.

NAME

features:removeRepository – removes the specified repository from the features service

SYNOPSIS

```
features:removeRepository [
  --help
] {
  repository
}
```

ARGUMENTS

Table 11.10, “[features:removeRepository Arguments](#)” describes the command's arguments.

Table 11.10. `features:removeRepository` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>repository</i>	The name of a features repository.

NAME

`features:uninstall` – uninstalls a feature with the specified name and version

SYNOPSIS

```
features:uninstall [
  --help
] {
  features
}
```

ARGUMENTS

Table 11.11, “[features:uninstall Arguments](#)” describes the command's arguments.

Table 11.11. `features:uninstall` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>features</i>	A space-separated list of features to uninstall, where each feature is specified in the format <i>feature[/version]</i> (that is, the version is optional).

CHAPTER 12. JAAS CONSOLE COMMANDS

The `jaas` commands are used for editing JAAS realm and user data. Editing a JAAS realm is done in two stages. The changes are placed in a queue until they are applied by executing the `jaas:update`.

When editing JAAS settings the commands are used as follows:

1. **Start the editing session.**

```
jaas:manage
```

2. **Edit the realm's user data.**

- **jass:users**

Lists all of the users.

- **jass:useradd**

Add a new user.

- **jass:userdel**

Delete a user.

- **jass:roleadd**

Add a new role to a user.

- **jass:roledel**

Delete a role from a user.

- **jass:pending**

Lists all of the pending changes that have been made to the realms, but have not been applied to the container.

3. **Apply the changes to the JAAS realm and ends the editing session.**

jaas:update

You can abandon an editing session using `jaas:cancel` before the changes applied to the JAAS settings.

Type `jaas:` then press `Tab` at the prompt to view the available commands.

NAME

`jaas:cancel`, `cancel` – cancels a JAAS editing session without applying the pending changes

SYNOPSIS

```
jaas:cancel [
    --help
]
```

DETAILS

When editing a JAAS realm, the changes are buffered until the editing session is closed. The `jaas:cancel` command clears the buffer without saving the changes and closes the editing session.

You can see a list of the buffered changes using the `jaas:pending` command.

ARGUMENTS

[Table 12.1, “jaas:cancel Arguments”](#) describes the command's arguments.

Table 12.1. `jaas:cancel` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

NAME

`jaas:manage`, `manage` – opens a JAAS realm for editing

SYNOPSIS

```
jaas:manage [
    --help
] {[
    --realm realm
] | [
    --index index
]} [
    --module module
```

```

] [
  --force
]

```

DETAILS

The `jaas:manage` command is the first step in editing a JAAS realm. It opens the realm so that calls to the `jaas:*` editing commands will update the selected realm. The edits made by the `jaas:*` editing commands are placed in a buffer associated with the selected realm and not written to the realm until the editing session is ended by the `jaas:update` command.

If you use the `jaas:manage` command before saving the changes to a realm that is open for editing, the changes to the previously open realm are abandoned. The pending edits for the previous realm are cleared without being saved.

While editing a realm you can get a list of the pending changes using the `jaas:pending` command.

ARGUMENTS

Table 12.2, “`jaas:manage` Arguments” describes the command's arguments.

Table 12.2. `jaas:manage` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>--realm</code>	Select the realm to edit by specifying its realm name.
<code>--index</code>	Select the realm to edit by specifying its index.
<code>--module</code>	Specify which of the realm's login modules are to be edited.
<code>--force</code>	Force the switch to the specified realm. If a different realm was already opened for editing its changes are abandoned without being applied.

EXAMPLES

You can select the realm to manage *either* by specifying its realm name *or* by specifying its index.

If the installed realm names are all distinct (which you can check using `jaas:realms`), you can identify the realm to manage by specifying the `--realm` option. For example, if the container is a standalone instance (no fabric installed), you can start to edit the karaf realm as follows:

```
jaas:manage --realm karaf
```

If the container belongs to a fabric, however, the `fabric-jaas` feature automatically installs another realm named `karaf` at a higher priority, so that it overrides the default `karaf` realm. For example, in a fabric, the `jaas:realms` command returns a list similar to the following:

```
Index Realm Module Class 1 karaf
org.apache.karaf.jaas.modules.properties.PropertiesLoginModule 2 karaf
org.fusesource.fabric.jaas.ZookeeperLoginModule
```

In this case, you must identify the realm to manage using the `--index` option, specifying one of the index values from the list. The current active `karaf` realm is the `ZookeeperLoginModule`, which is selected by the index value, `2`, as follows:

```
jaas:manage --index 2
```

NAME

`jaas:pending`, `pending` – lists the changes waiting to be applied to the realm being edited

SYNOPSIS

```
jaas:pending [
    --help
]
```

DETAILS

When editing a JAAS realm, the changes are stored in a buffer until the editing session is closed. The `jaas:pending` command shows a list of the changes buffered during the currently open editing session.

The `jaas:update` command saves the changes and closes the editing session.

The `jaas:cancel` command clears the buffer without saving the changes and

closes the editing session.

ARGUMENTS

Table 12.3, “[jaas:pending Arguments](#)” describes the command's arguments.

