DigitalRoute

Command Line Tool User's Guide

Table of Contents

Command Line Tool User's Guide

This document describes the Command Line Tool mzsh. The Command Line Tool is a standard user interface for the MediationZone Platform.

The Command Line Tool `mzsh` is a shell that can be used both as a system administration tool or as a Platform client tool. Depending on whether the Platform is running or not, or if you are logged in or not, `mzsh` enables you to access different parts of the system.

All `mzsh` commands run in their own JVMs, and the number of simultaneously running commands is only limited by the amount of memory available. If the system expects many commands to be running at the same time it is recommended to run them on a separate machine. This is usually an issue that must be considered when using the options related to 2.2.51 wfgroupstart and 2.2.59 wfstart.

Note!

To start an application, or to activate a workflow, you must have *Execute* permissions for the application and configuration. For further information, see the Desktop User's Guide.

Some commands require that the user is the MZ_HOME owner. This information will be found together with the respective commands presented later on in this user guide.

Note!

Make sure that the firewall is correctly configured as some commands require communication between the Execution Contexts and the Platform. For information about communication through firewalls, see the System Administration Guide.

1. Starting mzsh

The `mzsh` is used either interactively or non-interactively.

Interactively

In the interactive mode, you start and use the `mzsh` available commands as you would with any other shell. The shell will prompt you to give additional input or arguments if it is required.

To start mzsh interactively enter:
`$ mzsh`

If the platform is running, you will be prompted to enter username and password.

Depending on if you are logged in or not, the prompt will now appear as follows:

`MZ >`	Not logged in
`MZ >>`	Logged in

The date/time of the last successful login and the IP address that was used are displayed on login.

To log into a specific configuration space, you can specify the space name in the command line:

```
$ mzsh mzadmin/<password> @<space name>
```

If you have not created any spaces, when you login, you automatically log into the active space, which is the default space.

For further information about managing configuration spaces, see the Configuration Spaces documentation.

Non-interactively

In non-interactive mode, the command and its arguments are typed in the same invoking command line, in the Unix command prompt.

```
$ mzsh mzadmin/<password> wfstart MyWF
```

Enter username and password, in a single parameter together with the preferred command. When the command is executed, the command prompt is returned to the Unix shell.

mzsh can also read commands from the standard input. This means that commands that are meant to be executed can be included in a script file or simply let other commands produce the commands for it, see the following example.

Example - Non-interactive mode

```
$ while read wf ; do
echo wfstart $wf
done < my-wfs.list | mzsh mzadmin/<password>
```

The command in the example reads the names of the workflows included in the file `my-wfs.list` and generates start commands for each of them. The output of this action is then fed from standard input to `mzsh`.

To execute a command in a specific configuration space in non-interactive mode, you can specify the space name in the command line:

```
$ mzsh mzadmin/<password> @<name of space> <command>
```

For further information about managing configuration spaces, see the Configuration Spaces documentation.

2. Commands

The `mzsh` can be used both as a system administration tool and as a system client tool.

When using `mzsh` without running the platform, `mzsh` enables you to startup or shutdown the system and perform other tasks that are not dependent on having a Platform running.

The Platform server process is started by entering the following in the shell:

```
MZ> startup platform
```

The following text will appear:

```
Starting platform...done.
```

If `mzsh` is entered while the platform is running, `mzsh` will become a pico instance, with the ability to manage the system processes. For further information about Pico Clients, see the Desktop User's Guide.

Using commands downloaded from Platform requires that the user is logged in. For further information about the Platform, see the System Administrator's Guide. If `mzsh` is started in interactive mode, the system will prompt that username and password be entered:

```
$ mzsh
Username: mzadmin
Password: ******
```

or:

```
$ mzsh mzadmin/<password> startup platform
```

For example, if the platform has already been started, the following command:

```
$ mzsh startup platform
```

Will generate the following message:

```
Platform is already running.
```

You can still use `mzsh` in either interactive or non-interactive mode as described earlier.

> **Note!**
>
> Some commands only available when the MediationZone Platform is running.

2.1 Always Available

Command line tool commands that are available even when the Platform is not running, vary both in argument requirements and in behavior, depending on the logged on user's user permissions.

This section includes a detailed description of each command, its required or optional arguments, and its operation.

The following commands are available:

For information about the commands that or only available when the Platform is running, see 2.2 Available When the Platform Is Running.

2.1.1 help

```
usage: help [command]
```

If `help` is used without an argument, a list containing available commands and their required arguments will be displayed.

> **Note!**
>
> If the user is logged in a list of available commands with short descriptions is displayed.

When an argument is added to the `help` command, information about the entered command is displayed.

Return Codes

Listed below are the different return codes for the `help` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the command supplied in the argument does not exit.

2.1.2 desktop

```
usage: desktop
```

This command starts the Desktop Launcher.

Return Codes

Listed below are the different return codes for the `desktop` command:

Code	Description
0	Always returns 0.

2.1.3 echo

```
usage: echo [ -n ] <argument>
```

Generates added arguments.

Options

The command accepts the following option:

Option	Description
[-n]	Output without newline.

Return Codes

Listed below are the different return codes for the `echo` command:

Code	Description
0	Always returns 0.

2.1.4 encryptpassword

```
usage: encryptpassword [[<password>] | [-a|-alias <alias> [<password> | -e <encryptedpassword>]]]
```

Encrypts a password and prints out the result. Use this command to create an encryption of a password.

When you run encryptpassword in non-interactive mode, special shell characters must be escaped or the password may become truncated. You can use backslash (\) to escape a special character.

Example - Escaping special character

```
$ mzsh encryptpassword example\$password
```

You can escape all special characters in a string by surrounding it with single quotes (').

Example - Escaping all characters in a string

```
$ mzsh encryptpassword '`examplepassword!#$&()|\";'<> '
```

If single quote characters are part of the password, these can be escaped with backslash (\).

Example - Escaping single quote characters

```
$ mzsh encryptpassword '&example#'\'password
```

Hint!

You can use the encrypted password as a password value in an external reference or in a password cell of a workflow table.

Options

The command accepts the following options:

Option	Description
`[<password>]`	The password you want to encrypt.
`[-a|-alias]`	Use this option to encrypt a password with an alias. **Note!** In order to use this option, an alias must have been generated with the Java keytool. When this option is not used, the system default key will be used.If you want to use this option, the path and password to the keystore has to be indicated by setting the Platform properties `mz.cryptoservice.keystore.path` and `mz.cryptoservice.keystore.password`. The keystore must also contain keys for all the aliases you want to use. For further information about these properties, see 2.6.4 Platform Properties in the System Administrator's Guide.
`[-e]`	Use this option to encrypt a password with another alias.

Note!

The Platform has to be started, and you have to log in to be able to use the `-a` and `-e` options.

Return Codes

Listed below are the different return codes for the `encryptpassword` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the argument count is incorrect.
3	Will be returned if the encryption went wrong.

2.1.5 exit

```
usage: exit [ N ]
```

Exits the command line tool.

The argument N is the final exit code returned to the calling process. If no argument is supplied, the result of the previous operation will be returned to the calling process.

Return Codes

Listed below are the different return codes for the `exit` command:

Code	Description
N	The value of the supplied argument or the final exit code from the last command.

2.1.6 kill

```
usage: kill [ -l ]
```

Ends the pico instances(s) stated after the kill command. If the `-l` option is used, a list will be displayed instead.

> **Note!**
>
> This command is valid only for the MZ_HOME owner.

Options

The command accepts the following option:

Option	Description
[-l]	Lists all running server processes.

Use this command only in operating systems that provide you with a reference to the started JVM, pid on Unix. This will cause an unconditional immediate termination of the running process, with no clean-up nor save operations.

Return Codes

Listed below are the different return codes for the `kill` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if there are not startables to kill, or if a process you want to kill is running but the pid file is missing, or if the you have no read permission for the pid file belonging to the process you want to kill.
2	Will be returned if the process you want to kill is not running, or if there is no such process available to kill.

2.1.7 pcreate

```
usage: pcreate <name> <version> <package-file> [ -level <default level> ] [-revision <revision>] [-repository
<repository>]
[-metadata <data>] [-hidden] [[-level <level name>] file=<file-to-include>] [-osgi <true/false> ] [-
exportpackages <classpath>]... ]
```

Creates a software package (`.mzp`) that can be installed into the MediationZone system. It is usually used by software developers to create additional functionality and updates.

Arguments

Argument	Description
<name>	The name of the package.
<version>	The version string of the package name.
<package-file>	The resulting package.

For further information, see the Development Toolkit User's Guide.

Return Codes

Listed below are the different return codes for the `pcreate` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the argument count was incorrect.
2	Will be returned if the package name is too long.
3	Will be returned if if the package version is too long.
4	Will be returned if an install class or a jar argument is missing.
5	Will be returned if a meta data argument is missing, or if a revision argument is missing.
6	Will be returned if a repository argument is missing, or if a level argument is missing.
7	Will be returned if the level is invalid.
8	Will be returned if a file argument is missing
9	Will be returned if no level is given for the jar file.
10	Will be returned if a file argument is missing.
11	Will be returned if the file cannot be read.
12	Will be returned if the package filename cannot be found.
13	Will be returned if the install jar file cannot be read.

2.1.8 picoversion

```
usage: picoversion
```

Displays the version information from the `picostart.jar` file.

```
MZ >> picoversion
8.0.0.0 [compatibility: 7.0.4.0]
```

Return Codes

Listed below are the different return codes for the `picoversion` command:

Code	Description
0	Will be returned if the command was successful, which it always is.

2.1.9 quit

```
usage: quit [ N ]
```

Exits the command line tool.

The argument N is the final exit code returned to the calling process. If no argument is supplied, the result of the previous operation will be returned to the calling process.

Return Codes

Listed below are the different return codes for the `quit` command:

Code	Description
N	The value of the supplied argument or the final exit code from the last command.

2.1.10 restart

```
usage: restart [ -q ] [ -f ] <running server process> ...
```

This command is used to stop and start pico instances.

> **Note!**
>
> This command is valid only for the MZ_HOME owner.

If you have specified more than one pico instance, these will be shut down in the reverse order of the arguments. Once the pico instances have stopped, the pico instances will be started in the same order as the arguments.

> **Example - Restarting Pico Instances**
>
> The following command will close down the pico instance named ec1 and then the Platform. When both processes have stopped, the platform will start the Platform followed by ec1.
>
> ```
> MZ>> restart platform ec1
> ```
>
> The command will fail if the processes are specified in the reverse order, since it would attempt to start ec1 before the Platform.

> **Note!**
>
> The command will behave differently depending on if the user is logged in or not. If a user is *not* logged in when the `shutdown` command is executed, the immediate, forced shutdown will be used regardless of if the `-f` option is used or not.

For further details, see the descriptions for the `shutdown` and `startup` commands in 2.1.11 shutdown and 2.1.12 startup.

Options

The command accepts the following options:

Option	Description
[-q]	Generates less or no information messages.
[-f]	Forced restart of the pico instance(s). If the pico instance is an EC/ECSA, it will be shut down without waiting for workflows to stop first. This option has no impact on the Platform.

Return Codes

Listed below are the different return codes for the `restart` command:

Code	Description
0	Will be returned if no arguments are given to the command, at successful restart, or if there are no matching server processes to restart.
1->	The sum of the return codes from shutdown and startup.
104	Will be returned if the Java Virtual Machine (JVM) fails to start.

2.1.11 shutdown

```
usage: shutdown [ -q ] [ -f ] <running server process> ...
```

This command is used to stop pico instances.

> **Note!**
>
> This command is valid only for the MZ_HOME owner.

If you have specified more than one pico instance, the command will close down the processes in the order specified by the arguments.

If the pico instance is an EC/ECSA, the command will first close down running workflows and then it shuts down the specified running server processes.

Each pico instance will be shut down in an orderly fashion, that is, it will do the required cleanup before terminating. The last thing that the pico instance does is to report back to `mzsh` that it is about to terminate. When this is done `mzsh` will continue with the next pico instance.

In rare occasions the shutdown command is not forceful enough. In these cases, the kill command can work. For further information, see 2.1.6 kill.

Options

The command accepts the following option:

Option	Description
[-q]	Generates less or no information messages.
[-r<#>]	Number of retries to check for process shutdown. Default: -r20
[-f]	Forces workflow stop when shutting down the EC/ECSA regardless of system configuration This is the default behavior, When the Execution Context property `pico.ec.do_graceful_shutdown` has been set to `false`, you can use this option for graceful shutdown of EC/ECSAs.

Return Codes

Listed below are the different return codes for the `shutdown` command:

Code	Description
0	Will be returned if the command was successful, or if the pico instances are not running.
1	Will be returned if no pico instance can be found
130	Will be returned if the shutdown is interrupted
4	Will be returned if the shutdown fails

2.1.12 startup

```
usage: startup [ -e <property=value> ] [ -f ] [ -q ] <server process>...
```

This command is used to start pico instances.

> **Note!**
>
> - This command is valid only for the MZ_HOME owner.
> - The Platform must be started before any other pico instance. If there is a problem during startup an error message will be shown and more information will be visible in the Platform log.

This command will not terminate until a pico instance has reported back that it is up and running. If you have specified more than one pico instance in the arguments, it will not continue with the next one until the previous instance has reported that it is up and running.

Options

The command accepts the following option:

Option	Description
[-e <property=value>]	Use this option to set or override a system property. This is useful for test purposes but not recommended for production environments. **Example. Setting properties in the mzsh startup command.** `$ mzsh startup platform ec1 -e pico.log.level="FINEST"`
[-f]	Use this option to force start of an existing pico instance that is unreachable. **Note!** When you want to start a pico instance that is already started but unreachable, you must first delete it or use the `-f` option. For further information, see 6.10 Pico Viewer in the Desktop User's Guide.
[-q]	Use this option to generates less or no messages in the output.

Return Codes

Listed below are the different return codes for the `startup` command:

Code	Description
-1	Will be returned if there is an old pico instance running or if the remote (../temp/.remote) file cannot be deleted.
0	Will be returned if the command was successful, or if the pico instances are already started.
1	Will be returned if the JVM failed to start. (The JVM has logged too much on stderr.)
102	Will be returned if the JVM failed to start. (The timeout on the callback from the JVM was exceeded.)
130	Will be returned if the command has been interrupted with CTRL-C (for Macbook).
104	Will be returned if the JVM failed to start. (The JVM started with (a) critical error(s).)

2.1.13 status

```
usage: status [ -q ] [ -verbose ] <server process> ...
```

This command is used to retrieve the running status of pico instances that are defined in the local container.

The states are:

- running
- not running
- not responding but the process is still running. (Shown if the local server process is not responding.)
- running without contact with the platform RCP service. (Shown if information can not be collected from the platform.)

Note!

If the code server (a server closely connected to the platform with the ability to function independently) is not available the `status` command will wait for the RCP communication to time out, this requires some extra time.

Options

The command accepts the following options:

Option	Description
[-q]	Generates less or no information messages.
[-verbose]	Enables verbose mode.

Return Codes

Listed below are the different return codes for the `status` command:

Code	Description
0	Will be returned if a command was successful and all defined processes are running, or if there are picos defined in the system.
1	Will be returned if any of the processes is not responding but its JVM is running.
2	Will be returned if there is no server process with the specified name, or if any of the defined processes are not running.

2.1.14 system

```
usage: system <subcommand> <options>
```

This command starts the Platform if it is not running, and then starts/stops pico instances on containers that have been set up for remote execution.

Note!

This command is valid only for the MZ_HOME owner.

You can specify which pico instances that are to be started, stopped, or restarted by adding a target path to the subcommands. The target path is specified as follows:

```
container:<container>/pico:<pico>
```

or

```
container:<container>
```

Note!

Set your desired environment variables in $MZ_HOME/bin/mzshr.env as this file will be loaded with local variables. Add the desired profiles, such as ". /home/mzadmin/.profile_mz".

You can specify both the container and pico instance as a regular expression.

Example - Regular expression in target paths

```
container:.*/pico:.*
```

By adding tag attributes you can perform additional filtering of the pico instances:

Example - Adding tags to pico instances

```
$ mzsh topo set -l pico:ec1 'settings.tags=[tag1,tag2]'
$ mzsh topo set -l pico:ec2 'settings.tags=[tag1]'
```

Run the following command to start the pico instances with the tag `tag1`.

```
$ mzsh system start -t tag1
```

The following subcommands are available with `mzsh system`:

- `help`
- `restart`
- `start`
- `status`
- `stop`

help

```
Usage: system help [<subcommand>]
```

Use `system help` to retrieve a description of the help command or its subcommands.

Run the following command for an overview of the various subcommands:

```
$ mzsh system help
```

Run the following command for a description of a specific subcommand:

```
$ mzsh system help <command>
```

restart

```
Usage: system restart [--dry-run] [-services] [-t, --tag <tag>] [--timeout-seconds] [-v, --verbose] [<target
path>]
```

Use `system restart` to stop and start pico instances in one or more containers. The Platform will be started if it is not already running. However, the command does not stop the Platform.

Option	Description
[--dry-run]	Lists the picos instances that are addressed by the command, but the command is not executed.
[-t, --tag <tag>]	Filter that excludes all pico instance that do not contain the specified tag.
[--timeout-seconds]	Sets the maximum allowed time for all calls to complete. The default value is 300 seconds.
[-v, --verbose]	Use this option for detailed output from the command.

start

```
Usage: system start [--dry-run] [-t, --tag <tag>] [--timeout-seconds] [-v, --verbose]
```

Use `system start` to start pico instances in one or more containers. The Platform will be started if it is not already running.

Option	Description
[--dry-run]	Lists the picos instances that are addressed by the command, but the command is not executed.
[-t, --tag <tag>]	Includes pico instances in the target path that contain the specified tag.
[--timeout-seconds]	Sets the maximum allowed time for all calls to complete. The default value is 300 seconds.
[-v, --verbose]	Use this option for detailed output from the command.

status

```
Usage: system status [--dry-run] [-t, --tag <tag>] [--timeout-seconds] [-v, --verbose] [<target path>]
```

Use `system status` to retrieve the running status of pico instances in one or more containers. The Platform will be started if it is not already running.

Option	Description
[--dry-run]	Lists the picos instances that are addressed by the command, but the command is not executed.
[-t, --tag <tag>]	Filter that excludes all pico instance that do not contain the specified tag.
[--timeout-seconds]	Sets the maximum allowed time for all calls to complete. The default value is 300 seconds.
[-v, --verbose]	Use this option for detailed output from the command.

stop

```
Usage: system stop [--dry-run] [-t, --tag <tag>] [--timeout-seconds] [-v, --verbose] [<target path>]
```

Use `system start` to start pico instances in one or more containers. The Platform will be started if it is not already running.

Option	Description
[--dry-run]	Lists the picos instances that are addressed by the command, but the command is not executed.
-t, --tag <tag>]	Includes pico instances in the target path that contain the specified tag.
[--timeout-seconds]	Sets the maximum allowed time for all calls to complete. The default value is 300 seconds.
[-v, --verbose]	Use this option for detailed output from the command.

Return Codes

Listed below are the different return codes for the `system` command:

Code	Description
-1	Will be returned if there is an old process running or if the remote (../temp/.remote) file cannot be deleted.
0	Will be returned if the command was successful, or if there are not startable processes defined.
1	Will be returned if the JVM failed to start. (The JVM has logged too much on stderr.)
102	Will be returned if the JVM failed to start. (The timeout on the callback from the JVM was exceeded.)
103	Will be returned if the command has been interrupted with CTRL-C.
104	Will be returned if the JVM failed to start. (The JVM started with (a) critical error(s).)

2.1.15 topo

```
usage: topo <subcommand> <options>
```

This command is used to register containers in STR and to create, update, remove, and view pico configurations.

> **Note!**
>
> This command is valid only for the MZ_HOME owner.

When you make changes in pico configurations, using `topo`, these are automatically validated before they are copied to the active registry.

If the command and its arguments can be parsed but fails the validation, you can update the configuration or use a reset command to undo the changes. An error message will appear if the validation fails. You can disable the validation by using the option `--no-activation`. Changes performed by the `mzsh topo` will then remain in the master registry until you submit a separate `topo activate` command.

You can use the following subcommands with topo:

- `activate`
- `container`
- `convert`
- `diff`
- `env`
- `get`
- `hash`
- `help`
- `migrate`
- `open`
- `rebase-configs`
- `register`
- `reset`
- `set`
- `setupremote`
- `show`
- `unset`

The option `--allow-disconnected` is available for all subcommands except for `setupremote`. You should only include this option when the Platform is unreachable and you want to use cached data.

activate

```
Usage: topo activate [--dry-run] [-v, --verbose]
```

Use `topo activate` to move staged changes in the master registry to the active registry.

Option	Description
[--dry-run]	Use this option to validate the staged changes without performing the activation.
[–hash <hash value>]	Compare the provided hash value with the actual hash that represents the current state of active registry. The activation fails if the values are not equal. For further information, see hash below,
[-v, --verbose]	Use this option for detailed information about the changes.

The options `--dry-run` and `--verbose` are useful to learn the `mzsh topo` syntax. When you have edited the configuration manually, use the following command, to view the corresponding edits in a scripted syntax:

```
$ mzsh topo activate --dry-run --verbose
```

Example - Output from activate with verbose option

```
$ mzsh topo activate -v --dry-run
mzsh topo set topo://container:main1/pico:ec1/val:config.properties.ec.httpd.port 9096  #
(was: 9092)
Dry-run: Validation successful
Dry-run: Stopping without performing activation
Dry-run: Active registry not changed
```

You can then restore the master registry with the the command `mzsh topo reset`.

Restart the picos to apply the changes

Note!

Changes to the STR are not applied on running pico instances or services. For instance, if you have updated the properties of the Platform and an EC, both must be restarted after activation.

Example, after an mzsh topo activate of ec5, mzsh shutdown and startup needs to be done to apply the changes.

```
$ mzsh shutdown ec5
$ mzsh startup ec5
```

container

```
Usage: topo container
```

Use `topo container` to display the name of the current container.

Option	Description
[--allow-disconnected]	Use this option when the Platform is unreachable and you want to operate on cached data.

convert

```
Usage: topo convert [-c, --container <container>] [-g, --container-group <container group>] [--dry-run] [-f, --file <filename>]
```

Use `topo convert` to move the configuration of a specific XML file to STR.

Option	Description
`[-c, --container <container>]`	Use this option to specify a target container.
`[-g, --container-group <container group>]`	Use this option to specify a target container group.
`[--dry-run]`	Use this option to validate that the conversion and display the result of the conversion without updating the STR.
`[-f, --file <filename>`	Use this option to specify the source XML file.

Example - Converting an XML file

```
$ mzsh topo convert --container main1
```

diff

```
Usage: topo diff [-e, --show-entries] [-f, --from <registry>] [-q, --brief]
```

Use `topo diff` to view differences between the master repository and the active repository in the STR.

Option	Description

[-e, --show-entries]	Use this option for viewing differences in an easy-to-read format. By default, the output from the command displays `topo set` commands that correspond to the staged changes.
	
	Example - Output from diff command
	
	With -e option:
	
	`UPDATE (containers/main1/picos/ec1.conf) config.properties.aaa:"2" # (was: "1")`
	
	Without -e option:
	
	`mzsh topo set topo://container:main1/pico:ec1/val:config.properties.aaa "2" # (was: "1")`

[-f, --from] <registry>	Use this option when you want to compare the active registry with the backup registry
[-q, --brief]	Use this option to only view the names of the updated registry files. The default value is false.

Example - Comparing registry files

Run the following command to view the differences between the active registry and the master registry.

```
$ mzsh topo diff
```

or

```
$ mzsh topo diff --from master
```

Run the following command to view the differences between the active registry and the backup registry.

```
$ mzsh topo diff --from backup
```

env

```
Usage: topo env [-e, --effective] [--update-java-home <value>] [--update-mz-container <value>] [--update-mz-
home <value>]
[--update-mz-platform <value>]
```

Use `topo env` to display or set environment variables that are used my the mzsh command. These variables are written to the script file `MZ_HOME/bin/mzsh`.

Option	Description
[-e, --effective]	Use this option to read the environment parameters in runtime, i e the "effective values" after accounting for overrides. The default behaviour is to read the values as they are defineds in the mzsh script file, not accounting for the possibility to override these values with environment variables.
[--update-java-home <value>]	Use this option to update the value of JAVA_HOME
[--update-mz-container <value>]	Use this option to update the value of MZ_CONTAINER.
[--update-mz-home <value>]	Use this option to update the value of MZ_HOME.
[--update-mz-platform <value>]	Use this option to update the mzsh value of MZ_PLATFORM.

Example - Reading the environment variables

```
$ mzsh topo env
export JAVA_HOME="/opt/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home"
export MZ_CONTAINER="main1"
export MZ_CONTAINER_TYPE="platform"
export MZ_PLATFORM="http://localhost:9000"
export MZ_HOME="/user/home/mz/main1"
```

Example - Setting the environment variable JAVA_HOME

```
$ mzsh topo env --update-java-home /opt/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home
```

get

```
Usage: topo get [--default-val <value>] [ --exclude-dynamic] [--format <full | data-only>] [-l, --local] [-p,
--perspective] <target path>]
```

Use `topo get` to retrieve pico configurations in the target path from STR.

Paths in STR are structured as follows:

```
topo://container:<container>/pico:<pico>/val:<attribute>
```

Option	Description

[--default-val <value>]	Use this option to replace a missing value in the target path with a default value.
	
	Example - Using default-val
	
	If the property aaa, is not defined for ec1, 123 is returned instead.
	
	`$ mzsh topo get -l --default-val 123 \`
	`topo://pico:ec1/val:config.properties.aaa`

[--exclude-dynamic]	Use this option to exclude non-static data in the output e g `_status` in a pico configuration. This is useful in case of errors that blocks the topo command.
[--format <full|data-only>]	Use this option to exclude metadata from the command output. • full - Include meta data • data-only - exclude meta data Default: `full`
[-l, --local]	Use this option to select the local container, unless another container is specified in the target path. Default: `false`
[-p, --perspective <resolve | default>]	Use this option to retrieve the attributes of templates instead of the template names. • resolve - attributes • default - template names Default: `default`

Example - Viewing pico configurations

Run the following command to view one or more pico configurations.

```
$ mzsh topo get topo://container:main1/pico:ec2
```

You can view multiple pico configurations by replacing the full path with a regular expression.

```
$ mzsh topo get topo://container:main1/pico:.*
```

Example - Viewing pico attributes

Run the following command to view a specific attribute in a pico configuration.

```
$ mzsh topo get topo://container:main1/pico:ec2/val:_name
```

You can retrieve the attributes of multiple pico processes by replacing the full path with a regular expression.

```
$ mzsh topo get --format data-only topo://container:main1/pico:.*/val:_name
```

hash

```
Usage: topo hash
```

Use `topo hash` to retrieve a value that represents the current state of the active registry. This is useful when you need to handle concurrent changes of the STR. For instance, an application may need to retrieve a pico configuration to evaluate the required changes. In the meantime, a second application or a user may update the same configuration

Example - Using hash values

1. Application 1 retrieves a new hash value.

```
$ mzsh topo hash
"3a2e373fa1653c7f0e757e2682c70317-2028777631"
```

2. Application 1 retrieves the properties of ec1.

```
$ mzsh topo get -l pico:ec1/obj:config.properties
```

3. Application 1 updates a property but does not call `topo activate`. The hash is specified to ensure that changes by other users are not activated inadvertently later on.

```
$ mzsh topo set -l --no-activation --hash 3a2e373fa1653c7f0e757e2682c70317-2028777631 \
topo://pico:ec1/val:config.properties.ec.httpd.port 9090
```

4. Application 2 updates the properties of ec1. The hash value is updated.

```
$ mzsh topo set -l topo://pico:ec1/val:config.properties.ec.httpd.port 9090
```

5. Application 1 calls `topo activate` with hash value retrieved in step 1.

```
$ mzsh topo activate --hash 3a2e373fa1653c7f0e757e2682c70317-2028777631
```

6. The activation fails since the hash values do not match.

```
Specified hash does not match transaction id:  d9cd38f3793647028bd7e5d64c354ad5-2055434210 !=
3a2e373fa1653c7f0e757e2682c70317-2028777631)
This may indicate concurrent modification of registry: Operation aborted!
```

7. Application 1 resets the master registry, retrieves a new hash and starts over.

```
$ mzsh topo reset
$ mzsh topo hash
```

help

```
Usage: topo help [<subcommand>]
```

Use `topo help` to retrieve a description of a subcommand.

Run the following command for an overview of the various `topo` subcomands

```
$ mzsh topo help
```

Run the following command for a description of a specific subcomand

```
$ mzsh topo help <command>
```

migrate

```
Usage: topo migrate
```

Use `topo migrate` to move pico configurations from MZ_HOME to STR. The upgrader runs this command during upgrade.

open

```
Usage: topo open [-n, --no-activation] <target path>
```

Use `topo open` to open a cell, container- or pico configuration file in a text editor. When you save and close the editor, the command will call `topo activate` to move the staged changes in the master registry to the active registry.

Option	Description
[-n, --no-activation]	Use this option to skip activation after changes in master registry.

Run the following command to open a cell configuration:

```
$ mzsh topo open cell:<cell>
```

Example - Opening a cell configuration

```
$ mzsh topo open cell:default
```

Run the following command to open a container configuration:

```
$ mzsh topo open <container>
```

Example - Opening a container configuration

```
$ mzsh topo open main1
```

Run the following command to open a pico configuration:

```
$ mzsh topo open <pico>
```

Example - Opening a pico configuration

```
$ mzsh topo open ec1
```

or

```
$ mzsh topo open container:main1/pico:ec1
```

If the pico name is not unique in the system, you will be prompted to specify the container.

Example - Multiple pico configuration sharing the same name

```
$ mzsh topo open ec2 (/home/main1/common/config/cell/default/master/containers/main1/picos/ec2.conf,
ec2,topo://container:main1/pico:ec2)
(/home/main1/common/config/cell/default/master/containers/exec1/picos/ec2.conf,ec2,topo://container:
exec1/pico:ec2)

Multiple entries, select one:
 (1) topo://container:main1/pico:ec2
 (2) topo://container:exec1/pico:ec2
 [1] :
```

To avoid ambiguous references, specify the name of the container and the pico configuration.

Run the following command to open the custom or the standard services configuration:

```
$ mzsh topo open services:<custom|standard>
```

Example - Opening a service configuration

```
$ mzsh topo open services:custom
```

Hint!

When you save the configuration, `topo activate` is called with the `--verbose` option and the saved changes are displayed in a scripted syntax.

By default, the command opens the vi editor. To use a different editor set the environment variable `EDITOR`.

Example - Setting nano as the default editor

```
$ export EDITOR=nano
```

rebase-configs

```
Usage: topo rebase-configs [-a, --activate] <target path>
```

Use `topo rebase` to inset a standard template in a pico configuration and remove attributes that are identical to attributes in the template. The command automatically detects the pico configuration type and applies one of the following templates:

- `mz.standard-platform.conf`
- `mz.standard-ec.conf`
- `mz.standard-ecsa`
- `mz.standard-sc`

This command is useful to reduce the size of the pico configurations and thereby facilitate maintenance.

The changes are written to the master registry. To validate and activate the changes you can either use the --activate option or run `topo activate` after the `topo rebase-configs` command.

For further information about templates, see 1.4.2 STR File Structure in the System Administrator's Guide.

Option	Description
[-a, --activate]	Use this option to immediately activate after changes in master registry.

Example - Rebasing an EC configuration

```
$ mzsh topo rebase-configs topo://container:main1/pico:ec1$ mzsh topo active --verbose
```

or

```
mzsh topo rebase-configs --actoivate topo://container:main1/pico:ec1
```

register

```
Usage: topo register [-a, --address] [-c, --container] [-g, --container-group <container group>] [--mz-home
<mz home>] [-u]
```

When you install an execution container, and the Platform is running, it is automatically registered in the Platform Container. If the platform is not running during the installation, use `topo register` to register the Execution Container manually.

Option	Description
[-a, --address <ip /host>]	Use this when you need to set a different host address for the container than the one that is specified in the common property `pico.rcp.server.host`, which is the default value. This option is typically used together with the `-u` option.
[-c, --container <container>]	Use this option when you need to change the existing container name. This option is typically used together with the `-u` option.
[-g, --container-group <container group>]	Use this option when you need to change the existing container group. This option is typically used together with the `-u` option.
[--mz-home <path>]	Use this option when you need to set a different home directory for the container than the one that is specified in the environment variable MZ_HOME, which is the default value.
[-u]	Use this option to allow updates of an already registered container. By default, updates are not allowed and the command will attempt to register a new container.

reset

```
Usage: topo reset
```

Use this option to remove any changes to the master registry in STR since the activation

set

```
Usage: topo set [-l, --local] [-n, --no-activation] [-s, --strict-json] <target path> <config>
```

Use `topo set` to create and update pico configurations in the specified target-path of STR.

Option	Description
[-l, --local]	Use this option to select the local container, unless another container is specified in the target path.
[--no-activation, -n]	Use this option to skip activation after changes in master registry.
`[-s, --strict-json]`	Use this option when you want to specify the configuration in JSON format instead of HOCON format.

Run the following command to create a new pico configuration.

```
$ mzsh topo set topo://container:<container>/pico:<pico> <config>
```

The `<config>` argument may contain a key-value pair that specifies a template or a pico configuration in HOCON format.

Example - Creating a new pico configuration based on a template

```
 $ mzsh topo set topo://container:main1/pico:ec2 template:mz.standard-ec
```

Example. Creating pico configuration

When you specify a pico configuration that consists of multiple attributes, it is recommended that you use multi-line strings.

HOCON Format:

```
$ mzsh topo set --local pico:ec2 '
{
        template:mz.standard-ec
        config {
                properties {
                        ec.httpd.port : 9092
                }
                classpath {
                        jars=["lib/picostart.jar"]
                }
        }
}'
```

JSON Format:

```
mzsh topo set -l --strict-json pico:ec2 '
{
        "template": "mz.standard-ec",
        "config": {
                "properties": {
                        "ec": {
                                "httpd": {
                                        "port": 9092
                                }
                        }
                },
                "classpath": {
                        "jars": ["lib/picostart.jar"]
                }
        }
}'
```

Run the following command to add or update an attribute of a pico configuration.

```
mzsh topo set topo://container:<container>/pico:<pico>/val:<attribute> <attribute value>
```

Example - Updating a pico attribute

```
$ mzsh topo set topo://container:main1/pico:ec2/val:ec_type ecsa
```

Run the following command to add or update an object that contains one or more attributes.

```
$ mzsh topo set topo://container:<container>/pico:<pico>/obj:<object name> '<config>'
```

The `<config>` argument may contain a pico configuration in HOCON format.

Example - Updating a pico object

This command adds the properties `value1` and `value2`:

```
$ mzsh topo set topo://container:main1/pico:ec2/obj:config.properties.example_object '{
 value1=1
 value2=2
}'
```

The following commands does not overwrite the properties `value1` and `value2` in `example_object` but adds `value3`:

```
$ mzsh topo set topo://container:main1/pico:ec2/obj:config.properties.example_object '{
 value3=3
}'
```

setupremote

```
Usage: topo setupremote [-c, --container <container>] [-g, --container-group <container group>] [--host-key
<path>] [--javahome <path>] [--no-authorized-key] [--no-host-key] [-- no-ssh-details] [--ssh-address <ip
/host>] [--ssh-port <port>] [--ssh-username <username>]
```

Use the command `topo setupremote` to enable remote access via SSH to an Execution Container, e g from the Platform container.

Option	Description
[-c, --container <container>]	Use this option to specify a different container than the local one, which is the default value.
[-g, --container-group <container group>]	Use this option to setup remote access to a container in specific container group. This is useful when you have multiple containers with identical names in different containers groups.
[--host-key <path>]	Use this option to use a pre-generated host key instead of the one that is generated when you run `topo setupremote`.
[--java-home <path>]	Use this option when the target container is located on a different host. The default value is specified by the environment variable JAVA_HOME in the current shell.
[--no-authorized-key]	By default, the `topo setupremote` command will obtain a public authorization key from the user home directory on the Platform Container host and store it in the STR, i e the file `mz.conf`. Use the option `--no-authorized-key` to skip this operation.
[--no-host-key]	By default, the `topo setupremote` command will store the public host key of the Execution Container in the STR, i e the file `mz.conf`. Use the option `--no-host-key` to skip this operation.
[--no-ssh-details]	Use this option to exclude `ssh-username` and `ssh-address` from STR. These attributes are required for remote access. If you use this option you will need to update the STR manually.
[--ssh-address <ip /host>]	Use this option when the target container is located on a different host or when you want to bind to a specific IP address or hostname. The default value is specified by the `address` attribute for container in `mz.conf`.
[--ssh-port <port>]	Use this option when you want to use a different port than 22 for SSH.
[--ssh-username <username>]	Use this option when the target container is located on a different host or when a specific username is required for SSH. The default SSH user is the OS user that runs the `topo setupremote` command.

show

Use topo show to retrieve various types of information about pico instances that are defined in the STR.

```
Usage: topo show [ --exclude-dynamic] [--format <format>] [-l, --local] [--timeout-seconds <time>] <view>
```

Option	Description
[--exclude-dynamic]	Exclude non-static data in the output e g `_status` in a pico configuration. This is useful in case of errors that blocks the topo command.

[--format <format>]	Set the format of the returned data:
	• `csv`
	`json`
	`table (default)`

[-l, --local]	Use this option to view pico instances in the local container only. By default, all containers are included.
[--timeout-seconds <time>]	Use this option to limit the time for retrieving dynamic information, e g `_status`.The default value is 10 seconds.

The following views are available:

- `jvm-args` - Displays the JVM arguments that are used by the pico instances in the system. JVM arguments that are set in templates are included.
- `status` - Displays the container name, pico name, pico type and running state.
- `status-sc` - Displays similar view as `status` but only includes SCs.
- `status-ec` - Displays similar view as `status` but only includes EC/ECSAs.
- `status-long` - Displays similar view as `status` but also includes the status of replication between Platform Container and Execution Containers.
- `pico-view` - Displays similar view as `status` but also includes memory usage and the pico response time.
- `pico-view2` - Displays similar view as `pico-view` but also includes uptime.

- `ports` - Displays the ports that are used by the pico instances in the system. Ports that are set in templates and on cell- and container level, are included.