Table 12.3. `jaas:pending Arguments`

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.

NAME

`jaas:realms, realms` – lists the JAAS realms know to the container

SYNOPSIS

```
jaas:realms [
    --help
]
```

ARGUMENTS

Table 12.4, “[jaas:realms Arguments](#)” describes the command's arguments.

Table 12.4. `jaas:realms Arguments`

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.

NAME

`jaas:roleadd, roleadd` – adds a role to a user

SYNOPSIS

```
jaas:roleadd [
    --help
] {
    username
} {
    role
}
```

DETAILS

When editing a JAAS realm, the changes are buffered until the editing session is closed. When you add a new role using the `jaas:roleadd` command, the change is stored in the buffer and does not take effect until the editing session is closed.

The `jaas:update` command saves the changes and closes the editing session.

The `jaas:cancel` command clears the buffer without saving the changes and closes the editing session.

ARGUMENTS

[Table 12.5, “jaas:roleadd Arguments”](#) describes the command's arguments.

Table 12.5. `jaas:roleadd` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<i>username</i>	Specifies the name of the user to modify.
<i>role</i>	Specifies the role which is appended to the user data.

NAME

`jaas:roledel`, `roledel` – deletes a role from a user

SYNOPSIS

```
jaas:roledel [
  --help
] {
  username
} {
  role
}
```

DETAILS

When editing a JAAS realm, the changes are buffered until the editing session is closed. When you delete a role using the `jaas:roledel` command, the

change is stored in the buffer and does not take effect until the editing session is closed.

The `jaas:update` command saves the changes and closes the editing session.

The `jaas:cancel` command clears the buffer without saving the changes and closes the editing session.

ARGUMENTS

Table 12.6, “`jaas:roledel` Arguments” describes the command's arguments.

Table 12.6. `jaas:roledel` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<i>username</i>	Specifies the name of the user to modify.
<i>role</i>	Specifies the role which is removed from the user data.

NAME

`jaas:update` – applies all pending changes to the JAAS realm and closes the editing session

SYNOPSIS

```
jaas:update [
    --help
]
```

DETAILS

When editing a JAAS realm, the changes are buffered until the editing session is closed. The `jaas:update` command saves the buffered changes to the realm and closes the editing session.

You can see a list of the buffered changes using the `jaas:pending` command.

ARGUMENTS

[Table 12.7, “jaas:update Arguments”](#) describes the command's arguments.

Table 12.7. jaas:update Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.

NAME

`jaas:useradd`, `useradd` – adds a user to the JAAS realm being edited

SYNOPSIS

```
jaas:useradd [
  --help
] {
  username
} {
  password
}
```

DETAILS

When editing a JAAS realm, the changes are buffered until the editing session is closed. When you add a new user using the `jaas:useradd` command, the change is stored in the buffer and does not take effect until the editing session is closed.

The `jaas:update` command saves the changes and closes the editing session.

The `jaas:cancel` command clears the buffer without saving the changes and closes the editing session.

ARGUMENTS

[Table 12.8, “jaas:useradd Arguments”](#) describes the command's arguments.

Table 12.8. jaas:useradd Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.

Argument	Interpretation
<i>username</i>	Specifies the name of the user to add.
<i>password</i>	Specifies the password used to authenticate the user.

NAME

jaas:userdel, userdel – deletes a user from the JAAS realm being edited

SYNOPSIS

```
jaas:userdel [
  --help
] {
  username
}
```

DETAILS

When editing a JAAS realm, the changes are buffered until the editing session is closed. When you delete a user using the `jaas:useradd` command, the change is stored in the buffer and does not take effect until the editing session is closed.

The `jaas:update` command saves the changes and closes the editing session.

The `jaas:cancel` command clears the buffer without saving the changes and closes the editing session.

ARGUMENTS

[Table 12.9, “jaas:userdel Arguments”](#) describes the command's arguments.

Table 12.9. jaas:userdel Arguments

Argument	Interpretation
--help	Displays the online help for this command.
<i>username</i>	Specifies the name of the user to add.

NAME

`jaas:users, users` – lists the users in the JAAS realm being edited

SYNOPSIS

```
jaas:users [
    --help
]
```

ARGUMENTS

[Table 12.10, “jaas:users Arguments”](#) describes the command's arguments.

Table 12.10. jaas:users Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.

CHAPTER 13. JBI CONSOLE COMMANDS

The `jbi` commands allow you to manage JBI artifacts that are deployed in the Red Hat JBoss Fuse runtime. For information about working with JBI artifacts in JBoss Fuse, see ["Using Java Business Integration"](#).

Type `jbi:` then press Tab at the `JBossFuse:karaf@root>` prompt to view the available commands.

NAME

`jbi:list` – lists JBI endpoints

SYNOPSIS

```
jbi:list [
    --help
]
```

ARGUMENTS

This command takes the following arguments.

Table 13.1. `jbi:list` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

NAME

`jbi:shutdown` – shuts down a JBI artifact

SYNOPSIS

```
jbi:shutdown [
    --help
] [[
    -a
] | [
    --service-assembly
] serviceAssembly] [[
    -c
] | [
    --component
] component] [
```

```

    --force
  ] {
    VAL
  }

```

ARGUMENTS

This command takes the following arguments.

Table 13.2. `jbi:shutdown` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-a, --service-assembly</code>	Specifies a service assembly
<code>-c, --component</code>	Specifies a component
<code>--force</code>	Forces a shutdown of the specified JBI artifact
<code>VAL</code>	The name of the JBI artifact

NAME

`jbi:start` – starts a JBI artifact

SYNOPSIS

```

jbi:start [
  --help
] [[
  -a
] | [
  --service-assembly
] serviceAssembly] [[
  -c
] | [
  --component
] component] {
  VAL
}

```

ARGUMENTS

This command takes the following arguments.

Table 13.3. `jbi:start` Arguments

Argument	Interpretation
--help	Displays the online help for this command
-a, --service-assembly	Specifies a service assembly
-c, --component	Specifies a component
VAL	The name of the JBI artifact

NAME

jbi:stop – stops a JBI artifact

SYNOPSIS

```
jbi:stop [
  --help
] [[
  -a
] | [
  --service-assembly
] serviceAssembly] [[
  -c
] | [
  --component
] component] {
  VAL
}
```

ARGUMENTS

This command takes the following arguments.

Table 13.4. jbi:stop Arguments

Argument	Interpretation
--help	Displays the online help for this command
-a, --service-assembly	A service assembly
-c, --component	A component
VAL	The name of the JBI artifact

CHAPTER 14. LOG CONSOLE COMMANDS

The log commands allow you to display and change log levels.

Type `log:` then press `Tab` at the prompt to view the available commands.

NAME

`log:clear` – clears the log

SYNOPSIS

```
log:clear [
    --help
]
```

ARGUMENTS

[Table 14.1, “log:clear Arguments”](#) describes the command's arguments.

Table 14.1. log:clear Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

NAME

`log:display`, `display`, `ld` – displays log entries

SYNOPSIS

```
log:display [
    --help
] [
    -p pattern
] [
    -n numLines
] [
    --no-color
]
```

ARGUMENTS

[Table 14.2, “log:display Arguments”](#) describes the command's arguments.

Table 14.2. `log:display` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-p</code>	The pattern for formatting the output
<code>-n</code>	The number of entries to display
<code>--no-color</code>	Do not use syntax highlighting when displaying the log.

NAME

`log:display-exception`, `display-exception`, `lde` – displays the last thrown exception from the log

SYNOPSIS

```
log:display-exception [
    --help
]
```

ARGUMENTS

[Table 14.3, “`log:display-exception` Arguments”](#) describes the command's arguments.

Table 14.3. `log:display-exception` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

NAME

`log:get`, `get` – shows the log level

SYNOPSIS

```
log:get [
    --help
] {
    logger
}
```

ARGUMENTS

Table 14.4, “log:get Arguments” describes the command's arguments.

Table 14.4. log:get Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>logger</code>	Specifies the logger name, ALL , or ROOT . The default is ROOT .

NAME

`log:set`, `set` – sets the log level

SYNOPSIS

```
log:set [
  --help
] { [
  DEFAULT
] | [
  TRACE
] | [
  DEBUG
] | [
  INFO
] | [
  WARN
] | [
  ERROR
]} {
  logger
}
```

ARGUMENTS

Table 14.5, “log:set Arguments” describes the command's arguments.

Table 14.5. log:set Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

Argument	Interpretation
level	Specifies the logging level.
logger	Specifies the logger name. The default is ROOT .

NAME

log:tail – continually displays log entries

SYNOPSIS

```
log:tail [
  --help
] [
  -p pattern
] [
  -n numLines
] [
  --no-color
]
```

ARGUMENTS

[Table 14.6, “log:tail Arguments”](#) describes the command's arguments.

Table 14.6. log:tail Arguments

Argument	Interpretation
--help	Displays the online help for this command
-p	The pattern for formatting the output
-n	The number of entries to display
--no-color	Do not use syntax highlighting when displaying the log.

CHAPTER 15. THE NMR:LIST COMMAND

The `nmr:listcommand` allows you to list normalized message router (NMR) endpoints. The NMR enables the interaction between different components in the OSGi container. The main purpose of the NMR is to mediate between service consumers and service providers.

NAME

`nmr:list` – lists NMR endpoints

SYNOPSIS

```
nmr:list [
    --help
]
```

ARGUMENTS

This command takes the following arguments.

Table 15.1. `nmr:list` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

CHAPTER 16. OBR CONSOLE COMMANDS

The `obr` commands allow you to access the OSGi Bundle Repository (OBR) Service API.

**NOTE**

This feature is not installed by default. To install the `obr` shell, run the following command:

```
JBossFuse:karaf@root:> features:install obr
```

Type `obr:` then press Tab at the `JBossFuse:karaf@root>` prompt to view the available commands.

NAME

`obr:addUrl` – adds a list of repository URLs to the OBR service

SYNOPSIS

```
obr:addUrl [
  --help
] {
  urls
}
```

ARGUMENTS

This command takes the following arguments.