Example - Views

```
$ mzsh topo show jvm-args
+-----------------------------------------------------------
| container | name     | config.jvmargs                      |
+-----------+----------+-------------------------------------+
| main1     | platform | args=[                              |
|           |          |     "-XX:MaxMetaspaceSize=256M",    |
|           |          |     "-Xms192M",                     |
|           |          |     "-Xmx1024M"                     |
|           |          | ]                                   |
+-----------+----------+-------------------------------------+
| main1     | ecsa1    | args=[                              |
|           |          |     "-server"                       |
|           |          | ]                                   |
|           |          | maxDirect=[                         |
|           |          |     "-XX:MaxDirectMemorySize=4096M" |
|           |          | ]                                   |
|           |          | maxMetaspace=[                      |
|           |          |     "-XX:MaxMetaspaceSize=196M"     |
|           |          | ]                                   |
|           |          | xms=[                               |
|           |          |     "-Xms64M"                       |
|           |          | ]                                   |
|           |          | xmx=[                               |
|           |          |     "-Xmx256M"                      |
|           |          | ]                                   |
+-----------+----------+-------------------------------------+
| main1     | psc1     | args=[                              |
|           |          |     "-server"                       |
|           |          | ]                                   |
|           |          | maxDirect=[                         |
|           |          |     "-XX:MaxDirectMemorySize=4096M" |
|           |          | ]                                   |
|           |          | maxMetaspace=[                      |
|           |          |     "-XX:MaxMetaspaceSize=196M"     |
|           |          | ]                                   |
|           |          | xms=[                               |
|           |          |     "-Xms64M"                       |
|           |          | ]                                   |
|           |          | xmx=[                               |
|           |          |     "-Xmx256M"                      |
|           |          | ]                                   |
+-----------+----------+-------------------------------------+
| exec1     | ec2      | args=[                              |
|           |          |     "-server"                       |
|           |          | ]                                   |
|           |          | maxDirect=[                         |
|           |          |     "-XX:MaxDirectMemorySize=4096M" |
|           |          | ]                                   |
|           |          | maxMetaspace=[                      |
|           |          |     "-XX:MaxMetaspaceSize=196M"     |
|           |          | ]                                   |
|           |          | xms=[                               |
|           |          |     "-Xms64M"                       |
|           |          | ]                                   |
|           |          | xmx=[                               |
|           |          |     "-Xmx256M"                      |
|           |          | ]                                   |
+-----------------------------------------------------------
```

```
$ mzsh topo show status
+-------------------------------------------------------------
| container | name     | type     | state       | config-state |
+-----------+----------+----------+-------------+--------------+
| main1     | platform | platform | running     | in-sync      |
| main1     | ecsa1    | ecsa     | not-started |              |
| main1     | psc1     | sc       | not-started |              |
| exec1     | ec2      | ec       | not-started |              |
+-------------------------------------------------------------
```

```
$ mzsh topo show ports
+-------------------------------------------------------------------------------
| container | name     | type     | ports                                      |
+-----------+----------+----------+--------------------------------------------+
| main1     | platform | platform | "mz.pcc.restful.port"="9090"               |
|           |          |          | "mz.servicehost.port.range"="5451-5500"    |
|           |          |          | "mz.wi.port"="9000"                        |
|           |          |          | "pico.rcp.platform.port"="6790"            |
|           |          |          | "pico.synchronizer.port"="6791"            |
+-----------+----------+----------+--------------------------------------------+
| main1     | ecsa1    | ecsa     | "ec.httpd.port"="9093"                     |
|           |          |          | "pico.rcp.platform.port"="6790"            |
|           |          |          | "pico.synchronizer.port"="6791"            |
+-----------+----------+----------+--------------------------------------------+
| main1     | psc1     | sc       | "mz.servicehost.port.range"="5801-5850"    |
|           |          |          | "pico.rcp.platform.port"="6790"            |
|           |          |          | "pico.synchronizer.port"="6791"            |
+-----------+----------+----------+--------------------------------------------+
| exec1     | ec2      | ec       | "ec.httpd.port"="9090"                     |
|           |          |          | "pico.rcp.platform.port"="6790"            |
|           |          |          | "pico.synchronizer.port"="6791"            |
+-------------------------------------------------------------------------------
```

unset

```
Usage: topo unset [-l, --local] [-n, --no-activation] <target path>
```

Use `topo unset` to remove pico configurations in the specified target-path of STR.

Option	Description
[-l, --local]	Use this option to select the local container, unless another container is specified in the target path.
[-n, --no-activation]	Use this option to skip activation after changes in master registry.

Run the following command to remove a pico configuration.

`mzsh topo unset topo://container:<container>/pico:<pico>`

Example - Removing a pico configuration

```
$ mzsh topo unset topo://container:main1/pico:ec2
```

Example - Removing a pico attribute

```
$ mzsh topo unset topo://container:main1/pico:ec2/val:ec_type ecsa
```

File Paths in Attributes

When you enter a path that is relative to MZ_HOME in the value of an attribute, it is recommend that you use `${mz.home}` as a substitution.

In the following example MZ_HOME will be resolved to its current value e g `/home/user/mz.`

Example - Resolved path

```
$ mzsh topo set topo://container:main1/val:common.pico.rcp.tls.keystore $MZ_HOME/keys/platform.keys
```

The next example uses a path that is always relative to MZ_HOME.

Substituted path

```
$ mzsh topo set topo://container:main1/obj:common.pico.rcp.tls.keystore '{ keystore=${mz.home}"/keys"
}'
```

> **Note!**
>
> When you are using `${mz.home}` as a substitution, make sure to set attributes as part of an object, using the `obj` keyword.

Conflicting Attributes

The name of an attribute may contain the full name of another attribute. For instance, `mz.httpd.security.keystore` is a system property but its name is also a part of `mz.httpd.security.keystore.password.`

In this case you must ensure that the name of both properties are surrounded by quotes, or one of the properties will be overwritten at activation.

Example - Handling conflicting attributes, manual editing

```
common : {
    "pico.rcp.tls.keystore" : "home/mz/keys",
    "pico.rcp.tls.keystore.password" : "..."
}
```

When there are conflicting properties and you are using the mzsh topo command, also add single quotes, surrounding the target path (topo://..).

Example - Handling conflicting attributes, scripted editing

```
$ mzsh topo set 'topo://container:<platform container>/val:common."pico.rcp.tls.keystore"' "home/mz
/keys"

$ mzsh topo set 'topo://container:<platform container>/val:common."pico.rcp.tls.keystore.password"'
"..."
```

Return Codes

Listed below are the different return codes for the `topo` command:

Code	Description
0	Will be returned if the command is successful.
1	Will be returned if the argument count is incorrect or argument(s) are invalid.
3	Will be returned if the target path argument for the subcommand `get` does not exist.

2.1.16 version

```
usage: version
```

This command is used to find out the current version installed.

Return codes

Listed below are the different return codes for the `version` command:

Code	Description
0	Will be returned if the version was detected successfully.
1	Will be returned if the version could not be detected for some reason.

2.2 Available When the Platform Is Running

When the Platform is running and user is logged in, a more extensive list of commands, used to manage the system, will be available.

The following commands are available:

2.2.1 akka

```
usage: akka <subcommand> <arguments>
```

This command allows you to manage Akka clusters. For further information on Akka clusters, see 1.7 Akka Cluster in the System Administration guide.

The `akka` command has eight subcommands: `cluster-status`, `down`, `is-available`, `is-singleton`, `leaders`, `members`, `member-status` and `unreachable`. You must provide the name of the cluster and an SC in regex for all of the subcommands.

> **Hint!**
>
> You can enter a specific SC, or you can use `".*"` to run the command for all of the SCs in a cluster.

You can use the following subcommands with the `akka` command:

- cluster-status
- down
- is-available
- is-singleton
- leader
- members
- member-status
- unreachable

Arguments

Argument	Description
<cluster-name>	The name of the akka cluster
<sc in regex>	The SC that the akka cluster is on in regex
<cluster node address>	The address of the cluster node

cluster-status

```
usage: akka cluster-status <cluster name> <sc in regex>
```

Use `akka cluster-status` to get the current status for a specific akka cluster, e g information is provided on the member ring, available nodes, meta data etc.

Example - Using the command mzsh akka cluster-status

If you require information on the status of the cluster for your System Insight Akka cluster on one of the SCs:

```
$ mzsh akka cluster-status si sc5
```

The output returned is:

```
{
  "sc5": {
    "leader": "akka.tcp://si@172.27.239.120:6101",
    "members": [{
      "address": "akka.tcp://si@172.27.239.120:6101",
      "roles": ["si"]
    }],
    "unreachable": [],
    "roles": [{
      "name": "si",
      "leader": "akka.tcp://si@172.27.239.120:6101"
    }]
  }
}
```

down

```
usage: akka down <cluster name> <sc in regex> <cluster node address>
```

Use `akka down` to send a request to mark the node with the address that you provide as DOWN. For example, if one node in a cluster fails in a cluster of several nodes, you can tell the reachable nodes to remove the unreachable node using this command. For an example scenario, see 2.7.6 Managing Akka Cluster Failure.

Example - Using the command mzsh akka down

If you want to mark a cluster address for your System Insight as down:

```
$ mzsh akka down si sc5 akka.tcp://si@172.17.0.1:6151
```

is-available

```
usage: akka is-available <cluster name> <sc in regex>
```

Use `akka is-available` to find out if the availability of a node or nodes on a specific cluster. The output that you receive is `true` or `false`.

is-singleton

```
usage: akka is-singleton <cluster name> <sc in regex>
```

Use `akka is-singlementon` to find out if a specific cluster has a single node or not. The output that you receive is `true` or `false`.

leader

```
usage: akka leader <cluster name> <sc in regex
```

Use `akka leader` to find out who the current leader is for a cluster. The cluster address of the leader is returned.

Example - Using the command mzsh akka leader

If you want to know which is the current leader for your Conditional Trace Akka cluster:

```
$ mzsh akka leader akka-trace-cluster ".*"
```

The cluster address of the leader is provided for each node:

```
"sc1": "akka.tcp://akka-trace-cluster@172.27.239.120:5501"
"sc2": "akka.tcp://akka-trace-cluster@172.27.239.120:5501"
"sc3": "akka.tcp://akka-trace-cluster@172.27.239.120:5501"
```

members

```
usage: akka members <cluster name> <sc in regex>
```

Use `akka members` to list the addresses of the members for a specific cluster.

Example - Using the command mzsh akka members

If you require to know which members there are for your Conditional Trace Akka cluster:

```
$ mzsh akka members akka-trace-cluster sc1
```

The members are provided:

```
{
  "sc1": [{
    "address": "akka.tcp://akka-trace-cluster@172.27.239.120:5501",
    "roles": ["default"]
  }, {
    "address": "akka.tcp://akka-trace-cluster@172.27.239.120:5551",
    "roles": ["default"]
  }, {
    "address": "akka.tcp://akka-trace-cluster@172.27.239.120:5601",
    "roles": ["default"]
  }, {
    "address": "akka.tcp://akka-trace-cluster@172.27.239.120:6051",
    "roles": ["default"]
  }]
}
```

member-status

```
usage: akka member-status <cluster name> <sc in regex>
```

Use `akka member-status` to request the current status of a member node or member nodes. The output that you receive is `Up` or `Down`.

unreachable

```
usage: akka unreachable <cluster name> <sc in regex>
```

Use `akka unreachable` to find out which members of a specific cluster are unreachable.

Return Codes

Listed below are the different return codes for the `akka` command:

Code	Description
0	Will be returned if the command is successful.
1	Will be returned if the argument is invalid.

2.2.2 configuration

```
usage: configuration [options]
```

This command enables you to handle your configurations in various ways, such as enabling, disabling, checking status, etc.

Option	Description
[-de, --decrypt <passphrase>]	Use this option to decrypt all configurations of the default configuration type, or configurations matching the regular expression stated with the -n option. You must enter the passphrase that you used to encrypt the configuration(s).
[-d, --disable]	Use this option to disable all enabled configurations of the default configuration type, or configurations matching the regular expression stated with the -n option and of the type stated with the -t option. The default value is `false`.
[-e, --enable]	Use this option to enable all disabled configurations of the default configuration type, or configurations matching the regular expression stated with the -n option and of the type stated with the -t option. The default value is `false`.
[-en, --encrypt <passphrase>]	Use this option to encrypt all configurations of the default configuration type, or configurations matching the regular expression stated with the -n option. You must enter a passphrase.
[-l, --list]	Use this option to list all configurations of the default type, or of the type stated with the -t option. The default value is `false`.
[-o, --locked]	Use this option to list only locked configurations. The default value is `false`.
[-n, --name]	Use this option to indicate which configurations you want to perform an action on. Only configurations matching the regular expression stated with the -n option will be affected. The default value is `.*`.
[-s, --status]	Use this option to display the status of all configurations of the default type, or configurations matching the regular expression stated with the -n option and of the type stated with the -t option. The default value is `false`.
[-t, --type]	Use this option to indicate the type of configuration you want to perform an action on. Only configurations matching the stated type will be affected. **Note!** Currently, Event Notification and Alarm Detection are the only type of configuration supported. When used in combination with `--decrypt` or `--encrypt`, the `-t` option is ignored.

Note!

The options -d, -e, -l and -s are mutually exclusive, meaning that only one of the options will be applied even if you have stated several.

Examples - Using the configuration command

To list all the configurations in the Default folder:

```
$ mzsh <username>/<password> configuration -l -n "Default..*"
```

To list all the configurations in the Default folder with their status:

```
$ mzsh <username>/<password> configuration -s -n "Default..*"
```

To list all Alarm Detection configurations with their status:

```
$ mzsh <username>/<password> configuration -s -t "Alarm Detection"
```

To enable all Event Notification configurations in all folders beginning with name “event”:

```
$ mzsh <username>/<password> configuration -e -t "Event Notification" -n "event.*"
```

To list all locked Event Notification configurations:

```
$ mzsh <username>/<password> configuration -l -o -t "Event Notification"
```

To encrypt all configurations in the Default folder which have a name that begins with “test”:

```
$ mzsh <username>/<password> configuration -en "passphrase123" -n "Default.test.*"
```

Return Codes

Listed below are the different return codes for the configuration command:

Code	Description
0	Will be returned if the command was executed successfully.
1	Will be returned if the input could not be processed.
2	Will be returned if the input did not generate any action.
3	Will be returned if something unexpected happened.
4	Will be returned if one or more configurations were locked, or if the user does not have access to one or more of the configurations.

2.2.3 db-scripts

```
usage: db-scripts create-properties [-p, --properties-file <properties file>] |
extract [-d, --output-directory <db script directory>] [-p, --properties-file <properties file>]
```

This command is used to extract database creation files from an installed MediationZonesystem. This is useful when you want to change the Platform database type, e g from Derby to Oracle or PostgreSQL. For further information about how to change the database type, see 8.3 Changing Platform Database in the System Administrator's Guide.

create-properties

Use `db-scripts create-properties` to generate a properties file. For further information about the properties in this file, see 3.1.3.2 Properties for Oracle and 3.1.3.3 Properties for PostgreSQL in the Installation Instructions.

Option	Description
[-p, --properties-file <properties file>	The target path and filename for the properties file.

Example - Creating the properties file

```
MZ>> db-scripts create-properties --properties-file /tmp/dbscript.prop
```

extract

Use `db-scripts extract` to extract the database creation files. The database scripts are based on the properties file.

For further information about how to use these scripts, see 8.3 Changing Platform Database in the System Administrator's Guide.

Option	Description
[-p, --properties-file <properties file>	The properties file to be used as input.
[-d, --output-directory <db script directory>]	The location of the database scripts to be extracted.

Example - Extracting the create scripts

```
MZ>> db-scripts extract --output-directory /tmp/dbscripts \
--properties-file /tmp/dbscript.prop
```

Return Codes

Listed below are the different return codes for the `db-scripts` command:

Code	Description
0	Will be returned if the command was successful or unsuccessful.
1	Will be returned if the argument count is incorrect.
2	Will be returned if the properties file to be created already exists or if the specified target directory does not exist.
3	Will be returned if the specified target directory for the database scripts does not exist.
4	Will be returned if the `database` folder in the specified target directory for the database already exists.

2.2.4 derbybackup

```
usage: derbybackup <directory>
```

Performs a Derby database online backup. For further information about Derby backup, see the System Administration User's Guide.

Return Codes

Listed below are the different return codes for the derbybackup command:

Code	Description
0	Will be returned if the command was successful or unsuccessful.
1	Will be returned if the argument count is incorrect.

2.2.5 desktopadmin

```
usage: desktopadmin [options]
```

This command can be used for shutting down connected Desktops or for displaying an administrator message to all Desktop users. This is useful during system upgrades.

Options

The command accepts the following options:

Option	Description
[-a, --address]	Use this option to apply the entered command for only the stated Desktop session. To state a Desktop session, enter the session's <IP address:port> as presented using the --list option. **Note!** The `--address` command can only be used with one of the other commands at the time.
[-l, --list]	Use this option to list all the Desktop sessions currently connected to the platform. The returned list will show <Pico Name>, <IP Address:Port>, <User> for each session.
[-m, --message]	Use this option to write a message that will be displayed on all connected Desktops before shutdown. This message will also be displayed to any newly logged in Desktop. It can be removed using the reset option.
[-r, --reset]	Use this option to reset any message or timeout that have been set by a previous execution of the command.
[-x, --shutdown]	Use this option to enter a timeout interval in minutes that should pass before the Desktops are shutdown. If you want you can also append an optional message to this option within quotes, e g desktopadmin -shutdown 5 "My message".
[-s, --status]	Use this option to show the current status of message and timeout.

Return Codes

Listed below are the different return codes for the `desktopadmin` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if there was a problem parsing command parameters.
2	Will be returned if there was a failure scheduling shutdown of Desktops

2.2.6 disconnect

```
usage: disconnect [ -ecsa ] [ -ec ] [ -local ] [ -q ] [ -verbose] <running server process>
```

By using this command, it possible to let an EC/ECSA and its workflows run in disconnected mode during Platform upgrade, in order to avoid downtime.

> **Note!**
>
> Since a batch workflow cannot run without contact with the Platform, this command should only be executed on ECs/ECSAs running real-time workflows. If an EC/ECSA running a batch workflow is disconnected, the workflow will abort.

After disconnect, the EC/ECSA will continue executing real-time workflows without interference from the Platform and there will be no risk that new software is downloaded to the EC/ECSA during the Platform upgrade.

After the upgrade has been performed, each EC/ECSA has to be restarted in order to re-connect to the Platform, refer to 2.1.10 restart for more information.

The command accepts the following options:

Option	Description

[-ecsa]	Use this option to disconnect only ECSAs, for example:
	`MZ>> disconnect -ecsa`
	If using this option without any ECSA process specified, as in the example, all ECSAs will be disconnected.
	If a non-ECSA process is specified, fault code 4 will be returned, for example:
	`MZ>> disconnect -ecsa ec1`

[-ec]	Use this option to disconnect only ECs, for example:
	`MZ>> disconnect -ec`
	If using this option without any EC process specified, as in the example, all ECs will be disconnected.
	If a non-EC process is specified, fault code 4 will be returned, for example:
	`MZ>> disconnect -ec ecsa1`

[-local]	Use this option to disconnect an EC/ECSA that is running on a local machine, for example: `MZ>> disconnect -local ec1` In the example, the local EC process named ec1 will be disconnected. If using this option without any EC/ECSA process specified, all local ECs/ECSAs will be disconnected.
[-q]	Quiet mode. Use this option to eliminate the display of any report during execution.
[-verbose]	Verbose mode. This option will print extended error information.

By running the `disconnect` command from the Platform without any options, all running ECs/ECSAs, on all connected machines, will be disconnected without the need to login to each machine and disconnect them one by one:

```
MZ>> disconnect
```

> **Hint!**
>
> To get the latest status information for all local EC/ECSAs, execute the `status` command as described in 2.1.13 status.

Return Codes

Listed below are the different return codes for the `disconnect` command:

1-3: Errors only effecting one or more EC/ECSA.
10-11: Fatal error, no EC/ECSA was disconnected.

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if an EC/ECSA process is not running.
2	Will be returned if an EC/ECSA cannot be reached due to a communication problem.
3	Will be returned if an EC/ECSA does not exist.
10	Will be returned if the Platform cannot be reached due to communication problems.
11	Will be returned in case configuration files in `MZ_HOME/etc` contain errors that prevent the command from running.

2.2.7 dumpsyslog

```
usage: dumpsyslog [ -n <N>] [ -b ] [ -s <I | W | E | D> ] [ -d <date> ] [ -h <hour> ] [ -f <filename> ] [-q <C
| S | D>] [ -t ]
```

This command will allow displaying and saving entries from the System Log.