Table 16.1. `obr:addUrl` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>urls</code>	The repository URLs to add to the OBR service, separated by whitespaces

NAME

obr:deploy – deploys a list of bundles using the OBR service

SYNOPSIS

```
obr:deploy [
  --help
] {
  bundles
}
```

ARGUMENTS

This command takes the following arguments.

Table 16.2. obr:deploy Arguments

Argument	Interpretation
--help	Displays the online help for this command
<i>bundles</i>	A list of bundle names to deploy, separated by whitespaces

NAME

obr:info – prints information about OBR bundles

SYNOPSIS

```
obr:info [
  --help
] {
  bundles
}
```

ARGUMENTS

This command takes the following arguments.

Table 16.3. obr:info Arguments

Argument	Interpretation
--help	Displays the online help for this command

Argument	Interpretation
<i>bundles</i>	Specifies the bundles to query for information, separated by whitespaces

NAME

obr:list – lists OBR bundles

SYNOPSIS

```
obr:list [
  --help
] {
  args
}
```

ARGUMENTS

This command takes the following arguments.

Table 16.4. obr:list Arguments

Argument	Interpretation
--help	Displays the online help for this command
<i>args</i>	The arguments

NAME

obr:listUrl – displays the repository URLs currently associated with the OBR service

SYNOPSIS

```
obr:listUrl [
  --help
]
```

ARGUMENTS

This command takes the following arguments.

Table 16.5. obr:listUrl Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

NAME

`obr:refreshUrl` – reloads the repositories to obtain a fresh list of bundles

SYNOPSIS

```
obr:refreshUrl [
  --help
] {
  urls
}
```

ARGUMENTS

This command takes the following arguments.

Table 16.6. `obr:refreshUrl` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>urls</code>	The repository URLs to refresh (leave empty for all)

NAME

`obr:removeUrl` – removes a list of repository URLs from the OBR service

SYNOPSIS

```
obr:removeUrl [
  --help
] {
  urls
}
```

ARGUMENTS

This command takes the following arguments.

Table 16.7. `obr:removeUrl` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>urls</code>	The repository URLs to remove from the OBR service, separated by whitespace

NAME

`obr:source` – downloads the sources for an OBR bundle

SYNOPSIS

```
obr:source [
  --help
] [
  -x
] {
  folder
} {
  bundles
}
```

ARGUMENTS

This command takes the following arguments.

Table 16.8. `obr:source` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-x</code>	Extracts the archive
<code>folder</code>	The local directory or folder for storing sources
<code>bundles</code>	A list of bundles to download the sources for

NAME

`obr:start` – deploys and starts a list of bundles using OBR

SYNOPSIS

```
obr:start [
```

```
--help  
] {  
  bundles  
}
```

ARGUMENTS

This command takes the following arguments.

Table 16.9. obr:start Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>bundles</i>	List of bundle names to deploy, separated by whitespaces

CHAPTER 17. OSGI CONSOLE COMMANDS

The `osgi` commands provide for managing the OSGi runtime. It includes commands for listing OSGi bundles and services and managing bundle lifecycles.

Type `osgi`: then press Tab at the prompt to view the available commands.

NAME

`osgi:bundle-level`, `bundle-level` – gets or sets the start level of a given bundle

SYNOPSIS

```
osgi:bundle-level [
  --help
] [
  --force
] {
  id
} [
  startLevel
]
```

ARGUMENTS

[Table 17.1, “osgi:bundle-level Arguments”](#) describes the command's arguments.

Table 17.1. `osgi:bundle-level` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>--force</code>	Forces the command to execute.
<i>id</i>	Specifies the id for the bundle.
<i>startLevel</i>	Specifies the new start level for the bundle.

NAME

osgi:bundle-services, bundle-services – lists the OSGi services provided by a bundle

SYNOPSIS

```
osgi:bundle-level [
  -u
  ] [
  -p
  ] [
  -a
  ] [
  --help
  ] [
  --force
  ] {
  id
  }
```

ARGUMENTS

[Table 17.2, “osgi:bundle-services Arguments”](#) describes the command's arguments.

Table 17.2. osgi:bundle-services Arguments

Argument	Interpretation
-u	Displays the services used by the bundle.
-p	Displays the properties for each service.
-a	Displays all of the services provided by the bundle including the Apache Karaf commands which are hidden by default.
--help	Displays the online help for this command.
--force	Forces the command to execute.
id	Specifies the id for the bundle.

NAME

osgi:classes, classes – lists all of the classes in the specified bundle or bundles

SYNOPSIS

```

osgi:classes [
  --help
] [
  --force
] [[
  -a
] | [
  --display-all-files
]] {
  ids
}

```

ARGUMENTS

[Table 17.3, “osgi:classes Arguments”](#) describes the command's arguments.

Table 17.3. osgi:classes Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>--force</code>	Forces the command to execute.
<code>-a, --display-all-files</code>	Also lists the files contained in the bundles.
<code>ids</code>	Space-separated list of bundle IDs.

NAME

`osgi:find-class, find-class` – locates a specified class in any deployed bundle

SYNOPSIS

```

osgi:find-class [
  --help
] {
  className
}

```

ARGUMENTS

[Table 17.4, “osgi:find-class Arguments”](#) describes the command's arguments.

Table 17.4. osgi:find-class Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>className</code>	Class name or partial class name to find.

NAME

`osgi:headers, headers` – displays the headers of a specified OSGi bundle

SYNOPSIS

```
osgi:headers [
  --help
] {
  id
  ...}
```

ARGUMENTS

[Table 17.5, “osgi:headers Arguments”](#) describes the command's arguments.

Table 17.5. osgi:headers Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>id</code>	Specifies a space delimited list of bundle IDs.

NAME

`osgi:info` – displays detailed information about OSGi bundles

SYNOPSIS

```
osgi:info [
  --help
] {
  id
  ...}
```

ARGUMENTS

[Table 17.6, “osgi:info Arguments”](#) describes the command's arguments.

Table 17.6. `osgi:info` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>id</i>	Specifies a space delimited list of bundle IDs.

NAME

`osgi:install`, `install` – installs one or more OSGi bundles

SYNOPSIS

```
osgi:install [
  --help
] [[
  -s
] | [
  --start
]] {
  url
  ...}
```

ARGUMENTS

[Table 17.7, “osgi:install Arguments”](#) describes the command's arguments.

Table 17.7. `osgi:install` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-s, --start</code>	Starts the bundles after installation
<i>url</i>	Specifies a space delimited list of bundle URLs.

NAME

`osgi:list`, `list` – lists the installed bundles whose start level equals or exceeds the specified threshold

SYNOPSIS

```
osgi:list [
  --help
```

```

] [
  -u
] [
  -t threshold
] [
  -l
] [
  -s
]

```

ARGUMENTS

[Table 17.8, “osgi:list Arguments”](#) describes the command's arguments.

Table 17.8. osgi:list Arguments

Argument	Interpretation
--help	Displays the online help for this command
-u	Shows the update locations
-t	Specifies the start level threshold. The default is the value of the karaf.systemBundlesStartLevel property whose default value is 50.
-l	Shows the locations of the bundles
-s	Shows the symbolic names of the bundles

NAME

osgi:ls, ls – lists OSGi services

SYNOPSIS

```

osgi:ls [
  --help
] [
  -a
] [
  -u
] [
  --force
] [
  id
  ...]

```

ARGUMENTS

Table 17.9, “[osgi:ls Arguments](#)” describes the command's arguments.

Table 17.9. `osgi:ls` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-a</code>	Lists all services
<code>-u</code>	Lists the services in use
<code>--force</code>	Forces the command to execute
<i>id</i>	Specifies a space separated list of bundle IDs.

NAME

`osgi:refresh`, `refresh` – refreshes an OSGi bundle

SYNOPSIS

```
osgi:refresh [
  --help
] [
  --force
] {
  id
  ...}
```

ARGUMENTS

Table 17.10, “[osgi:refresh Arguments](#)” describes the command's arguments.

Table 17.10. `osgi:refresh` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>--force</code>	Forces the command to execute.
<i>id</i>	Specifies a space delimited list of bundle IDs.

NAME

osgi:resolve, resolve – resolves an OSGi bundle's dependencies

SYNOPSIS

```
osgi:resolve [
  --help
] [
  --force
] {
  id
...}
```

ARGUMENTS

[Table 17.11, “osgi:resolve Arguments”](#) describes the command's arguments.

Table 17.11. osgi:resolve Arguments

Argument	Interpretation
--help	Displays the online help for this command.
--force	Forces the command to execute.
id	Specifies a space delimited list of bundle IDs.

NAME

osgi:restart, restart – stops and restarts an OSGi bundle

SYNOPSIS

```
osgi:restart [
  --help
] [
  --force
] {
  id
...}
```

ARGUMENTS

[Table 17.12, “osgi:restart Arguments”](#) describes the command's arguments.

Table 17.12. `osgi:restart` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>--force</code>	Forces the command to execute.
<i>id</i>	Specifies a space delimited list of bundle IDs.

NAME

`osgi:shutdown`, `shutdown` – stops the OSGi framework

SYNOPSIS

```
osgi:shutdown [
  --help
] [[
  -f
] | [
  --force
]] [[
  hh:mm
] | [
  +m
]]
```

ARGUMENTS

[Table 17.13, “osgi:shutdown Arguments”](#) describes the command's arguments.

Table 17.13. `osgi:shutdown` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>-f</code> , <code>--force</code>	Forces the command to execute.
<i>hh:mm</i>	Specifies the time to shut down the broker in hours and minutes. The time is specified in 24 hour time. For example, 13:30 specifies that the container will shutdown at 1:30pm.

Argument	Interpretation
+m	Specifies the time, in minutes, to pause before shutting down the OSGi framework. For example, +30 specifies that the container will wait thirty minutes before shutting down the OSGi framework.

NAME

osgi:start, start – starts an OSGi bundle

SYNOPSIS

```
osgi:start [
    --help
] [
    --force
] {
    id
...}
```

ARGUMENTS

[Table 17.14, “osgi:start Arguments”](#) describes the command's arguments.

Table 17.14. osgi:start Arguments

Argument	Interpretation
--help	Displays the online help for this command.
--force	Forces the command to execute.
id	Specifies a space delimited list of bundle IDs.