The command accepts the following filter options:

<table>
<tr><td>[-n <N>]</td><td>Maximum number (<code>N</code>) of entries to dump. By default the most recent entries in the System Log are shown.</td></tr>
<tr><td>[-b]</td><td>Will select entries from the beginning of the System Log, that is, the oldest entries instead of the most recent entries. This option is preferably used in conjunction with the <code>-n</code> option.</td></tr>
<tr><td>[[-s]]</td><td>Displays the severity type for each entry. The possible severity types are:

<table><tr><td><code>I</code></td><td>Information</td></tr><tr><td>W</td><td>Warning</td></tr><tr><td>E</td><td>Error</td></tr><tr><td>D</td><td>Disaster</td></tr></table></td></tr>
<tr><td>[-d <DATE>]</td><td>Filter on a certain date or time interval. The date must be given in the format specified in the <code>config.xml</code> file.

If only one date is given, the display will hold entries for that date only (time 00:00:00 to 23:59:59), unless the <code>-h</code> option is used.

<code>MZ>> dumpsyslog -d 1982-01-01</code>

Will give all log entries for the specified date from 00:00:00 to 23:59:59.

If an interval is given, the entries must be enclosed within quotation marks "d1 d2":

<code>MZ>> dumpsyslog -d "1982-01-01 2007-12-07"</code>

Will give all log entries between the two dates, from date1 (time 00:00:00) to date2 (time 23:59:59).</td></tr>
<tr><td>[-h <HOUR>]</td><td>Filter on a certain hour or time interval. Time is given in the format <code>hh:mm:ss</code> and if an interval is given the times have to be enclosed within quotation marks; <code>"h1 h2"</code>.

If only one time is given, the display will hold entries for that hour only (time 00:00:00 to 00:59:59). If giving two times, all entries in the interval will be displayed.

Note!

If both date and time intervals are entered; <code>-d "d1 d2" -h "h1 h2"</code> , the displayed entries will be from date <code>d1</code> at time <code>h1</code> to date <code>d2</code> at time <code>h2</code> .</td></tr>
<tr><td>[-f <FILENAME>]</td><td>Direct output to a file (including directory path).</td></tr>
<tr><td>[-q <C|S|D>]</td><td>Displays general information about the System Log entries.

<table><tr><td>C</td><td>Number of entries.</td></tr><tr><td>S</td><td>Number of entries of each severity.</td></tr><tr><td>D</td><td>Earliest and latest date of entries.</td></tr></table></td></tr>
<tr><td>[-t]</td><td>To include stack trace information in the System Log printout.</td></tr>
</table>

Return Codes

Listed below are the different return codes for the `dumpsyslog` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if there was a problem parsing command parameters.
2	Will be returned if there was an error when trying to access the system log database.
3	Will be returned if the output file already exists, or in the event of file write errors.

2.2.8 ecgroup

```
usage: ecgroup -add [-m, --member <group members>] [-t, --type <group type>] <group name>| -delete <group
name> | -list <group name>
```

This command enables you to add, delete or list pico groups of EC/ECSAs.

-add

Use `ecgroup -add` to add a new EC group:

Example - Adding an EC group

```
$ mzsh mzadmin/<password> ecgroup -add --member ec1 ec2 --type ec ecgroup1
```

Option	Description
[-m, --member <members>]	The name of the EC/ECSAs that to add to the group.
[-t, --type <type>]	The type of group, i e `ec` or `ecsa`.

-delete

Use `ecgroup -delete` to delete an EC group:

Example - Deleting an EC group

```
$ mzsh mzadmin/<password> ecgroup -delete ecgroup1
```

-list

Use `ecgroup -list` to view a specific ecgroup or all EC groups:

Example - Listing EC groups

```
$ mzsh mzadmin/<password> ecgroup -list ecgroup1
```

```
$ mzsh mzadmin/<password> ecgroup -list
```

Return Codes

Listed below are the different return codes for the ecgroup command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned in case of an illegal argument.
2	Will be returned in case of communication error.
11	Will be returned if you try to add a group that already exists.
12	Will be returned if the group could not be added.
13	Will be returned if the group could not be added because the group member you have have tried to add does not exist.
14	Will be returned if the group could not be added because the group name was invalid.
15	Will be returned if the group could not be added because the EC type was invalid.
16	Will be returned if the group could not be added because the group member was of an invalid type.
20	Will be returned if the group could not be deleted because the group does not exist.
21	Will be returned if the group could not be deleted.

2.2.9 importrollback

```
usage: importrollback <rollback file>
```

This command reverses the effects of the `systemimport` command by removing the imported configuration and reverting to an older configuration.

Note!

Use the `importrollback` command only to revert the `systemimport` command and not for the purpose of a general system rollback. To be able to properly rollback the entire system, make sure that you occasionally create a backup file with `systemexport`. This way, you can revert to an earlier system configuration by simply using `systemimport`.

Note!

The ECS Reprocessing Groups and ECS error codes are not removed by the `importrollback` command.

The *`rollback file`* parameter provides `importrollback` with information that enables MediationZone to reconstruct the status of your system prior to applying the `systemimport` command.

For further information about `systemimport` see 2.2.33 systemimport.

Return Codes

Listed below are the different return codes for the `importrollback` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the argument count is incorrect.
2	Will be returned if the rollback file does not exist.
3	Will be returned if the import rollback could not be started due to locked import.
1->	If import rollback is started, a return code greater than zero is returned if the rollback of any of the configurations fail. The exit code is then the number of failed rollbacks, i e return code is 1 if one rollback fails, 2 if two rollback fails, etc.

2.2.10 jcreate

```
usage: jcreate <project-path> <project-name>
```

This command allows you to create a java project space for the Java agent. For further information about the Java agent, see 9.46 Java Agent in the Desktop User's Guide

Return Codes

Listed below are the different return codes for the jcreate command:

0	Will be returned if the command is successful.
1	Will be returned if the argument is invalid.

2.2.11 kafka

```
usage: kafka --service-key <service key> [ --alter --topic <topic-name> --partitions <number of partitions>],
[ --create --topic <topic-name> --partitions <number of partitions> --replication-factor <number of
replications>],
[ --delete --topic <topic name>], [ --describe [--topic <topic-name-regexp>]], [ --list] [--topic <topic-name-
regexp>],
[ --verbose-on-error]
```

The `kafka` command enables you to configure your embedded Kafka. If you are using multiple or non-default Kafka embedded in, you use `kafka --service-key <service key>` to specify the Kafka Service Key.

> **Note!**
>
> You must specify the relevant embedded service key when you call the `kafka` command.

For further information on using embedded Kafka, refer to 9.48.1.1 Quick Start of Embedded Kafka in the Desktop user's guide.

The command accepts the following options:

`[--alter --topic <topic-name> --partitions <number-of-partitions>]`	This option allows you to add partitions to a topic. It is impossible to remove partitions. For further information on the arguments you can alter, refer to http://kafka.apache.org/082/documentation.html.
`[--create --topic <topic-name> --partitions <number-of-partitions> --replication-factor <number-of-replications>]`	You create your Kafka configuration using this option. You use this action to determine the name of the topic, how many partitions there are in the topic, and the replication factor for each partition. See the `--create` example provided below. You can create several topics, but you can only create them one by one. For further information on the arguments you can create, refer to http://kafka.apache.org/082/documentation.html.
`[--delete --topic <topic-name>]`	You use this option to delete a topic.
`[--describe [--topic <topic-name-regexp>]]`	This option provides a description of all the topics that have been created, or you can specify for which topic you want to view the description. See the `--describe` example provided below. For further information on the arguments, for which you can display a description, refer to http://kafka.apache.org/082/documentation.html.
`[--list [--topic <topic-name-regexp>]]`	A list of all the topics is provided, or you can specify a topic.
`[ --verbose-on-error]`	If you are using Kafka embedded in MediationZone, you use this option to display the stack trace information in more detail when an error occurs.

Example - Using the --create action

Creates a topic with the name `test1-topic`, with three partitions and a replication factor of one:

```
$ kafka --service-key kafka1 --create --topic test1-topic --partitions 3 --replication-factor 1
```

Example - Using the --describe action

```
$ kafka --service-key kafka1 --describe --topic test1-topic
```

Provides the following output:

```
Topic:test1-topic PartitionCount:3 ReplicationFactor:1 Configs
Topic: test1-topic Partition: 0 Leader: 1 Replicas: 1 Isr: 1
Topic: test1-topic Partition: 1 Leader: 2 Replicas: 2 Isr: 2
Topic: test1-topic Partition: 2 Leader: 3 Replicas: 3 Isr: 3
```

Return Codes
Listed below are the different return codes for the `kafka` command:

Code	Description
0	Will be returned if the command is successful.
1	Will be returned if a syntax error has occurred.
127	Will be returned if an internal Kafka error has occurred.

2.2.12 keytool

```
usage: keytool generate [-k, --keytore <keystore>] [-o, --overwrite] [--enable-tls <http/rcp>] [--password
<password>] |
enable-tls <http/rcp> [-k, --keystore] [-a, --alias <alias>] [--password <password>] | disable-tls <http/rcp>
```

This command is used to generate a Java keystore or to enable/disable the network security over HTTP and RCP. For more information regarding the network security feature, see 4. Network Security

You can use the following subcommands with keytool:

- generate
- enable-tls
- disable-tls

generate

Use keytool generate to create or update a self-signed Java keystore.

Option	Description
[-k, --keystore <keystore>]	States the directory and keystore filename in which the keystore will be stored. If this option is not used, the command will save the keystore file to the default location: `MZ_HOME/keys/keystore_generated.key`
[-o, --overwrite]	Use this option to overwrite any existing keystore.
[--enable-tls <http/rcp>]	Enables network security over HTTP or RCP after the keystore is successfully generated. If neither http or rcp is determined in the option, the command will enable the network security on both HTTP and RCP.
[–password <password>]	Use this option to include the password for the keystore. The user will be prompted for the password if this option is not used.

enable-tls <http/rcp>

Use keytool enable-tls to enable the network security over HTTP or RCP.

If neither http or rcp is determined in the option, the command will enable the network security on both HTTP and RCP.

Option	Description
[-k, --keystore <keystore>]	Use this option to validate the staged changes without performing the activation. If this option is not used, the command will save the keystore file to the default location: `MZ_HOME/keys/keystore_generated.key`
[-a, --alias <alias>]	Compare the provided hash value with the actual hash that represents the current state of active registry. The activation fails if the values are not equal. For further information, see hash below,
[–password <password>]	Use this option to include the password for the keystore. The user will be prompted for the password if this option is not used.

disable-tls <http/rcp>

Use keytool disable-tls to disable the network security over HTTP or RCP.

> If neither http or rcp is determined in the option, the command will disable the network security on both HTTP and RCP.

Return Codes
Listed below are the different return codes for the `keytool` command:

Code	Description
0	Will be returned if the command is successful.
1	Will be returned if a syntax error has occurred.
11	Will be returned if keystore already exists.
12	Will be returned if keystore does not exist.

2.2.13 kpimodel

```
usage: kpimodel display [-f,--file <filename>] [-v,--view <view>]| help [<subcommand>] |
validate [-f,--file <filename>]
```

This command is used to view and validate models in KPI Management.

display

Use `kpimodel display` to view a model.

```
$ mzsh kpimodel display --file <filename> --view tree
```

Example - Viewing a model

```
MZ>> kpimodel display --file /home/mz/Downloads/example_model.json --view tree
Region
   Region.KPI_01
   SubRegion
      SubRegion.KPI_02
      Area
         Area.KPI_03
```

Option	Description
[-f, --file <filename>]	An absolute path to the file that contains service model.
[-v, --view <view type>]	The view type. The only type that is currently available is `tree`.

help

Use `kpimodel help` to retrieve a description of a subcommand.

```
mz>> kpimodel help <subcommand>
```

validate

Use `kpimodel validate` to ensure the integrity of the service model.

```
mz>> kpimodel validate --file <filename>
```

Example - Validating a model

```
mz>> kpimodel validate --file /home/mz/kpi/servicemodels/model1.json
```

Option	Description
[-f, --file <filename>]	The file that contains service model.

Return Codes

Listed below are the different return codes for the `kpimodel` command:

Code	Description
0	Will be returned if the command is successful.
1	Will be returned if the argument count is incorrect or argument(s) are invalid or if the validation of the service model fails.

2.2.14 loglevel

```
usage: apl-loglevel [ display | set <level> ] <ec-names>
```

This command is used to select the active log4j APL logging configuration. The level argument to the command is based on the corresponding configuration filename in `$MZ_HOME/etc/logging`. For instance, the command below will activate the configuration in `ec1-apl-log4j-myloglevel.properties` on the Execution Context named `ec1` and `ec2-apl-log4j-myloglevel.properties` on `ec2`. If any of these files cannot be found, the command will (if possible) apply the settings of a file without the Execution Context prefix, i e `apl-log4j-myloglevel.properties`.

Example - Setting log level

```
$ mzsh <username>/<password> apl-loglevel set myloglevel ec1 ec2
```

You can view the configured log directory and refresh interval with the subcommand display.

Example - Display log level

```
$ mzsh <username>/<password> apl-loglevel display ec1 ec2
logdir = /opt/mz/etc/logging
refresh interval = 1000 ms
-------------------------------------------
Contents of: /opt/mz/etc/logging/ec1-apl-log4j.properties
-------------------------------------------
log4j.rootLogger=ALL, a
log4j.appender.a=com.digitalroute.apl.log.DRRollingFileAppender
log4j.appender.a.file=${mz.home}/log/{pico}_{workflow}.log
log4j.appender.a.layout=com.digitalroute.apl.log.JsonLayout
log4j.appender.a.layout.MdcFieldsToLog=pico, workflow, agent,
tag
-------------------------------------------
-------------------------------------------
Contents of: /opt/mz/etc/logging/ec2-apl-log4j.properties
-------------------------------------------
log4j.rootLogger=ALL, a
log4j.appender.a=com.digitalroute.apl.log.DRRollingFileAppender
log4j.appender.a.file=${mz.home}/log/{pico}_{workflow}.log
log4j.appender.a.layout=com.digitalroute.apl.log.JsonLayout
log4j.appender.a.layout.MdcFieldsToLog=pico, workflow, agent, tag
-------------------------------------------
```

The refresh interval is set to 1000 ms by default. You can change this value by setting the platform property `mz.logging.refreshinterval`.

When you activate a configuration the contents of the corresponding file is copied to `<ec name>-apl-log4j.properties`. Any changes in this file will become effective in the next watch interval. For more information about configuring log4j APL logging, see 10. log4j APL Logging Configurations in the System Administrator's Guide.

Return Codes
Listed below are the different return codes for the `apl-loglevel` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the command was unsuccessful.

2.2.15 pcommit

```
usage: pcommit <package-file> ...
```

This command is used to install a package in the system. The command should only be used by system administrators with authority to maintain the MediationZone software.

For further information, see the Development Toolkit User's Guide.

Return Codes

Listed below are the different return codes for the `pcommit` command:

Code	Description
0	Will be returned if the argument count is incorrect, or if the command is successful.
3	Will be returned if the package you want to commit does not exist.
4	Will be returned if the user name and password is missing.
5	Will be returned if the commit failed.
6	Will be returned if any uncaught failures occur during commit.

2.2.16 pexport

```
usage: pexport <package name> <target directory>
```

This command is used to copy .mzp files from the codeserver to a specified target directory.

Note!

You must specify the package name argument is case-sensitive. Package names that contain spaces must be surrounded by quotes.

Example - Exporting packages

```
$ mzsh mzadmin/<password> pexport ultra /home/tmp/packages
```

```
$ mzsh mzadmin/<password> pexport "Ultra XML Support" /home/tmp/packages
```

Return Codes

Listed below are the different return codes for the `pexport` command:

Code	Description
0	Will be returned if the command was successful or unsuccessful.
1	Will be returned if the argument count is incorrect.
2	Will be returned if the specified package does not exist

2.2.17 pico

```
Usage: pico -list <ip/host> ... | -view <pico name> | -add <ec name> <EC | ECSA> <ip/host> | -delete <ec name>
<EC |
ECSA> <ip/host> | -mark_for_shutdown <ec name> <EC | ECSA> <ip/host> <true | false>
```

This command is used to:

- List, view, add, or delete pico instances
- Signal to the Platform that an EC/ECSA is scheduled to be shut down

Note!

With the introduction of STR, it is typically no longer required to add pico instances. Provided that the value of the property `pico.rcp.server.host` is set to its default value or that the value can be found in the address attribute of the container, EC /ECSA, and SCs are implicitly added via the STR.

If `pico.rcp.server.host` does not match with any of the container addresses you can grant access to the system by:

- Adding the missing address in the container attribute `additional-addresses` in the STR.
- Setting the Platform property `mz.pico.skip-registration-check` to `true`. This will grant access for any pico host.
- Adding a new pico host in the Pico Manager.

-list

usage: pico -list <ip/host> ...

`pico -list` to list configured pico instances.

Example - pico -list

The following commands lists all pico hosts and pico instances.

```
MZ>> pico -list
```

Output:

```
ec_standard    ec    10.0.0.48
ec_standard    ec    10.46.100.26
ecsa_extended  ecsa  10.0.0.100
```

-view

usage: `pico -view <pico name>`

Use `pico -view` to view status of pico instances in the system.

Example - pico -view

The following displays the status for all pico instances in the system.

```
MZ>> pico -view
```

Output:

```
Pico                Start       Memory                Response      Error    Marked For
Name                Time        (Used, Commited, Max) Time (ms)     Status   Shutdown
ec1 (ip:port)       <time>      1.1, 24.2, 27.6       1             OK       true
ec2 (ip:port)       <time>      1.2, 23.3, 28.7       1             OK
MZSH:45881 (ip:port) <time>     5.9, 15.5, 47.5       2             OK
Platform (ip:port)  <time>      8.2, 20.0, 27.6       2             OK
```

The `Memory` column is a comma separated list of Used, Committed and Maximum memory, and `Response Time` is measured in milliseconds.

If `Error Status` indicates "Error", an OutOfMemoryError has occurred and additional information will be included in the output. For example, in case of an OutOfMemoryError on the Platform, the following could be shown:

```
Platform: Mon Jan 23 09:59:47 CET 2012 OutOfMemoryError on platform at platform-e6410.
See log/platform.log for more information
```

OutOfMemoryError is further described in 2.12 Out of Memory Info in System Log in the System Administration User's Guide.

If `Error Status` indicates "Connection failure", an RCP error has occurred.

If `Marked for Shutdown is set to true,` the EC/ECSA is scheduled to be shut down and will not be assigned workflows.

The following command displays the status of the pico instance named ec1.

```
MZ>> pico -view ec1
```

Output:

```
Pico Name:        ec1 (10.0.0.8:37197)
OS:               LINUX
OS Version:       2.6.38-11-generic-pae
Architecture:     I386
Processors:       8
Java Version:     1.8.0_121
Loaded Classes:   2854
```

-add

usage: `pico -add <ec name> <EC | ECSA> <ip/host>`

Use `pico -add` to add a pico instance to the system.

Example - pico -add

The following command adds an EC named ec1 to host 10.0.0.48.

```
MZ>> pico -add ec1 ec 10.0.0.48
```

-delete

usage: `pico -delete <ec name> <EC | ECSA> <ip/host>`

Use `pico -delete` to delete a pico instance from the system.

Example - pico -delete

The following command deletes an ECSA named ecsa1 from host 10.0.0.48.

```
MZ>> pico -delete ecsa1 ecsa 10.0.0.48
```

mark_for_shutdown

usage: `pico -mark_for_shutdown <ec name> <EC | ECSA> <ip/host> <true | false>`

Use `pico -mark_for_shutdown` to signal to the Platform that an EC/ECSA is scheduled to be shut down. As a result, the Platform will no assign workflows to the EC/ECSA.