NAME

osgi:start-level, start-level – gets or sets the OSGi framework's active start level

SYNOPSIS

```
osgi:start [
    --help
] [
```

```

    level
]

```

ARGUMENTS

[Table 17.15, “osgi:start-level Arguments”](#) describes the command's arguments.

Table 17.15. osgi:start-level Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<i>level</i>	Specifies the new start level to set.

NAME

`osgi:stop`, `stop` – stops an OSGi bundle

SYNOPSIS

```

osgi:stop [
    --help
] [
    --force
] {
    id
...}

```

ARGUMENTS

[Table 17.16, “osgi:stop Arguments”](#) describes the command's arguments.

Table 17.16. osgi:stop Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>--force</code>	Forces the command to execute.
<i>id</i>	Specifies a space delimited list of bundle IDs.

NAME

osgi:uninstall, uninstall – uninstalls an OSGi bundle

SYNOPSIS

```
osgi:uninstall [
  --help
] [
  --force
] {
  id
...}
```

ARGUMENTS

[Table 17.17, “osgi:uninstall Arguments”](#) describes the command's arguments.

Table 17.17. osgi:uninstall Arguments

Argument	Interpretation
--help	Displays the online help for this command.
--force	Forces the command to execute.
<i>id</i>	Specifies a space delimited list of bundle IDs.

NAME

osgi:update, update – updates an OSGi bundle

SYNOPSIS

```
osgi:update [
  --help
] [
  --force
] {
  id
} [
  location
]
```

ARGUMENTS

[Table 17.18, “osgi:update Arguments”](#) describes the command's arguments.

Table 17.18. `osgi:update` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>--force</code>	Forces the command to execute.
<i>id</i>	Specifies ID of the bundle.
<i>location</i>	Specifies the location from which the update is loaded. If no location is specified the container will use either the bundle's <code>Bundle-UpdateLocation</code> property or the bundle's original location.

CHAPTER 18. PACKAGES CONSOLE COMMANDS

The `packages` commands are used for showing all packages imported and exported by the OSGi bundles currently installed.

Type `packages:` then press Tab at the prompt to view the available commands.

NAME

`packages:exports`, `exports` – displays the packages exported OSGi bundles

SYNOPSIS

```
packages:export [
  --help
] [[
  -d
] | [
  --details
]] [
  -s
] [[
  -i
] | [
  --imports
]] [
  id
  ...]
```

ARGUMENTS

[Table 18.1, “package:exports Arguments”](#) describes the commands arguments.

Table 18.1. package:exports Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-d</code> , <code>--details</code>	Reformat the output in master/detail layout, which makes it easier to see how related details are grouped together.

Argument	Interpretation
-s	Show the Symbolic name column, which shows the symbolic name of the bundle to which the exported package belongs.
-i, --imports	Show the Imported by column, which lists all of the bundles that import the exported package.
id	Specifies a whitespace separated list of bundle IDs to check.

NAME

packages:imports, imports – displays the packages imported by OSGi bundles

SYNOPSIS

```
packages:imports [
  --help
] [[
  -i
] | [
  --show-importer
]] [
  id
  ...]
```

ARGUMENTS

[Table 18.2, “package:imports Arguments”](#) describes the commands arguments.

Table 18.2. package:imports Arguments

Argument	Interpretation
--help	Displays the online help for this command
-i, --show-importer	Show the bundle(s) that import a package.
id	Specifies a whitespace separated list of bundle IDs to check.

CHAPTER 19. PATCH CONSOLE COMMANDS

The patch commands allow you to download, install, and manage patches.

Patches contain a discreet set of bundles intended to update a standalone container. Each patch includes the following metadata:

- the patch name
- a description of the patch
- the list of bundles included in the patch

The basic procedure applying a patch is:

1. You receive a notice from customer support that a patch is available.
2. Using the URL provided by customer support, you download the patch using the `patch:add` command.

This command downloads an archive file, unzips the archive, and puts the relevant JAR files under the container's `system/` directory. The patch does *not* overwrite any of the existing JAR files and the patch is not actually installed until you run the `patch:install` command.

3. You install the patch using the `patch:install` command.
4. If you notice that the patch is causing issues, you can remove it using the `patch:rollback` command.

**IMPORTANT**

These commands are *not* suitable for use with containers that are part of a fabric. They are *only* for use in applying patches to standalone containers.

Type `patch:` then press `Tab` at the prompt to view the available commands.

NAME

`patch:add`, download – download a patch file from a remote location and places the relevant JAR files in the container's system directory

SYNOPSIS

```
patch:add [
  --help
] [
  --bundles
] {
  URL
}
```

ARGUMENTS

[Table 19.1, “patch:add Arguments”](#) describes the command's arguments.

Table 19.1. `patch:add` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>--bundles</code>	List the bundles included in the patch.
<code>URL</code>	Specifies the URL from which the patch is downloaded.

NAME

`patch:install` – installs a patch that was previously downloaded

SYNOPSIS

```
patch:install [
  --help
```

```

] {
  patch
}

```

ARGUMENTS

[Table 19.2, “patch:install Arguments”](#) describes the command's arguments.

Table 19.2. patch:install Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>patch</code>	Specifies the name of the patch to install.