Example - pico -mark_for_shutdown

The following command signals that the EC named EC1 on 10.0.0.48 should not be assigned workflows.

```
MZ>> pico -mark_for_shutdown ec1 ec 10.0.0.48 true
```

Return Codes

Listed below are the different return codes for the `pico` command:

Code	Description
0	Will be returned if the command was successful, or if arguments are missing.
1	Will be returned if arguments could not be parsed
2	Will be returned if the communication with the Platform failed.
3	Will be returned if checking of user privileges failed, or if pico already exists when trying to add a new pico.
4	Will be returned if the user does not have permission to add or delete picos.
5	Will be returned if an unexpected error occurred.

2.2.18 plist

```
usage: plist [ -name <package name> ] [ -gen <nogenerated|onlygenerated|all> ] [ -archive [ <package archive
name> ]
[ -class [ <java class name> ] ] [ -resource [ <java resource name> ] ] [ -compact ]
```

This command lists packages installed in the system. The list shows an overview of each package's user, version, repository, revision, and date. You can use options to filter packages and view detailed information such as archives, java classes, and resources.

Option	Description
-name <package name>	Use this option to show package info for <package name>.
-gen <nogenerated|onlygenerated|all>	Use this option to show packages that are generated by Ultra and APL configurations.
-archive [<package archive name>]	Use this option to show archive information for <package archive name>. Omit <package archive name> to show information for all archives.
-class [<java class name>]	Use this option together with archive <package archive name> to show class information for an archive. Omit <java class name> to show information for all classes.
-resource [<java resource name>]	Use this option together with archive <package archive name> option to show resource information for an archive. Omit <java resource name> to show information for all resources.
-compact	Use this option to show the package information in a compact format.

Example - Using the -name flag

```
$ mzsh plist -name Core
```

Output:

```
Overview of Core:
 Version: 7.2.0.0
 Repository:
 Revision: ae3245cb47185027d32f28c2098af8fdd0f724db
 Date: 2015-09-02 17:35:38
```

Example - Using the -gen flag

```
$ mzsh plist -gen onlygenerated
```

Output:

```
Overview of _JavaCode_MZ1438686422047:
 Version: 8/7/15 3:25 PM
 Repository: n/a
 Revision: 8/7/15 3:25 PM
 Date: 2015-08-07 15:25:02
```

Example - Using the -archive flag

```
$ mzsh plist -name Analysis -archive
```

Output:

```
Overview of Analysis:
 Version: 7.2.0.0
 Repository:
 Revision: ae3245cb47185027d32f28c2098af8fdd0f724db
 Date: 2015-09-02 17:35:38
 Archives in Analysis:
 (Analysis, 7.2.0.0) analysis/mz-ANALYSIS-ui.jar, 70471 bytes (platform)
 (Analysis, 7.2.0.0) analysis/mz-ANALYSIS-main.jar, 87980 bytes (execution)
```

Example - Using the -resource flag

```
$ mzsh plist -name Analysis -archive analysis/mz-ANALYSIS-main.jar -resource
```

Output:

```
Details for package Analysis archive analysis/mz-ANALYSIS-main.jar:
 (Analysis, 7.2.0.0) analysis/mz-ANALYSIS-main.jar, 87980 bytes (execution)
 Resources for analysis/mz-ANALYSIS-main.jar:
 META-INF/MANIFEST.MF
 com/digitalroute/wfc/analysis/AnalysisTC_en.properties
 com/digitalroute/wfc/analysis/analysis.svg
```

Example - Using the -class flag

```
$ mzsh plist -name Analysis -archive analysis/mz-ANALYSIS-main.jar -class
```

Output:

```
Details for package Analysis archive analysis/mz-ANALYSIS-main.jar:
 (Analysis, 7.2.0.0) analysis/mz-ANALYSIS-main.jar, 87980 bytes (execution)
 Classes for analysis/mz-ANALYSIS-main.jar:
 com.digitalroute.wfc.analysis.AnalysisAgent
 com.digitalroute.wfc.analysis.AnalysisRealtimeExec
 com.digitalroute.wfc.analysis.AnalysisRealtimeInsp$1
 ...
```

Return Codes

Listed below are the different return codes for the `plist` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the command was unsuccessful.

Note!

This command does not require that the user is logged in.

2.2.19 premove

```
usage: premove <package-name>
```

Removes the selected package from the system. The command should only be used by system administrators with authority to maintain the MediationZone software.

For further information about managing packages, see the System Administrator's Guide.

Return Codes

Listed below are the different return codes for the `premove` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if there is no package with the specified name.

2.2.20 refreshdbmeta

```
usage: refreshdbmeta [options]

This command is used to refresh the metadata for existing Database profiles.

  Options:
    -p, --profile                  State a specific profile to be refreshed.
                                                    The name should be entered in the following format:
                                                    Folder.ProfileName
                                                    If no profile is set, all existing profiles will be
refreshed.
                           Default: []
    -v, --verbose          More detailed output from the refreshdbmeta command.
                           Default: []
```

Options

Option	Description
[-p, --profile]	This option can be used if you want to state a specific profile to be refreshed. If this option is not used, all profiles will be refreshed.
[-v --verbose]	Use this flag for detailed output from the `refreshdbmeta` command.

Return codes

Listed below are the different return codes for the refreshdbmeta command:

Code	Description
0	Will be returned if the metadata was successfully refreshed.
1	Will be returned if the stated profile is not valid or could not be found.

2.2.21 reloadkeystore

```
usage: reloadkeystore
```

Reloads the keystore file located in the directory specified by the property: `mz.cryptoservice.keystore.path`, with the password specified by the property: `mz.cryptoservice.keystore.password`.

Return Codes

Listed below are the different return codes for the `encryptpassword` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the command was unsuccessful.

2.2.22 resumeexecution

```
usage: resumeexecution
```

This command enables you to resume the execution of a workflow, or a workflow group, that is stuck in the Hold state.

> **Example 9.**
>
> If the workflow, or the workflow group, does not leave the Hold state after running `systemimport -holdexecution` , either due to a system crash or because you performed `Ctrl+C`, use this command to resume execution.

For further information, see Section 2.2.2.23.6, [-he|-holdexecution [r | sr | sir | wr]]

For information about workflow and workflow group states, see the Desktop User's Guide.

Return Codes

Listed below are the different return codes for the `resumeexecution` command:

Code	Description
0	Will be returned if all workflows and groups resume execution normally.
1	Will be returned if any errors occur.

2.2.23 service

```
usage: service start [-s, --scope <all | custom | standard>] | restart [--publish-only] | dump | info [-d, --
detailed]
[-i, --instance <provider/service instance>] | list | update [-c, --command] [-i, --instance <provider/service
instance>]
```

Note!

This command is valid only for the MZ_HOME owner.

The command enables you to manage the services which are configured and hosted in Service Contexts.

start

Use `service start` to start the platform and Service Contexts. When you call service start, the platform orchestrates the start-up of defined service instances (`standard-services.conf` and `custom-services.conf`) and saves the configuration produced.
The command accepts the following option:

[-s, --scope <all | custom | standard>	Use this option to choose which services you want to start.

Example.

To start both standard and custom services, you may call the following:
```
mzsh service start
```
or
```
mzsh service start --scope all
```

To start only custom or standard services, run
```
mzsh service start --scope custom
```
or
```
mzsh service start --scope standard
```

restart

Use `service restart` when the platform is restarted while some or all of the Service Contexts are kept running, e g, after Platform failure and recovery, or after scheduled maintenance of the Platform. When you call `service restart`, the command applies the configuration state saved by the `service start` command, so that all service instances are configured the same way as when the `service start` was issued. At the very least, service restart republishes the information required to connect to a service instance, e g, from a workflow). However, it also causes any service members found to be missing in the Service Contexts, to be restarted - this may happen when a failure and recovery affects the platform and some of the Service Contexts.

When the Platform is restarted, while Service Contexts are active, you must also restart the embedded services using this command.

The command accepts the following option:

`[--publish-only]`	Use this option to only publish the service configuration in the platform registry. This option can be used if you first upgrade the platform and then execute a rolling upgrade of Service Contexts. Using the option will not restart any service members.

dump

Use `service dump` to display very detailed information in HOCON format on all the services running. This information is mainly intended for support and troubleshooting.

info

Use service info to display information on all of the services running or a specific service instance.

The command accepts the following options:

[-d, --detailed]	Use this option to display detailed information on all of the services running.
[-i, --instance <provider/service instance>]	Use this option to display detailed information on a specific service instance.

Example.

For information on all of the services running, you call the following:
`mzsh service info`

For detailed information on all of the services running, you call the following:
`mzsh mzadmin/dr service info --detailed`

For information on a specific service, you must specify the service provider/service instance, for example:
`mzsh service info --instance kafka/kafka1`

list

Use service list to display information on all available service providers.

update

Use `service update` to send custom commands to the service instance.

[-c, --command]	Use this option to specify a custom command.
[-i, --instance <provider/service instance>]	Use this option to specify a specific service instance that should be updated.

Return Codes

Listed below are the different return codes for the service command:

Code	Description
0	Will be returned if the command was successful
1	Will be returned if an argument is invalid
2	Will be returned when a known error occurs
3	Will be returned when a general error occurs

2.2.24 sldreg

```
usage: sldreg [topologyFilePath] [outputFilePath]
```

This command allows you to generate the SLD (System Landscape Directory) configuration data for registration in the SAP Solution Manager.

It is optional to enter the sld.conf file as an argument, but if you do not enter the sld.conf file, it is assumed that the relevant file is the default, namely `MZ_HOME/etc/sld.conf`. If you want to use another directory and another filename, you have the option to enter one or two different directories/file names.

Example - sldreg

Using No Arguments

```
$ mzsh sldreg
```

The file `cluster-sld.conf` will be generated in the directory `$MZ_HOME/sap-sldreg/`.

Using Directory Arguments

```
$ mzsh sldreg mydirectory/1.conf mydirectory/2.conf
```

The files `1.conf` and `2.conf` will be generated in the directory `$MZ_HOME/mydirectory/`.

The `sld.conf` file provides the static information for the output generated.

You must run this command from the machine where the Platform is running.

Return Codes

The different return codes for the `sldreg` command are listed below:

Code	Description
0	Will be returned if the command is successful.
1	Will be returned if platform is not running
2	Will be returned if writing the temporary topology file fails. Will be returned if converting to the sld topology file fails.
7	Will be returned if the command is not executed on the machine where the platform is running or if the topologyFilePath is given and does not exist.

2.2.25 slowmethods

```
usage: slowmethods [ -all ] [ -class regexp ] [ -threshold <byte code size> ]
```

This command enables you to list too long methods including more than 8000 byte code instructions (or the size set by the `-threshold <byte code size>` option). When there are too many byte code instructions in a method, they tend to run very slowly because the JVM will not be able to perform a proper JIT compilation. This will result in lower runtime performance.

> **Hint!**
>
> Two methods including 7000 byte code instructions is a better alternative than one method with 14000 instructions.

The following is an example of `mzsh mzadmin/dr slowmethods` output:

> **Example.**
>
> ```
> Scanned 20 classes
>
> The following methods have too many byte code instructions:
>
> 9393 Folder.Workflow.Agent.method: Default.myWorkflow.RequestAnalysis.consume
> ```

`9393` is the method size in bytes. `Default` is the folder where the workflow is stored, `myWorkflow` is the name of the workflow, `RequestAnalysis` is the agent that includes the too long method, and `consume` is the method.

> **Note!**
>
> If the workflow configuration cannot be retrieved using an internal key, the class and method name will be returned instead, as in the following example:

> **Example.**
>
> ```
> 8930 com.mysql.jdbc.DatabaseMetaData.getTypeInfo
> ```

`8930` is the method size in bytes, `com.mysql.jdbc.DatabaseMetaData` is the class name and `getTypeInfo` is the too long method.

Option	Description
[-all]	Use this option to search through all code in the cache, not only the APL code.
[-class <regexp>]	Use this option to add a regular expression to be matched when searching through the code. For example, `mzsh mzadmin/dr slowmethods -class com.<product>.*` will only search for classes and packages in `com.<product>`.
[-threshold <byte code size>]	Use this option to set a byte code threshold, to be able to list slow methods with a byte code size lower than "8000".

Return Codes

Listed below are the different return codes for the `slowmethods` command:

Code	Description
1	Will be returned if an argument is invalid.

2.2.26 spaceactivation

```
usage: spaceactivation <passive space name> [-list [-format data-only|full]] [-set <name of workflow or
workflow group> <ENABLE|DISABLE>]
[-unset <name of workflow or workflow group>]  [-setdefault <name of workflow or workflow group>
<ENABLE|DISABLE>]
```

This command allows you to enable or disable workflows and/or workflow groups for space activation in a passive space. When you then use the `spacecopy` command to copy your passive space to the active space, the workflows and/or workflow groups in space activation are run automatically. The `spaceactivation` command can only be used in passive spaces. This allows you to control the activation mode of workflows and/or workflow groups when executing a `spacecopy`.

The command accepts the following options:

Option	Description
[-list [-format data only|full]]	This option displays all of the workflows and workflow groups in the passive space for space activation. You can also select the format of the list between data only or full. This option supports regular expression.
[-set <name of workflow or workflow group> <ENABLE|DISABLE>]	Use this option to set a workflow or workflow group for space activation, and select ENABLE or DISABLE. Using this option, adds the workflow or workflow group to the list for space activation. This option supports regular expression.
[-unset <name of workflow or workflow group>]	Use this option to remove a workflow or workflow group from the space activation list. This option supports regular expression.
[-setdefault <name of workflow or workflow group> <ENABLE|DISABLE>]	Set the default activation mode for a new workflow or workflow group by selecting ENABLE or DISABLE.

Hint!

To display which workflows and or workflow groups are in a space, use the `wflist` command. See 2.2.58 wflist.

Return Codes

Listed below are the different return codes for the `spaceactivation` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned when a general error occurs.
2	Will be returned if interactive mode has been enabled.
3	Will be returned if the parameters are incorrect.
4	Will be returned if the source space does not exist.

2.2.27 spacecopy

```
usage: spacecopy [ -c, --clear ] [-f, --force ] [-s, --stop-workflows <timeout>] [-sf, --stop-workflows-force
<timeout>]
<source space name | source filename.zip> <destination space name | destination filename.zip>
```

You can only execute this command in non-interactive mode.

This command copies the content of one space to another. This allows you to make changes to configurations in a passive space without affecting the configurations running in the active space.

You can use spacecopy with files in order to transfer configurations from one system to another.

> **Note!**
>
> In order to avoid compatibility issues, the source and destination system must be identical in terms of version and installed packages.

If the contents of a source file was created with a different version of MediationZone, a message is displayed:

```
Caution! - Space archive file <source> was created using <version>, running version is <version>. Do you want
to continue? [Y/N]
```

> **Note!**
>
> Since you duplicate the content of a space during a spacecopy operation, you must ensure that you have enough extra disk space available.

Before the content of the source space is copied to a space or an existing file, a message is displayed:

```
Caution! - '<destination>' will be overwritten. Do you want to continue? [Y/N]
```

[-c, --clear]	Use this option if you wish to empty the source space when you execute a `spacecopy`. You cannot use this option if you are copying from or to a zip file. **Hint** By using the `--clear` option, you lose a copy of the content you have copied, but it makes the `spacecopy` operation faster as the space contents are moved from one location to another.
[-f, --force]	Use this option to force a spacecopy. A spacecopy is then executed without issuing the questions above.
[-s, --stop-workflows <timeout>]	Use this option to stop your workflows before the space is switched in the `spacecopy` operation. If the timeout expires, the `spacecopy` is cancelled. Timeout is in minutes. **Note** The workflows causing timeout might be stopped later by the pending stop action. However, scheduled workflows are only stopped until the next scheduling period.
[-sf, --stop-workflows-force <timeout>]	Use this option to stop your workflows before the space is switched in the `spacecopy` operation. If the timeout expires, an attempt is made to immediately stop the workflows. The timeout is reused for the 'stop immediate' attempt. Timeout is in minutes.

> **Note!**
>
> You cannot execute this command if an import to the source space is in progress, or if workflows are running in the destination space.

During a spacecopy, any destination space activity, e.g. a running desktop or mzsh command), is rejected. For further information about managing configuration spaces, see the Configuration Spaces documentation.

Return Codes

Listed below are the different return codes for the `spacecopy` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned when a general error occurs.
2	Will be returned if interactive mode has been enabled.
3	Will be returned if the parameters are incorrect.
4	Will be returned if the source space does not exist.
5	Will be returned if the destination space does not exist.
6	Will be returned if the source space was not copied.

2.2.28 spacecreate

```
usage: spacecreate <space name>
```

You can only execute this command in non-interactive mode.

This command creates a space which only contains the mandatory system tasks. You must create a destination space before you copy an existing space. For further information about managing configuration spaces, see the Configuration Spaces documentation.

Return Codes

Listed below are the different return codes for the spacecreate command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned when a general error occurs.
2	Will be returned if interactive mode has been enabled.
3	Will be returned if the parameters are incorrect.
4	Will be returned if the destination space already exists.
5	Will be returned if the destination space was not created.

2.2.29 spacelist

```
usage: spacelist
```

You can only execute this command in non-interactive mode.

This command returns a list of all the configuration spaces in the system. If you have not created any spaces, this list is not available. For further information about managing configuration spaces, see the Configuration Spaces documentation.

Return Codes

Listed below are the different return codes for the `spacelist` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned when a general error occurs.
2	Will be returned if interactive mode has been enabled.
3	Will be returned if the parameters are incorrect.
4	Will be returned if a list was not returned.

2.2.30 spaceremove

```
usage: spaceremove [-f, --force] <space to be removed>
```

You can only execute this command in non-interactive mode.

This command deletes a space. Before the space is deleted, a message is displayed:

```
Caution!! The spaceremove operation will remove space '<space to be removed>'. Do you want to
continue? [Y/N]
```

Option	Description
[-f, --force]	Use this option to force a `spaceremove`. A `spaceremove` is then executed without issuing the question above.

When this command is executed, if there is a user working in the space to be removed, they receive a message informing them that a spaceremove has been executed, and they are rejected from the system.

> **Note!**
>
> Do not remove the active space. You can only remove passive spaces.

For further information about managing configuration spaces, see the Configuration Spaces documentation.

Return Codes

Listed below are the different return codes for the `spaceremove` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned when a general error occurs.
2	Will be returned if interactive mode has been enabled.
3	Will be returned if the parameters are incorrect.
4	Will be returned if the space to be removed does not exist.
5	Will be returned if the space was not removed.

2.2.31 spark

```
usage: spark app-info <cluster> | app-stop <cluster> <application> |app-submit <cluster> <application> |
app-flush <cluster> <application> | cluster-info <cluster> | cluster-shutdown <cluster> | cluster-start
<cluster> |
cluster-update <cluster> | help [<command>] | worker-restart <cluster>
```

You can use the following subcommands with spark:This command is used to manage the Spark cluster and Spark applications in KPI Management.

- app-flush
- app-info

- app-stop
- app-submit
- cluster-info
- cluster-shutdown
- cluster-start
- cluster-update
- help
- worker-restart

The Spark cluster is specified in the following format:

```
<service provider>/<service instance>
```

Example - Spark cluster

```
spark/spark1
```

app-flush

Use spark app-flush If you stop running a KPI-management system and want to make sure all the pending KPIs are sent to the Kafka output topic.

Example - app-flush

```
mzsh spark app-flush spark/spark1 spark-kpi-app1
```

app-info

Use spark app-info to display information about Spark applications in the specified cluster instance.

Example - app-info

```
mzsh spark app-info spark/spark1
```

Example output

```
+-----------------------------------------------------------------------------------------------------------
```

Note!

In order to use this command you must change log level of Spark to INFO. To set the log level set the property `log4j.rootCategory` in `MZ_HOME/external/spark/runtime/conf/log4j.properties`.

app-stop

Use spark app-stop to stop a Spark application on the cluster.

Example - app-stop

```
mzsh spark app-stop spark/spark1 spark-kpi-app1
```

app-submit

Use spark app-submit to start a Spark application on the cluster.

Example - app-submit

```
$ mzsh spark app-submit spark/spark1 spark-kpi-app1
```

cluster-info

Use spark cluster-info to retrieve information about the Spark cluster.

Example - cluster-info

```
mzsh spark cluster-info spark/spark1
```

Example output

```
+-----------------------------------------------------------------------------------------------------
```

cluster-shutdown

Use `spark cluster-shutdown` to shut down the Spark cluster. This command does not have any impact on the Spark service.

Example - cluster-shutdown

```
$ mzsh spark cluster-shutdown spark/spark1
```

cluster-start

The Spark cluster is started automatically when you start the Spark service. Use `spark cluster-start` to start the cluster after a shutdown.

Example - cluster-start

```
$ mzsh spark cluster-start spark/spark1
```

cluster-update

Use spark cluster-update after changes in the `deployment-info` configuration of the Spark service. For instance, after adding an SC.

Example - cluster-update

```
$ mzsh spark cluster-update spark/spark1
```

help

Use spark `help` to retrieve a description of a subcommand.

Run the following command for an overview of the various `spark` subcomands

```
$ mzsh spark help
```

Run the following command for a description of a specific subcomand

```
$ mzsh spark help <command>
```

worker-restart

Use spark worker-restart to restart "dead" Spark workers on each Service Context. This command has no impact on running Spark workers.

Example - worker-restart

```
$ mzsh spark worker-restart spark/spark1
```

2.2.32 systemexport

```
usage: systemexport [ -select <xml-selection file> ] [ -includesysdata ] [ -overwrite ] [ -directory ] <export
file|directory> [password]
```

This command exports configuration data from MediationZone to a file or directory in the client's local repository. As this command is available only to the `mzadmin` user, all the entries are exported, regardless of their access permissions.

This command's output log information is displayed during the command execution. Although no log file is generated, you can view log information in the shell and save it in a file.

[-select]	Specify the name of the Selection file that you want to use. For information about this parameter, see XML Selection File in 2.2.33 systemimport.
[-includesysdata]	Use this option to include system data such as users, pico hosts, and other system data.
[-overwrite]	Use this option to specify that the export- file or directory should be overwritten.
[-directory]	Use this option to specify that the export data is sent to a directory, instead of a file.
`<export file|directory>`	Specify the path of the directory or ZIP file that contain the configurations that you want to export.
[password]	To export encrypted configurations, provide a password.

Return Codes

Listed below are the different return codes for the `systemexport` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the argument count is incorrect or if the export fails.
2	Will be returned if the output directory exists, or if write permission is missing, or if the directory cannot be created for any other reason.
3	Will be returned if the XML selection file cannot be read.
4	Will be returned if any errors was reported during export.

2.2.33 systemimport

```
usage: systemimport
[ -s|-skipexisting ]
[ -pp|-preservepermissions ]
[ -nameconflict re|an|sk|ask ]
[ -keyconflict re|an|sk|ask ]
[ -namekeyconflict rn|rk|an|sk|ask ]
[ -he|-holdexecution [ r|sr|sir|wr ] ]
[ -nr|-norollback ]
[ -no|-newowner]
[ -select <xml-selection file> [ -dryrun ]  [ -onerror < ABORT | ASK | IGNORE > ]  ]
[ -m|-message ]
[ -u|-upgrade ]
[ -eu|-enableusers ]
<export file|directory> [password]
```

This command imports a ZIP file or a directory. If an import entry is in conflict with an entry that already exists in the current system, the entry will not be imported. An example of such a conflict might be an identical entry name, but a different ID. This command is only available to the mzadmin user, and therefore all the entries will be imported, regardless of their access permissions.

A workflow group that is scheduled to start while an import activity is happening, will not start until the import is complete.

[-s|-skipexisting]

If you choose to set the `skipexisting` parameter, the imported data will not overwrite existing configurations that have the same key, name, type, and folder. This means that repeatedly importing a configuration, does not overwrite the data.

[-pp|-preservepermissions]

When user permissions are modified, set the `preservepermissions` parameter to prevent user permissions in the current system from being updated while importing a configuration.

[-nameconflict re | an | sk | ask]

This parameter enables you to identify an imported configuration with a name that is identical to the name of a configuration that already exists in your system. Using this parameter requires that you specify what you want to do with the imported configuration:

- `re` (REplace): Overwrite the current configuration with the imported one.
- `an` (Add New): Add the imported configuration to your system. To resolve the name conflict, the new configuration's name is extended with `_1` at the end.

  Note!

  A name that already ends with `_n` is modified at the end to `_n+1`.

- `sk` (SKip): Ignore the conflicting imported configuration.

- `ask`: Prompt when a conflict is detected.

Note!

If you do not set this parameter, the conflicting configuration is skipped and ignored.

[-keyconflict re | an | sk | ask]

This parameter enables you to identify an imported configuration with a key that is identical to the key of a configuration that already exists in your system. Using this parameter requires that you specify what you want to do with the imported configuration. For information about the `keyconflict` options, see [-nameconflict re | an | sk | ask].

Example.

The command `mzsh mzadmin/dr systemimport -keyconflict ask import.zip` is applied with the ask option and detects a key conflict.

```
 ...
 [Configuration/Event Notification]
  - Importing Default.extraKey2...
 Event Notification configuration with the same key
 but with a different type, folder or name already exist!

 Existing name is: Default.existingConfiguration
 Imported name is: Default.importedConfiguration

 What do you want to do?
 1) Replace existing configuration
 2) Import as a new configuration
 3) Skip
> 1
 Use this alternative for all configurations of this type?
 1) Yes
 2) No
> 2
```

Note!

If you do not set this parameter, the conflicting configuration is skipped and ignored.

[-namekeyconflict rn | rk | an | sk | ask]

This parameter enables you to identify an imported configuration with a name as well as a key that are identical to a name and a key of two different configurations that already exist in your system. Using this parameter requires that you specify what you want to do with the imported configuration:
`rn` (Replace Name conflict): Replace the configuration that name conflicts with the imported configuration.
`rk` (Replace Key conflict): Replace the configuration that key conflicts with the imported configuration.
an (Add New): Add the imported configuration to your system.
`sk` (SKip): Ignore the conflicting imported configuration.
`ask`: Prompt a message when a conflict is detected.

Note!

If you do not set this parameter, the conflicting configuration is skipped and ignored.

[-he|-holdexecution [r | sr | sir | wr]]

Example. Using the holdexecution parameter

```
$ systemimport -s -he r export.zip
$ systemimport -s -he sr export.zip
$ systemimport -s -he sir export.zip
$ systemimport -s -he wr export.zip
```

Use the holdexecution parameter to prevent scheduled workflow groups from being started while importing configurations.

If a workflow or a workflow group does not stop within 5 minutes (300 seconds) when applying systemimport with either one of the following holdexecution parameters: `sr`, `sir`, and `wr`, a timeout will occur. You can change the timeout value by setting the Platform property `mz.import.suppress.timeout`.

When import is complete and a workflow group is still running, systemimport -holdexecution `[ r | sr | sir | wr ]` awaits the current running workflow member to come to a stop, and then restarts the whole group instead of continuing the execution of the member that follows.

If you do not specify any of the `r`, `sr`, `sir` or `wr` options:

- A batch workflow or a workflow group will remain suppressed until all the workflows finish executing. Then, the workflow or the members of the workflow group, become idle.
- A real-time workflow group will return to the running state.

`systemimport -holdexecution` generates events. To retrieve the events data, configure the Event Notification Editor to transfer it according to your preferences. For information about the Suppressed Event event type, see 4.3.7 Suppressed Event in the Desktop user's guide.

holdexecution Parameters

Select the action that should resolve held executions:

- `r` (restart): When the import activity is done, and a workflow group is partly executed, specifying this option will restart that workflow group. Workflow groups that are fully executed, will not be restarted. A workflow that is started manually, will be restarted if it has been stopped.
- `sr` (stop and restart): Stops the workflows or workflow groups that are still running after an import is complete, waits for them to come to a stop or finish processing a batch, and then restarts synchronously all the workflow groups and all the manually started workflows that had not executed completely within the timeout boundaries.

  Note!

  A workflow that becomes Unreachable after a system import has begun, will be restarted only when and if the contact with the Execution Context that it runs on, is regained while still importing.

  If a workflow is unreachable when system import is started, the import is aborted and the following error message is generated: `Abort Import: At least one wf that is Unreachable`.

  For further information about the Unreachable state, see 3.1.11 Workflow Monitor in the Desktop User's Guide.

- `sir` (stop immediately and restart): Stops the workflows or workflow groups that are still running after an import is complete, even in the middle of processing a batch, and synchronously restarts all the workflow groups and all the manually started workflows that had not executed completely within the timeout boundaries.

  Note!

  An Unreachable workflow is restarted once contact with the Execution Context that it runs on, is regained. For further information about the Unreachable state, see 3.1.11 Workflow Monitor in the Desktop User's Guide.

- `wr` (wait for completion and restart): After an import is complete, synchronously restarts all the workflow groups and all the manually started workflows that had not executed completely within the timeout boundaries.

Note!

Any workflow that runs past the timeout limit is restarted as soon as it completes execution.

For further information about holding an execution, see the description of the Suppressed workflow group state in the Desktop User's Guide .

[-nr|-norollback]

When you use systemimport, a file that contains rollback information will be created. This file contains data about configuration changes during the system import, and is saved in the directory where you apply the command. You use the rollback file to undo a system import and return to the configuration that you had before the system import.

Note!

Use the importrollback command only to revert the `systemimport` command and not for the purpose of a system rollback. For further information see 2.2.9 importrollback.

To suppress the creation of the rollback file, provide either a `nr` or a `norollback` option.

[-no|-newowner]

Change ownership of the configuration on import. Must match an user already defined in the 6.1 Access Controller.

[-select <xml-selection file> [-dryrun] [-onError < ABORT | ASK | IGNORE >]

The `-select <xml-selection file>` parameter enables you to:

- Provide systemimport with an XML file that specifies your selection of configurations and workflow tables.
- Use `[ -dryrun ]` to test data compatibility prior to actually importing
- Use `[ -onError < ABORT | ASK | IGNORE > ]` to manage an occurrence of an error

XML Selection File

This selection information that you find in the XML selection file corresponds to the selection information that you specify on the System Import tool view, in the Desktop user interface.

The XML selection file consists of two main tags:

- `<configurations>`: contains your configuration import selections
- `<workflows>`: contains your workflow tables import selections

<configurations>

Select configurations from the `<Export file|directory>` that systemimport imports.

- Use the
- `resolveDependencies` attribute to either include (true), or ignore (false), dependent configurations. See the following example.

Example - The resolveDependencies attribute

```
<import>
 <configurations>
<!-- Ignoring dependencies of the Default.x configuration-->
 <configuration name="Default.x" resolveDependencies="false"/>
<!-- Including dependencies of the Default.y configuration-->
 <configuration name="Default.y" resolveDependencies="true"/>
<!-- Ignoring dependencies of the Default.z configuration-->
 <!-- Note: Equal to selection of Default.x above -->
 <configuration name="Admin.C07E02_DEMO_BWF"/>
<!-- Ignoring dependencies of
  configurations within the folder -->
 <configuration foldername="systemunits"/>
<!-- Including dependencies of all the
  configurations in the folder -->
 <configuration foldername="billing"
  resolveDependencies="true"/>
</configurations>
 </import>
```

- To import all the configurations that are included in a specific folder, include the following text in the XML file: `<configuration foldername="myFolder"/>`

  Note!

  If you specify configuration selections in the XML file that do not exist in the Export file, a warning is generated.

<workflows>

This XML tag enables you to use `systemimport [-select <xml-selection file>]` to import both workflow configurations and their CSV file in one action.
Use this tag to associate workflow tables with CSV data files.

Note!

- Set the `keepOld` attribute to true in order to prevent removal of workflow table data which has no match in the export file. Use false to overwrite the data. This parameter is only used during import, and has no effect during export.
- The `onError` attribute can either be set from the XML selection file, or from the systemimport in-line command. If set from both, the XML selection file attribute is the value that applies. For further information about the values that you can choose from, see `onError.` This parameter is only used during import, and has no effect during export.
- Set the `encryptPassword` attribute to the workflow configuration password if the workflow configuration is password protected.

Example. The XML selection tags

```
<import>
  <configurations>
  <configuration name="Common.DB_PROFILE"/>
  <configuration name="Common.APL_PROFILE"
  resolveDependencies="true"/>
  <configuration foldername="myFolder"/>
  </configurations>
  <workflows >
  <workflow name="Mobile.FTP_workflow"
  wfTable="/home/user1/FTP_workflows.csv"/>
  <workflow name="Mobile.SFTP2_workflow"
  wfTable="/home/user1/SFTP2_workflows.csv"
  resolveDependencies="true"/>
  <workflow name="Mobile.GGSN1_workflow"
  wfTable="/home/user1/GGSN1_workflows.csv"
  resolveDependencies="true"
  onError="ask"/>
  <workflow name="Mobile.GGSN3_workflow"
  wfTable="/home/user1/GGSN3_workflows.csv"
  resolveDependencies="true"
  onError="ignore"/>
  <workflow name="Mobile.GGSN4_workflow"
  wfTable="/home/user1/GGSN4_workflows.csv"
  resolveDependencies="true"
  onError="abort"/>
  <workflow name="Mobile.GGSN5_workflow"
  wfTable="/home/user1/GGSN5_workflows.csv"
  resolveDependencies="true"
  keepOld="false"
  encryptPassword="password"
  onError="ask"/>
  </workflows>
 </import>
```

[-dryrun]

Prior to importing, use systemimport with the dryrun switch to verify that the CSV data, such as number of columns or names, matches the contents of the workflow table. If a mismatch is detected a report will be generated.

[-onError]

If the onError attribute is not specified in the XML selection file, the value that you set it to in systemimport, is the value that applies. Otherwise, the attribute value applies. Set `[-onError]` to any of the following values:

- `ask`: to generate an interactive message
- `ignore`: to do nothing
- `abort`: to abort the command and stop a current configuration import

[-m|-message]

If you want to add a comment when making a systemimport, the `-m` or `-message` option can be used as in the following example:

Example - message

```
mzsh <username>/<password> systemimport -m "My Import" /home/Directory/<file to import>.zip
```

"My Import" will be the commented.

The comment will replace the default information saved when making a system import, and will both be included in the System Log message that is generated, as well as visible when selecting to view history in any of the configurations in the imported data.

[-u|-upgrade]

When exports have been made in a previous version of MediationZone, they may have to be upgraded. In this case you can use the `-u` or `-upgrade` option as in the following example:

Example - upgrade

```
$ mzsh <username>/<password> systemimport -u <file to import>.zip
```

The configurations will then be upgraded.

[-eu|-enableusers]

If you want imported users to be active after you run the import, use the `-eu` or `-enableusers` option as shown in the following example:

Example - enableusers

```
$ mzsh <username>/<password> systemimport -eu <file to import>.zip
```

By default users are imported as inactive and must be activated manually via the Access Controller. This option is only available when you run an import as a super user. If you use this option as another user, users will be skipped during the import.

<export file|directory>

Specify the path to the directory or ZIP file that contains the configurations you want to import.

[password]

To import encrypted configurations, provide a password.

Return Codes

Listed below are the different return codes for the `systemimport` command:

Code	Description
0	Will be returned if the command is successful.
1	Will be returned if the argument count is incorrect or argument(s) are invalid.
2	Will be returned if the command was unable to find an export (file|directory) at the supplied path.
3	Will be returned if the import could not be started due to locked import.
4	Will be returned if any errors were reported during the import.
5	Will be returned if the XML file did not contain any configurations or workflows to import.
8	Will be returned if the XML file did not contain any workflows to use in dry run.
10	Will be returned if the import results in invalid workflows due to compilation errors.
14	Will be returned if there are configurations not imported
20	Will be returned if the value for a supplied option or option is missing.
100	Will be returned if an error occurs while parsing the selection file
101	Will be returned if an error occurs while parsing a node in the selection XML file.
102	Will be returned if an error occurs while getting the attribute from a configuration tab, or if expected wfTable attribute in tag with the supplied name is missing.
103	Will be returned if the import was unable to parse the value for a booolean XML attribute.
104	Will be returned if an error occurs while parsing dependencies for a configuration.
300	Will be returned if an OutOfMemoryError occurs during import. Additional information from the "critical error log" will be included, this is further described in 2.12 Out of Memory Info in System Log in the System Administrator's Guide.

2.2.34 systeminsight

```
usage: systeminsight
profile create -folder <folder name> -name <profile name> -desc, --description <description of profile> -r, --
retentionPolicy <retention policy> |
remove [-f, --force] -profile <qualified profile name> |
addFilter [-f, --force] -profile <qualified profile name> -e, --regularExpression <"regular expression">, [-t,
--tagRegularExpressions <tag value regular expression>] |
removeFilter -profile <qualified profile name> -id <filter id> |
disable <qualified profile name> |
enable <qualified profile name> |
list -profile <qualified profile name> [-j,--json]
```

usage: `systeminsight metrics [-j, --json], [-m, --metric <metric name>], [-t, --tag <tag name>]`

usage: `systeminsight retentionPolicy list [ -j, --json ]`

usage: `systeminsight test [-n, --name <name of metric>] | [-t, --tags <tag name and value pairs>]`

This command enables you to manage a System Insight metrics, by adding and removing profiles and filters for the metrics that you want to produce. You can also use the command to list the metrics available on the running system on which you can apply filters, to list the retention policies in place, and test which filters there are for a metric.

The `systeminsight` command has four subcommands: `profile`, `metrics`, `retentionPolicy` and `test`.

profile

To use System Insight, you must determine the profile that you want to apply, which you configure using the `systeminsight profile` command. The subcommands described below enable you to manage a profile.

Note!

The System Insight service must be running before you create, modify or remove a profile. You start the service using the command `mzsh service start`. For information on how to configure System Insight services, see 2.2.1 Configure System Insight Services.

create

Use `systeminsight profile create` to create a System Insight profile. To create a profile, you must specify the folder in which you want to save the profile, the profile name, a description of the profile, and the retention policy for the profile.

The command accepts the following options:

Option	Description
-folder <name of folder>	Provide the name of the folder where you want to save the profile.
-name <name of profile>	Provide a unique name for the profile.
-desc, --description <description of profile>	Provide a description of the profile.
-r, --retentionPolicy <retention policy>	Use this option to specify the retention policy for the profile.
-enabled	Use this option to set the profile to enabled status so that you can use the filters in the profile. If a profile is not set to enabled, you cannot use the filters it contains to produce the required statistics.

Example of how to create a System Insight profile

```
systeminsight profile create -folder Default -name siprofile1 -desc siprofile1 -r one_month -enabled
```

Note!

After you have created a profile, the qualified profile name that is thereafter required when you use the `systeminsight` command to manage the profile consists of the folder name + the unique profile name: <folder name>.<profile name>, e g `Default.siprofile1`.

remove

Use `systeminsight profile remove` to delete a specific System Insight profile.

The command accepts the following options:

Option	Desription
-profile <qualified name of profile>	Provide the qualified name of the profile that you want to delete.
[-f, --force]	Use this option to forcibly remove a profile.

addFilter

Use `systeminsight profile addFilter` to add a filter to the filter list. To add a filter, you must specify the qualified profile name of the profile to which you want to add the filter, and the regular expression that you want to include in the filter. You can also add the tag value regular expressions that you want to include.