NAME

`patch:list` – lists all known patches, showing the patch name and status (installed or not)

SYNOPSIS

```

patch:list [
  --help
] [
  --bundles
]

```

ARGUMENTS

[Table 19.3, “patch:list Arguments”](#) describes the command's arguments.

Table 19.3. patch:list Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>--bundles</code>	List the bundles for each patch.

NAME

`patch:rollback` – reverses a patch installation

SYNOPSIS

```
patch:rollback [
    --help
] {
    patch
}
```

ARGUMENTS

[Table 19.4, “patch:rollback Arguments”](#) describes the command's arguments.

Table 19.4. patch:rollback Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>patch</code>	Specifies the name of the patch to roll back.

NAME

`patch:simulate`, `simulate` – logs all of the actions that would be performed during a patch install, without actually performing the install

SYNOPSIS

```
patch:simulate [
    --help
] {
    patch
}
```

ARGUMENTS

[Table 19.5, “patch:simulate Arguments”](#) describes the command's arguments.

Table 19.5. patch:simulate Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command.
<code>patch</code>	Specifies the name of the patch to simulate installing.

CHAPTER 20. SSH CONSOLE COMMANDS

The `ssh` commands allow you to connect to or create a secure shell (SSH) server.

Type `ssh:` then press `Tab` at the prompt to view the available commands.

NAME

`ssh:ssh`, `ssh` – connects to a remote SSH server

SYNOPSIS

```
ssh:ssh [
  --help
] [[
  -l username
] | [
  --username username
]] [[
  -P password
] | [
  --password password
]] [[
  -p port
] | [
  --port port
]] {
  hostname
} [
  command
]
```

ARGUMENTS

[Table 20.1, “ssh:ssh Arguments”](#) describes the commands arguments.

Table 20.1. `ssh:ssh` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-l</code> , <code>--username</code>	The username for remote login
<code>-P</code> , <code>--password</code>	The password for remote login

Argument	Interpretation
-p, --port	The port to use for the SSH connection
<i>hostname</i>	The hostname to connect to via SSH
<i>command</i>	Specifies a command to execute upon connecting.

NAME

ssh:sshd, sshd – creates an SSH server

SYNOPSIS

```
ssh:sshd [
  --help
] [[
  -b
] | [
  --background
]] [[
  -p port
] | [
  --port port
]]
```

ARGUMENTS

[Table 20.2, “ssh:sshd Arguments”](#) describes the commands arguments.

Table 20.2. ssh:sshd Arguments

Argument	Interpretation
--help	Displays the online help for this command
-b, --background	Specifies that the service will run in the background.
-p, --port	Specifies the port to setup for the SSH server. The default is 8101 .

CHAPTER 21. WEB CONSOLE COMMANDS

The `web` command group is used to get information about WARs deployed in the container.

Type `web:` then press `Tab` at the prompt to view the commands in this group.

NAME

`web:list` – lists the WARs deployed in the container

SYNOPSIS

```
web:list [
    --help
]
```

ARGUMENTS

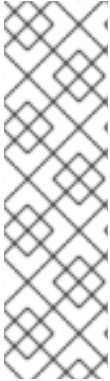
[Table 21.1, “web:list Arguments”](#) describes the command's arguments.

Table 21.1. `web:list` Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command

CHAPTER 22. THE WRAPPER:INSTALL COMMAND

The `wrapper:install` command installs Red Hat JBoss Fuse as a system service.

**NOTE**

This feature is not installed by default. To install the wrapper shell, run the following command:

```
JBossFuse:karaf@root:> features:install wrapper
```

NAME

`wrapper:install` – installs the container as a system service in the operating system

SYNOPSIS

```
wrapper:install [
  --help
] [[
  -s
] | [
  --start-type
] mode] [[
  -n
] | [
  --name
] serviceName] [[
  -d
] | [
  --display
] displayName] [[
  -D
] | [
  --description
] description]
```

ARGUMENTS

This command takes the following arguments.

Table 22.1. `wrapper:install` Arguments

Argument	Interpretation
--help	Displays the online help for this command
-s, --start-type	The mode in which the service is installed, either AUTO_START or DEMAND_START ; the default is AUTO_START
-n, --name	The service name used when installing the service; the default is karaf
-d, --display	The display name of the service
-D, --description	The description of the service

CHAPTER 23. ZOOKEEPER CONSOLE COMMANDS

By default, the ZooKeeper commands are not installed in a Fabric Container. To make the ZooKeeper commands available, install the `fabric-zookeeper-commands` feature, as follows:

```
features:install fabric-zookeeper-commands
```

NAME

`zk:create` – create a znode

SYNOPSIS

```
zk:create [
  --help
] [
  -r|--recursive
] [
  -i|--import
] [
  -e|--ephemeral
] [
  -s|--sequential
] [
  -a|--acl ListOfACLs
] [
  -o|--overwrite
] {
  path
} {
  data
}
```

DESCRIPTION

Using this command, you can create the following different types of znode:

Persistent

The new znode is permanently stored in the ZooKeeper registry. This is the default.

Persistent sequential

The new znode is permanently stored in the ZooKeeper registry and a 10-digit sequence number is appended to the specified znode name.