The command accepts the following options:

Option	Description
-profile <qualified name of profile>	Provide the qualified name of the profile to which you want to add the filter.
-e, --regularExpression "<regular expression>"	Provide the regular expression that you want to include in the filter, with quotation marks. A regular expression must begin with one of the following: `pico, host, mim, service` or `custom`, followed by ".", for example, "`host\.xyz`".
[-f, --force]	Use this option when the regular expression that you want to add does not begin with one of the following `pico, host, mim, service` or `custom`, followed by ".". For example a filter definition that would allow anything containing ".kafka.", "`.*\.kafka\..*`".
-t, --tagRegularExpressions "<tag value regular expression>"	Provide the regular expressions for tag values. For example, "`host_name=myhost[0-9]*`".

When you use this command to create a filter, it is given a filter id, determined by the numeric order of when the filter is created: the first filter is given filter id 1, the second id given filter id 2 etc.

Examples of how to add a filter

A simple rule:

```
systeminsight profile addFilter -profile Default.siprofile1 -e "host\.comp3\.comp3"
```

A rule which requires forced addition:

```
systeminsight profile addFilter -f -profile Default.siprofile1 -e ".*pico\.comp1\.comp1"
```

Example of a filter including tag value reglar expressions:

```
systeminsight profile addFilter -profile Default.siprofile1 -e "pico\..*" -t "host_name=myhost[0-9]*"
```

removeFilter

Use `systeminsight profile removeFilter` to delete a filter from the filter list in a profile. You must specify the profile that contains the filter, and the filter that you want to delete.

The command accepts the following options:

Option	Description
-profile <qualified name of profile>	Specify the qualified name of the profile that contains the filter that you want to delete.
-id <filter id>	Specify the filter that you want to delete.

disable

Use `systeminsight profile disable` to set a profile to disabled status. You must provide the qualified name of the profile that you want to disable.

enable

Use `systeminsight profile enable` to set a profile to enabled status. You must provide the qualified name of the profile that you want to enable. If a profile is not set to enabled, you cannot use the filters it contains to produce the required statistics.

list

Use `systeminsight profile list` to list all the profiles, or to list all the filters that have been added for a specific profile. You are also provided with the retention policy that applies to each profile and whether the profile is enabled or disabled.

> **Note!**
>
> The System Insight service must be running for InfluxDB retention policies to be included in the profile listing. You start the service using the command `mzsh service start`. For information on how to configure System Insight services, see 2.2.1 Configure System Insight Services.

The command accepts the following option:

Option	Description
-profile <qualified name of profile>	Specify the qualified name of the profile that contains the filters.
[-j, --json]	Use this option if you want to print the filters in json output format.

metrics

Use `systeminsight metrics` to list all of the metrics available on the running system, to which you can apply a filter in order to produce statistics using System Insight. Each metric is listed with the available tags and tag values.

The command accepts the following option:

Option	Description
[-j, --json]	Use this option if you want to print the metrics in json output format.
[-m, --metric]	Use this option if you want to refine the result based on the metric name as a regular expression. For example, if you want to see all the metrics with `pico` in the name: `systeminsight metrics -m "pico.*"`
[-t, --tag]	Use this option if you want to refine the result based on the tag name as a regular expression. For example, if you want to see all the tags with `host` in the name: `systeminsight metrics -t "host.*"`

The list of metrics displayed alphanumerically with the relevant tags and tag values depends on the running system. The following metrics, with the relevant tags always appear by default after you have created a profile. Since the tag values depend on your setup, the values listed below are examples.

Metrics	Tags	Tag Values
host.compute	host_name	<"host name">
	reporting_pico	<"ec1", "platform", "psc1", "sc1">
	reporting_pico_type	<"ec", "platform", "sc">
host.network	host_name	<"host name">
	nic	<"awdl0", "bridge0", "en0", "en1", "en2", "en3", "lo0", "p2p0">
	reporting_pico	<"ec1", "platform", "psc1", "sc1">
	reporting_pico_type	<"ec", "platform", "sc">
host.storage.iostats	host_name	<"host name">
	mount_dir	<"/">
	reporting_pico	<"ec1", "platform", "psc1", "sc1">
	reporting_pico_type	<"ec", "platform", "sc">
host.storage.swap	host_name	<"host name">
	reporting_pico	<"ec1", "platform", "psc1", "sc1">
	reporting_pico_type	<"ec", "platform", "sc">
host.storage.usage	host_name	<"host name">
	mount_dir	<"/", "/dev">
	reporting_pico	<"ec1", "platform", "psc1", "sc1">
	reporting_pico_type	<"ec", "platform", "sc">
pico.events	event_type	<"Code insertion", "Code removal">
	title	<"Code Server Package Event">
pico.jvm	host_name	<"host name">
	pico_instance	<"ec1", "platform", "sc_si1">
service.akka.router.receiver	actor_system	<"si">
	host_name	<"host name">
	pico_instance	<"ec1", "platform", "sc_si1">
service.akka.router.sender	actor_system	<"si">
	host_name	<"host name">
	pico_instance	<"ec1">
service.systeminsight.dispatcher	actor_system	<"si">
	host_name	<"host name">
	pico_instance	<"sc_si1">

As the output displayed when you use this command depends on the running system, the metrics available for the workflows which you are running based on the relevant MIMs are also listed.

Example - MIM metrics based on a running workflow

If you have a workflow with a Workflow Bridge processing agent, the metrics available may look as follows:

Metrics	Tags	Tag Values
`mim.realtime.processing.workflow_bridge`	`agent_category`	`<"processing">`
	`agent_name`	`<"Workflow_Bridge_1">`
	`host_name`	`<"host name">`
	`pico_instance`	`<"ec1">`
	`workflow_folder`	`<"Default">`
	`workflow_instance`	`<"wf">`
	`workflow_name`	`<"fwd">`
	`workflow_type`	`<"realtime">`

retentionPolicy

You can also list the retention policies.

Note!

The System Insight service must be running before the `systeminsight retentionPolicy` command is available. You start the service using the command `mzsh service start`. For information on how to configure System Insight services, see 2.2.1 Configure System Insight Services.

list

Use `systeminsight retentionPolicy list` to list all the retention policies in place.

The command accepts the following option:

Option	Description
[-j, --json]	Use this option if you want to print the list in json output format.

Example of retention policy list

```
systeminsight  retentionPolicy  list
Name           Duration
one_month      720h
one_day        24h
one_week       168h (Default)
three_months   2160h
one_year       8760h
ten_years      87600h
three_hours    3h
```

test

Use `systeminsight test` to get a list of the filters for a metric. Specifying a tag refines the metric, and is optional.

The command accepts the following options:

Option	Description
-n, --name "<name of metric>"	Provide the name of the metric for which you want to list the filters.
-t, --tags <tag name and tag value pairs>	You can provide the tag name and tag value pair(s) for which you want to list the filters.

Examples of how to test which filters match a metric

If you have the following four filters in place:

Filterid	Regular expression	Tag Regular Expression
1	custom\.system	
2	custom\.system	tag1=a.*
3	custom\.system	tag1=a.*, tag2=b.*
4	custom\.system	tag1=a.*, tag2=b.*, tag3=c.*

Example 1

If you test which filters match the metric name "custom.system" by entering the following:

```
systeminsight test -n custom.system
```

The output is as follows:

```
Profile: Default.siprofile1
FilterId      RegularExpression    Tag RegularExpressions
1             custom\.system       Map()
2             custom\.system       Map(tag1=a.*)
3             custom\.system       Map(tag1=a.*, tag2=b.*)
4             custom\.system       Map(tag1=a.*, tag2=b.*, tag3=c.*)
```

Example 2

If you test which filters match the metric name "custom.system" and the tag name and tag value pair tag1=aaa by entering the following:

```
systeminsight test -n custom.system -t tag1=aaa
```

The output is as follows:

```
Profile: Default.siprofile1
FilterId      RegularExpression    Tag RegularExpressions
1             custom\.system       Map()
2             custom\.system       Map(tag1=a.*)
```

Example 3

If you test which filters match the metric name "custom.system" and the tag name and tag value pairs tag1=aaa and tag2=bbb, by entering the following:

```
systeminsight test -n custom.system-t tag1=aaa tag2=bbb
```

The output is as follows:

```
Profile: Default.siprofile1
FilterId      RegularExpression    Tag RegularExpressions
1             custom\.system       Map()
2             custom\.system       Map(tag1=a.*)
3             custom\.system       Map(tag1=a.*, tag2=b.*)
```

Example 4

If you test which filters match the metric name "custom.system" and the tag name and tag value pairs tag1=aaa, tag2=bbb and tag3=ccc, by entering the following:

```
systeminsight test -n custom.system-t tag1=aaa tag2=bbb tag3=ccc
```

The output is as follows:

```
Profile: Default.siprofile1
FilterId     RegularExpression    Tag RegularExpressions
1            custom\.system       Map()
2            custom\.system       Map(tag1=a.*)
3            custom\.system       Map(tag1=a.*, tag2=b.*)
4            custom\.system       Map(tag1=a.*, tag2=b.*, tag3=c.*)
```

Return Codes

Listed below are the different return codes for the `systeminsight` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned when a general error occurs.
2	Will be returned if an unknown error occurs.
7	Will be returned if you are not on the Platform when you execute the command.
10	Will be returned if the parameters are incorrect.

2.2.35 udrview

```
usage: udrview [ -d <delimiter> ] [ -xml ] [ -z ] [ -f | -first <udr nr> ] [ -l | -last <udr nr> ] <decoder>
[file...]
```

`udrview` enables you to decode a file of UDRs and generate decoded content such as CSV or XML file.

A CSV file includes a table that consists of a header, where the UDR field names are specified, and rows of fields that comprise the UDRs. When using `udrview` you retrieve UDRs and UDR fields by specifying a UDR range, column numbers, and by using the Unix cut.

An input file may include different UDR types. In the generated output the first column is always the decoded UDR type. A new table header is generated for every decoded UDR type that is not identical to the preceding UDR type.

Example.

A header example:

```
1 [UDRType: Folder.UltraConfigName.DecoderName], 2 fieldNameA, 3 fieldNameB, 4 fieldNameC, ...
```

A decoded file example:

```
header udrType1, field_A, field_B
udrType1, 2, 3
udrType1, 5, 7
...
header udrType2, field_X, field_Y
udrType2, a, b
udrType2, a, c
...
header udrType1, field_A, field_B
udrType1, 45, 66
```

Option	Description
[-d <delimiter>]	Enter the character that should serve as a delimiter between the UDR fields. Default is a comma (,).
[-xml]	To generate the decoded output in the XML format specify xml in the command. Otherwise, CSV is the default output format. **Note!** For practical reasons, field types such as list and set cannot be included in a CSV file. To include these types in the output, use the <xml> parameter in `udrview`.

[-z]	To decode a GZIP compressed file specify its name.
[-f | -first <udr nr>]	Enter the number of the UDR in the input file that is the lowest range boundary of the UDRs that you want to decode. See example below.

[-l | -last <udr nr>]	Enter the number of the UDR in the input file that has the highest range boundry of the UDRs that you want to decode.
	
	Example.
	
	Consider the following input file:
	
	`1,Header`
	`2,222222,111111`
	`2,111111,444444`
	`2,444444,222222`
	`2,111111,333333`
	`3,Trailer`
	
	This file is comprised of a header, four UDRs with A-Number and B-Number, and a trailer.
	
	To generate selected decoded UDRs 2, 3, 4, and 5, run the following command:
	
	`$ mzsh mzadmin/<password> udrview Default.Ultra.fileDec asciifile.ascii -f 2 -l 5`
	
	UDRs in file: `asciifile.ascii`
	
	The output is:
	
	`1 [UDRType:Default.Ultra.udr],`
	`2 OriginalData,`
	`3 id,`
	`4 A_number,`
	`5 B_number`
	`[Default.Ultra.udr],0x32 ...`
	`0x0a,2,222222,111111`
	`[Default.Ultra.udr],0x32 ...`
	`0x0a,2,111111,444444`
	`[Default.Ultra.udr],0x32 ...`
	`0x0a,2,444444,222222`
	`[Default.Ultra.udr],0x32 ...`
	`0x0a,2,111111,333333`
	
	To generate only specific fields, run:
	
	`$ mzsh mzadmin/<password> udrview Default.Ultra.fileDec asciifile.ascii -f 2 -l 5 | cut -d, -f4,5`
	
	UDRs in file: `asciifile.ascii`
	
	The output is:
	
	`4 A_number,5 B_number`
	`222222,111111`
	`111111,444444`
	`444444,222222`
	`111111,333333`

[file...]	Either specify a file, or a list of files, as an input, or pipe an input from another command. **Example.** `$ cat /tmp/afile | mzsh mzadmin/<password> udrview Format.ultraname.decoder`

Return Codes

Listed below are the different return codes for the `udrview` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if a command line argument error occurs.
2	Will be returned if the input file is not found.
3	Will be returned if the decoder could not be created.
4	Will be returned if the range specified on command line is invalid.
5	Will be returned if the input file is not a valid Gzip file.
6	Will be returned if a decoding error occurs.

2.2.36 ultra

```
usage: ultra export <target-jar-file> | import <source-jar-file> |list [-v, --verbose] [-h --historic-only]
```

When you make changes to an ultra format, historic formats a stored on the system. However, these are not included when you perform a system export. You can use the `ultra` command to export both current and historic ultra formats from one system and import them as historic formats in another. This is useful when the target system must be able to handle e g persisted runtime data that is consistent with a previous version of an Ultra format.

Option	Description
[-v --verbose]	Use this option for detailed output from the `ultra` command.
[-h --historic-only]	Use this option to only include historic formats in the export. This option is not applicable for the `import` command.

export

Use the `export` command to write Ultra formats on the system to disk. These will be stored in a JAR file that contains the Ultra class definitions.

Example - Exporting Ultra formats

```
MZ>> ultra export /home/user/mz/ultra/ultraexport.jar
```

Note!

The export command cannot overwrite an existing export file.

Import

Use the `import` command to import Ultra formats from disk. For each Ultra format (class) in the specified JAR file, the command will perform the import if the format is not already present in the Code Server. If a format in the JAR file is historic or not does not matter during import, since it will be considered historic in the target system.

Example - Importing Ultra formats

```
MZ>> ultra import /home/user/mz/ultra/ultraexport.jar
```

List

Use the `list` command to list Ultra formats on the system.

Example. Listing Ultra formats

```
$ ultra list
```

Note!

If an empty list field disappears when the ascii encodes, you need to set a system property called *mz.ultra.terminator.backcomp*. If set to true, empty list fields do not disappear and backward compatibility is preserved from MediationZone version 8.1.7.0 and later versions.

To set this up, do the following:

- mzsh topo set [topo://container]:main1/pico:platform/val:config.properties.mz.ultra.terminator.backcomp true
- mzsh restart platform
- mzsh mzadmin/dr regenUltra
- Re-run the workflow

Return Codes

Listed below are the different return codes for the `ultra` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the command could not be interpreted, e g if a option that does not exist has been entered.
2	Will be returned if the input file is not found or if the output file already exists.

2.2.37 unregister

```
usage: unregister <unreachable ECSA>...
```

This command is used to unregister an unreachable ECSA. It can be useful when the ECSA has been unreachable for too long and/or if the ECSA should be moved to another host.

Return Codes

Listed below are the different return codes for the `unregister` command:

Code	Description
0	Will always be returned.

2.2.38 user

This command is used for scripted management of users. For instance, when using the systemimport command that imports all users by default as disabled, you need this command to enable the users again.

To know how use the *mzsh user* command, see below for input and output:

Input

```
mzsh mzadmin/dr help user
```

Output

```
usage: user <command> <command options>

 Commands:

   user enable <name>

   user disable <name>
```

> Specifying the [command] and [command options] is mandatory.

When running a command without specifying any arguments, an error is displayed immediately.

```
mzsh mzadmin/dr user

Invalid command was entered.

usage: user <command> <command options>

 Commands:

   user enable <name>

   user disable <name>
```

Return Codes

Listed below are the different return codes for the `user` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the argument count is incorrect.

2.2.39 vcexport

```
usage: vcexport [options]
```

Options

Options	Description
[-d, --directory]	Use this option to specify in which directory you want to place the exported data. If the directory does not exist, it will be created.
[es, --excludesysdata]	Use this option to exclude system data from the export. For example, you can exclude ECS related data, event categories, and workflow alarm values.
[-f, --folders]	Use this option to specify which folders you want to include in the export. For example, `mzsh <user name> /<password> vcexport -d MyDirectory/ -f Default Alarms` will export the configurations in the folders Default and Alarms to the directory MyFolder. If this option is not used, the configurations in all folders will be exported.
[-im, --includemeta]	Use this option to include meta data in the export. This will not be done by default, since not including meta data will make it easier to make a file compare between the exports.
[-o, --overwrite]	Use this option to enable existing exports in the stated directory to be overwritten.

This command exports configurations from MediationZone in a format that is adapted for version control systems. The `vcexport` command exports the configurations into a flat structure, i e with file extensions instead of directories. For each exported configuration, a `.xsd` file will be generated in which the structure of the data is stored, which will produce a more compact format than the other export commands can offer.

Return Codes

Listed below are the different return codes for the `vcexport` command:

Code	Description
0	Will be returned if the export was successful.
1	Will be returned if the command could not be interpreted, e g if a option that does not exist has been entered.
2	Will be returned if the export failed.
3	Will be returned if the folder you want to export to, stated with the -d, --directory option, could not be created.
5	Will be returned if the folder(s) stated, when using the -f, --folders option, does not exist.

2.2.40 vcimport

```
usage: vcimport [options]
```

This command imports exports made with the `vcexport` command.

Option	Description
[-d, --directory]	This option is mandatory and is used to state from which directory you want to import data.
[-y, --dryrun]	Use this option to parse import files without importing them.
[-f, --folders]	Use this option to specify which folders you want to include in the import. For example, `mzsh <user name> /<password> vcimport -d MyDirectory/ -f Default ECS` will import the configurations in the folders Default and ECS. If this option is not used, configurations in all folders will be included.
[-m, --message]	Use this option to add a message to identify the import.

Return Codes

Note!

If a key or name conflict occur, the imported data will not overwrite existing configurations.

Listed below are the different return codes for the `vcimport` command:

Code	Description
0	Will be returned if the import was successful.
1	Will be returned if the command could not be interpreted, e g if a option that does not exist has been entered.
2	Will be returned if the import failed.
3	Will be returned if the folder you want to import from, stated with the -d, --directory option, does not exist.
5	Will be returned if the folder(s) stated, when using the -f, --folders option, does not exist.

2.2.41 webdesktop

```
$ mzsh webdesktop [configuration file]
```

This command starts the Web Desktop Server, which is required to run the Desktop in a browser.

Example - Starting the Web Desktop Server with default settings

```
$ mzsh webdesktop
```

To start the Web Desktop Server with custom settings, you may pass a configuration file in the argument to the `mzsh webdesktop` command or set environment variables in the shell. The settings in the configuration file will override the values in the environment variables. For further information about these settings, see 14.1. Starting the Web Desktop Server.

Example - Example - Starting the Web Desktop Server with custom settings

```
$ mzsh webdesktop webdesktop.properties
```

Return Codes

This command runs continuously until interrupted and does not have any defined return codes.

2.2.42 wfcommand

```
usage: wfcommand <pattern matching expression for workflow name> <'Workflow' | workflow service | agent>
<instructions> ...
```

Debug

The command accepts wild cards, such as '*' and '?'. For further information see 3. Textual Pattern Matches.

To turn debug on for a workflow:

```
wfcommand <workflow> Workflow debug on
```

To turn the debug for a workflow off:

```
wfcommand <workflow> Workflow debug off
```

Supervision

By stating the Supervision service you can select to trigger and clear supervision actions with the `wfcommand` command.
If you want to use the wfcommand to trigger the Supervision service, you can append the following instructions:

- `manual` which has the following syntax:

  Usage: `manual [-action actionName actionParameters]`

  Simply entering manual will deactivate the Supervision Service configurations and switch to manual mode, which means that no conditions will be evaluated and no actions will be executed.