Selected by the `--sequential` option.

Ephemeral

The new znode exists only for the duration of the current client session. When the session is over, the znode is removed. Selected by the `-ephemeral` option.

Ephemeral sequential

The new znode exists only for the duration of the current client session and a 10-digit sequence number is appended to the specified znode name. When the session is over, the znode is removed. Selected by combining the `--ephemeral` option with the `--sequential` option.

You can optionally specify a list of ACLs to apply to the newly created znode. The ACL is specified as a comma-separated list, where each entry in the list has the following format:

| `Scheme:ID:Permissions`

ZooKeeper supports the following built-in schemes:

`world:anyone`

The permissions apply to all users.

`auth:`

The permissions apply to all authenticated users, irrespective of their identity (the *ID* field is left empty).

`digest:MD5Hash`

The permissions apply to the user whose username and password generate the specified MD5 hash value, *MD5Hash*.

`ip:IPAddress`

The permissions apply to the ZooKeeper client with the specified IP address.

The *Permissions* string consists of one or more of the following characters: *r* (read), *w* (write), *c* (create), *d* (delete), and *a* (admin). For example, to create a new znode that explicitly grants all permissions to all users (which is, in fact, the default), you could use a command like the following:

```
karaf@root> zk:create --acl world:anyone:rwcd /path/to/the/new/znode
```



IMPORTANT

To avoid corruption of the fabric registry, you should *not* create any znodes under the `/fabric/` path using the `zk:create` command. These registry nodes should only be created through the fabric console commands—see [Chapter 10, Fabric Console Commands](#).



NOTE

Fuse Fabric does *not* use the ACL security features of ZooKeeper. Currently, all znodes in the fabric registry are created without any ACL restrictions (equivalent to the `world:anyone:rwcd` ACL setting).

ARGUMENTS

[Table 23.1, “zk:create Arguments”](#) describes the commands arguments.

Table 23.1. zk:create Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-r, --recursive</code>	Automatically create any missing parent nodes in the specified path.
<code>-i, --import</code>	Interpret the data argument as a URL that locates a resource containing the initial data for the new znode.

Argument	Interpretation
-e, --ephemeral	Make the new znode ephemeral, so that it is automatically deleted after the current ZooKeeper client session closes.
-s, --sequential	Make the new znode sequential, which implies that a unique 10-digit suffix is appended to the znode name.
-a, --acl	Specifies the znode's ACL as a comma-separated list, where each entry in the list has the format, Scheme:ID:Permissions . The Permissions string consists of the following characters, concatenated in any order: r (read), w (write), c (create), d (delete), and a (admin).
-o, --overwrite	Overwrite the existing znode at this location, if there is one.
<i>path</i>	<i>(Required)</i> Path of the znode to create.
<i>data</i>	Initial data for the node or, if --import is specified, a URL pointing at a location that contains the initial data.

NAME

zk:delete – delete the specified znode

SYNOPSIS

```
zk:delete [
  --help
] [
  -v|--version version
] [
  -r|--recursive
] {
  path
}
```

ARGUMENTS

[Table 23.2, “zk:delete Arguments”](#) describes the commands arguments.

Table 23.2. zk:delete Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-v, --version</code>	The ZooKeeper znode version to delete. Defaults to -1 (all versions).
<code>-r, --recursive</code>	Recursively delete children. Defaults to false .
<i>path</i>	Path of the znode to delete.

NAME

`zk:get` – get a znode's data

SYNOPSIS

```
zk:get [
    --help
] {
    path
}
```

ARGUMENTS

[Table 23.3, “zk:get Arguments”](#) describes the commands arguments.

Table 23.3. zk:get Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<i>path</i>	<i>(Required)</i> Path of the znode to get.

NAME

`zk:list` – list a znode's children

SYNOPSIS

```
zk:list [
    --help
] [
    -r|--recursive
] [
```

```

    -d|--display
  ] {
    path
  }

```

ARGUMENTS

Table 23.4, “zk:list Arguments” describes the commands arguments.

Table 23.4. zk:list Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-r, --recursive</code>	List children recursively.
<code>-d, --display</code>	Display a znode's value, if set.
<code>path</code>	Path of the znode to list. Defaults to <code>/</code> .

NAME

zk:set – set a znode's data

SYNOPSIS

```

zk:set [
  --help
] [
  -i|--import
] {
  path
} {
  data
}

```

DESCRIPTION

The data stored in a znode should not be too large. ZooKeeper imposes an absolute limit of 1 MB, but in practice a data item should normally be much smaller than that.

**IMPORTANT**

To avoid corruption of the Fabric Registry, you should *not* modify any znodes under the `/fabric/` path using the `zk:set` command. These registry values should only be changed through the fabric console commands—see [Chapter 10, Fabric Console Commands](#).

ARGUMENTS

[Table 23.5, “zk:set Arguments”](#) describes the commands arguments.

Table 23.5. zk:set Arguments

Argument	Interpretation
<code>--help</code>	Displays the online help for this command
<code>-i, --import</code>	Import data from a URL.
<i>path</i>	<i>(Required)</i> Path of the znode to set.
<i>data</i>	<i>(Required)</i> The new data or URL to import.