  When using the `-action` option, the syntax will be as follows:

  Usage: `manual -action overload ( <ratio> | -trigger | -clear ) [strategy]`

Option	Description
ratio	If you want to trigger overload and reject a specific ratio of incoming requests, e g reject every fifth request, you can enter the ratio number here. 2 equals every second request, 5 equals every fifth request, etc.
-trigger	Use this option if you want to trigger overload for everything. An event will then be sent out for monitoring purposes. This will disable the automatic Supervision configuration.
-clear	Use this option if you want to clear overload for everything. Any Diameter/Radius overload protection strategies will be cleared and no requests will be rejected. The automatic Supervision configuration will then be disabled. **Note!** Using this option will not enable the automatic Supervision configuration.
[strategy]	Use this option to state a specific Diameter/Radius strategy, e g Diameter_AbortSessionRequest, Diameter_CCEventRequest, etc. See the Desktop User's Guide for further information about these strategies.

- `automatic` is used to revert to automatic Supervision configuration.

When switching from automatic to manual mode, all active overload protection will remain active. However, in the opposite direction, when switching from manual to automatic mode, all active overload protection will be reset and the appropriate overload protection will be set by the automatic rules.

Each time you switch between manual an automatic mode, an entry will be logged in the System Log for monitoring purposes.

Example.

To trigger overload protection for a workflow:

```
wfcommand <workflow> "Supervision" manual -action overload -trigger
```

An event will then be sent out for monitoring purposes.

To clear overload protection for a workflow:

```
$ wfcommand <workflow> "Supervision" manual -action overload -clear
```

To reject every fifth Diameter AbortSessionRequest in a workflow:

```
$ wfcommand <workflow> "Supervision" manual -action overload 5 Diameter_AbortSessionRequest
```

To switch back to automatic mode:

```
$ wfcommand <workflow> "Supervision" automatic
```

See the Desktop User's Guide for further information about Supervision Service, Diameter, Radius, and overload.

Aggregation and Diameter agents

The Aggregation and Diameter agents allows the user to interact with the flow of data via the wfcommand. For further information, see 9.3 Aggregation Agent and 9.16 Diameter Agents in the Desktop User's Guide.

GTP agent

The GTP agent allows the user to view the MIM counters via the wfcommand. For further information, see 9.37 GTP' Agent in the Desktop user's guide.

The request counters and the timestamp, which are published as MIM values are also possible to view from the commandline tool.

Syntax to run the wfcommand function in mzsh:
`mzsh wfcommand <Workflow Name> <Agent Name> printcounters`

Parameters:

`Workflow Name`	Name of the workflow which contains the GTP agent.
`Agent Name`	Name of the GTP agent in the workflow.

Example. Executing the wfcommand

```
$ mzsh mzadmin/dr wfcommand "Default.myWorkflow" "GTP_1"
printcounters
Default.myWorkflow (3):
Data Record Count: 138
Cancel Data Count: 2
Message Error Count: 0
Possible Duplicate Count: 5
Release Data Count: 1
Out of Sequence Count: 0
Duplicate Message Count: 0
Redirect Count: 0
Last Request Timestamp: Thu Mar 07 14:38:12 CET 2013
```

Return Codes

Listed below are the different return codes for the wfcommand command:

Code	Description
1	Will be returned if command line arguments are missing.
2	Will be returned if no matching workflows can be found, or if the workflow is not running.
3	Will be returned if the target agent cannot be found, or if the agent is not accepting/supporting the command.

2.2.43 wfdebug

```
usage: wfdebug <pattern matching expression for workflow names> ... <ON|OFF>
```

This command is used to enable or disable debug information for a workflow. Note that turning on the Debug mode might slow down the workflow due to log filing.

Return Codes

Listed below are the different return codes for the `unregister` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the argument count is incorrect.

2.2.44 wfdisable

```
usage: wfdisable <pattern match expression for workflow names> ...
```

This command disables one or more workflows.

With this command you compare a single pattern matching expression, or several, with the full workflow name, <folder>.<workflowconfigurationname>.<workflowname>, of all the workflows.

The command accepts wild cards, such as '*' and '?'. For further information see 3. Textual Pattern Matches.

Return Codes

Listed below are the different return codes for the `wfdisable` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the argument count is incorrect.
2	Will be returned if the user is not found or not logged in.
3	Will be returned if no matching workflow was found.

2.2.45 wfenable

```
usage: wfenable <pattern match expression for workflow names> ...
```

This command enables one or more workflows.

With this command you compare a single pattern match expression, or several, with the full workflow name, `<folder>.<workflowconfigurationname>.<workflowname>`, of all the workflows.

The command accepts standard wild cards, such as '*' and '?'. For further information see 3. Textual Pattern Matches.

Return Codes

Listed below are the different return codes for the wfenable command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the argument count is incorrect.
2	Will be returned if the user is not found or not logged in.
3	Will be returned if no matching workflow was found.

2.2.46 wfexport

```
usage: wfexport <workflow configuration> <export file> [-csv|-tsv|-ssv] [workflow configuration password]
```

This command creates a file, (CSV, TSV, or SSV), of the data that is stored in the Workflow Table. This file contains a header row that lists the names of the Workflow Table columns.

> **Note!**
>
> `wfexport` is identical to the Workflow Table right-click command Export Table. For further information see Export Table in the Desktop User's Guide.

`<workflow configuration>`	Specify the workflow configuration that you want to export.
`<export file>`	Specify the name of the export file
[-csv|-tsv|-ssv]	Specify one of the wfexport supported formats according to which the export file should be created: • CSV (Comma Separated Value) - Default format • SSV (Semicolon Separated Value) • TSV (Tab Separated Value) **Note!** • Text strings within each value are delimited by a quotation mark ("). • In the export file, External References are enclosed in braces ({}) preceded by a dollar symbol ($). For example: `${mywf_abcd}`. For further information see External References in the Desktop User's Guide. • Similarly, Execution Settings in the export file, are enclosed in braces ({}), but are preceded by a pound symbol (#). For example: `#{mywf_exsettings}`. For further information see Execution Settings below. • To prevent a workflow table column from being updated by the export file data when importing, delete that same column from the export file.
[workflow configuration password]	For an encrypted export file, provide a password.

Example - General use of wfexport

Create the file `wf_disk_collection.csv` under the `tmp` directory.

```
$ wfexport Default.disk_collection /tmp/wf_disk_collection
```

The `wf_disk_collection.csv` export file:

```
"ID","Name","[Disk_1]Directory","[Disk_1]Filename"
 1,"workflow_1","/tmp/in","in.file"
 3,"workflow_3","/tmp/in3","in3.file"
 100,"workflow_100","/tmp/in100","in100.file"
 109,"workflow_110","/tmp/in101","in101.file"
 110,"workflow_110","/tmp/in101","in101.file"
 111,"workflow_112","/tmp/in112","${a}"
 112,"workflow_113","/tmp/in113","a"
```

Execution Settings

An export file of a workflow configuration may include settings for an EC or an ECSA.

Setting	Description	Valid Values
`type`	The configuration type.	`execsettings`
`enabled`	Specifies whether or not the configuration is enabled	`true` or `false`
`disttype`	The workflows load balancing method.	`sequential`, `wfcount`, `machineload, or roundrobin`. For further information see the Desktop User's Guide, in the Workflow Properties section.
`ectype`	The Execution Context type on the eclist list.	`ec` or `ecsa`
`eclist`	A vertical bar (|) delimited string of the configured ECs and ECSAs. The list is enclosed with brackets. **Note!** • Only one ECSA at a time can be specified. • `[]` = an empty eclist.	

Example - Export data of EC/ECSA configurations

- With an EC configuration, the export file will include:

```
#{type=execsettings#enabled=true
    #disttype=wfcount#ectype=ec#eclist=[ec1 | ec2]}
```

- With an ECSA configuration, the export file will include:

```
#{type=execsettings#enabled=true
    #disttype=sequential#ectype=ecsa#eclist=[ecsa1]}
```

- With a disabled EC configuration, the export file will include:

```
#{type=execsettings#enabled=false}
```

Return Codes

Listed below are the different return codes for the wfexport command:

Code	Description
0	Will be returned if the command was successful.
1->	Number of errors that has occurred during the export.

2.2.47 wfgroupdisable

```
usage: wfgroupdisable <pattern match expression for workflow group names> ...
[ -mode < a >]
```

This command disables one or more workflow groups.

With this command you compare a single pattern matching expression, or several, with the full workflow group name, `<folder>.<workflowgroupconfigurationname>.<workflowgroupname>`, of all the workflow groups.

The command accepts wild cards, such as '*' and '?'. For further information see 3. Textual Pattern Matches.

The command accepts the following options:

Option	Description
[-mode < a >]	Disable only workflow groups marked with a specified mode: • a - Only Autostart groups are disabled.

Return Codes

Listed below are the different return codes for the `wfgroupdisable` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the argument count is incorrect.
2	Will be returned if the user is not found or not logged in.
3	Will be returned if no matching workflow group was found.

2.2.48 wfgroupenable

```
usage: wfgroupenable <pattern match expression for workflow group names> ...
[ -mode < a >]
```

This command enables one or more workflow groups.

With this command you compare a single pattern match expression, or several, with the full workflow group name, `<folder>.<workflowgroupconfigurationname>.<workflowgroupname>`, of all the workflow groups.

The command accepts standard wild cards, such as '*' and '?'. For further information see 3. Textual Pattern Matches.

The command accepts the following options:

Option	Description
[-mode < a >]	Enable only workflow groups marked with a specified mode: • a - Only Autostart groups are enabled.

Return Codes

Listed below are the different return codes for the `wfgroupenable` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the argument count is incorrect.
2	Will be returned if the user is not found or not logged in.
3	Will be returned if no matching workflow group was found.

2.2.49 wfgrouplist

```
usage: wfgrouplist <pattern match expression for workflow group names> ...
[ -valid ] [ -invalid ] [ -active ] [ -inactive ]  [ -scheduled ] [ -unscheduled ] [ -mode < D | E >] [ -short
]
```

Lists the groups that are configured in the MediationZone system. If no option is used, the list consist of four columns: Workflow Group Names, State, Mode and Scheduled (true/false).

With this command you compare a single pattern match expression, or several, with the full workflow group name, `<folder>.<groupconfigurationname>.<workflowgroupname>`, of all the workflows.

The command accepts wild cards, such as '*' and '?'. For further information see 3. Textual Pattern Matches.

The command accepts the following options:

Option	Description
[-valid]	Lists all valid groups.
[-invalid]	Lists all invalid groups.
[-active]	Lists all active groups.
[-inactive]	Lists all inactive groups.
[-scheduled]	Lists all the workflow groups that are scheduled.
[-unscheduled]	Lists all the workflow groups that are not scheduled.
[-mode <D|E>]	Lists only workflow groups marked with a specified mode: • D - Disabled • E - Enabled
[-short]	If set only the Workflow Group Names will be listed.

Return Codes

Listed below are the different return codes for the `wfgrouplist` command:

Code	Description
0	Will be returned if the command was successful.
13	Will be returned if the arguments are not recognized.

2.2.50 wfgroupmodify

```
usage: wfgroupmodify -group NAME (-memberwf WF | -membergrp GRP) [-prereqwf WF...] [-prereqgrp GRP...] [-q]
```

This command enables you to modify the prerequisites list for an entry in a workflow group.

The workflow group name and workflow group member name are required parameters. Only one workflow group member at a time can be modified by the command.

Parameters/Options	Description
<group NAME>	Enter the name of the workflow group which prerequisites you want to modify.
(-memberwf WF | -membergrp GRP)	Enter the name of the member that you want to modify, either a workflow or a workflow group.
[-prereqwf WF...]	Enter a list of workflows that the prerequisites list should include. **Note!** Leaving this parameter empty sets the workflow prerequisites list of the workflow group to be empty.
[-prereqgrp GRP...]	Enter a list of workflow groups that the prerequisites list should include. **Note!** Leaving this parameter empty sets the workflow-groups prerequisites list of the workflow group to be empty.
[-q]	Quiet mode. Use this parameter to eliminate the display of any report during execution.

Return Codes

Listed below are the different return codes for the `wfgroupmodify` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the command was unsuccessful.

2.2.51 wfgroupstart

```
usage: wfgroupstart <pattern match expression for workflow group names> ... [-w <timeout s>] [-b]
```

Starts one or more workflow groups.

With this command, you compare a single pattern match expression, or several, with the full workflow group name, <folder>.<groupconfigurationname>.<workflowgroupname>, of all the workflows.

The command accepts wild cards, such as '*' and '?'. For further information see 3. Textual Pattern Matches.

To start the workflow group instantly it should be in the `Idle` state. Otherwise, start occurs once processing is finished and the group is back in `Idle` state.

A workflow group in `Invalid` state can not be started as the configuration is invalid.

If the wfgroupstart command is used in combination with any of the options when the Platform is restarted, MediationZone cannot retain the workflow group state, so the `wfgroupstart` command will immediate return the exit code 101 - 'Platform down group aborted'

Option	Description
[-w]	Use this option to wait for workflow group completion, that is wait for whichever comes first of either a timeout or received exit code declaring the status of the workflow; completed, aborted etc. For further information about exit codes, see Appendix 1 . **Note!** The [-w] option does only allow *one* workflow group to be started at the time.
[-b]	Use this option (block) to wait for the return code that indicates that workflow group has for example completed, aborted or another code. For further information about exit codes, see Appendix 1. **Note!** The [-b] option only allows *one* workflow group to be started at the time. The [-w] option has precedence over the [-b] option. If both are used at the same time the [-w] will be active.

Return Codes

Listed below are the different return codes for the `wfgroupstart` command:

Code	Description
0	Will be returned if the command was successful.
50	Will be returned if the number of arguments is incorrect.
51	Will be returned if an error occurred when parsing arguments.
60	Will be returned if the command line timed out.
70	Will be returned if the group was not found.
71	Will be returned when trying to start more than one group, or if there was an exception due to no user being logged in.
80	Will be returned if there was an exception due to no group being started
90	Will be returned if an unexpected error occurred.
230	Will be returned if the workflow is already running.
231	Will be returned if permission is denied (no execute permission).
232	Will be returned if the group does not exist.
240	Will be returned if the group is invalid.

2.2.52 wfgroupaddwf

```
usage: wfgroupaddwf <workflow group name> <pattern match expression for workflow names>
```

This command adds one or more workflows to an existing workflow group.

With this command you compare a single pattern match expression, or several, with the full workflow group name, `<folder>.<workflowconfigurationname>.<workflowname>`, of all the workflows.

For further information about pattern match expressions see 3. Textual Pattern Matches.

A workflow that is already a member of the workflow group is skipped by the command.

Return Codes

Listed below are the different return codes for the `wfgroupaddwf` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the number of arguments is incorrect.
2	Will be returned if the group is not found.
3	Will be returned if the workflow(s) cannot be found.
4	Will be returned if there is no connection to Mgmt_Utils.
5	Will be returned if the group is locked.
6	Will be returned if the updating of group data failed.
7	Will be returned if the workflow type is not supported.
8	Will be returned if the configuration lock could not be released.

2.2.52 wfgroupstop

```
usage: wfgroupstop [ -immediate ] <pattern match expression for workflow group name>...
```

Stops one or more workflow groups. The workflow group does not stop instantly, but rather waits for all the workflow group members to stop running, before the whole group is fully stopped and return to the `Idle` state.

With this command you compare a single pattern match expression, or several, with the full workflow group name, <folder>.<groupconfigurationname>.<workflowgroupname>, of all the workflows.

The command accepts wild cards, such as '*' and '?'. For further information see 3. Textual Pattern Matches.

Option	Description
[-immediate]	When this option is used, the workflow group is stopped without waiting for a batch to finish.

Return Codes

Listed below are the different return codes for the `wfgroupstop` command:

Code	Decsription
0->	Will indicate the number of failed group stops or the number of arguments that are incorrect.
0	Will be returned if the command is successful.
1	Will be returned if no user is logged in.

2.2.54 wfgroupaddwfgroup

```
usage: wfgroupaddwfgroup <workflow group name> <pattern match expression for workflow names>
```

This command adds one or more workflow groups to an existing workflow group.

Note!

A workflow group that is already a member of the workflow group is skipped by the command.

Similarly, you can not add a workflow group to itself.

Example - Adding all the workflow groups with names beginning with wf_grp_, but skips wf_grp_1

```
MZ>>wfgoupaddwfgroup <wf_grp_1> <wf_grp_*>
```

With this command you compare a single pattern match expression, or several, with the full workflow group name, `<folder>.<workflowgroupconfigurationname>.<workflowgroupname>`, of all the workflows.

For further information about pattern match expressions see 3. Textual Pattern Matches.

Return Codes

Listed below are the different return codes for the `wfgroupaddwfgroup` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the number of arguments is incorrect.
2	Will be returned if the group is not found.
3	Will be returned if the workflow(s) cannot be found.
4	Will be returned if there is no connection to Mgmt_Utils.
5	Will be returned if the group is locked.
6	Will be returned if the updating of group data failed.
7	Will be returned if the workflow type is not supported.
8	Will be returned if the configuration lock could not be released.

2.2.55 wfgroupremovewf

```
usage: wfgroupremovewf <workflow group name> <pattern match expression for workflow names>
```

This command removes one or more workflows from a workflow group.

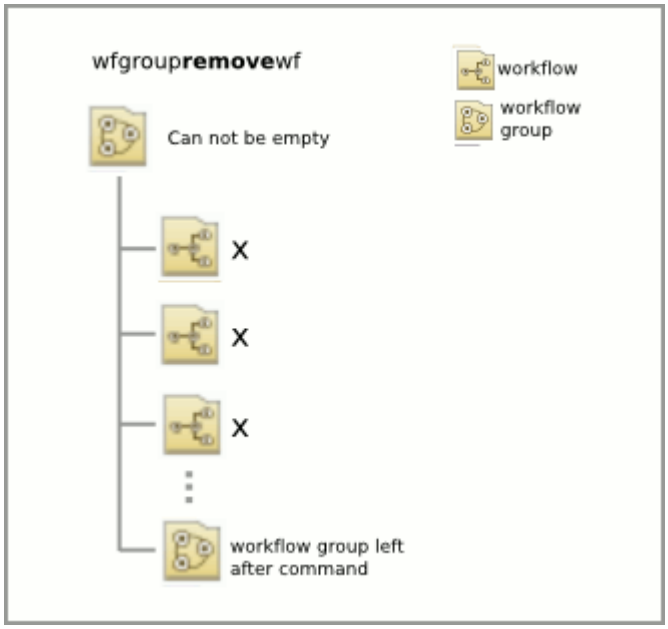
With this command you compare a single pattern match expression, or several, with the full workflow name, `<folder>.<workflowconfigurationname>.<workflowname>`, of all the workflows.

For further information about pattern match expression see 3. Textual Pattern Matches.

Note!

With `wfgroupremovewf` you cannot remove all the workflows from a group as this will result in an invalid workflow group configuration. In that case, the command aborts and an error message informs you about the abort cause.

As the workflow group should not be emptied, the command enables you to remove all the workflows from the workflow group only if the workflow group also contains a workflow group. This way, after removing all the workflows, the parent workflow group is not empty. See the figure below, The wfgroupremovewf Command.



The wfgroupremovewf Command

Return Codes

Listed below are the different return codes for the `wfgroupremovewf` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the number of arguments is incorrect.
2	Will be returned if the group is not found.
3	Will be returned if the workflow(s) you want to remove cannot be found.
4	Will be returned if there is no connection to Mgmt_Utils.
5	Will be returned if the group is locked.
6	Will be returned if the updating of group data failed.
7	Will be returned if the configuration lock could not be released.
8	Will be returned if the group is empty (all members have been removed).

2.2.56 wfgroupremovewfgroup

```
usage: wfgroupremovewfgroup <workflow group name> <pattern match expression for workflow group names>
```

This command removes one or more workflow groups from a workflow group.

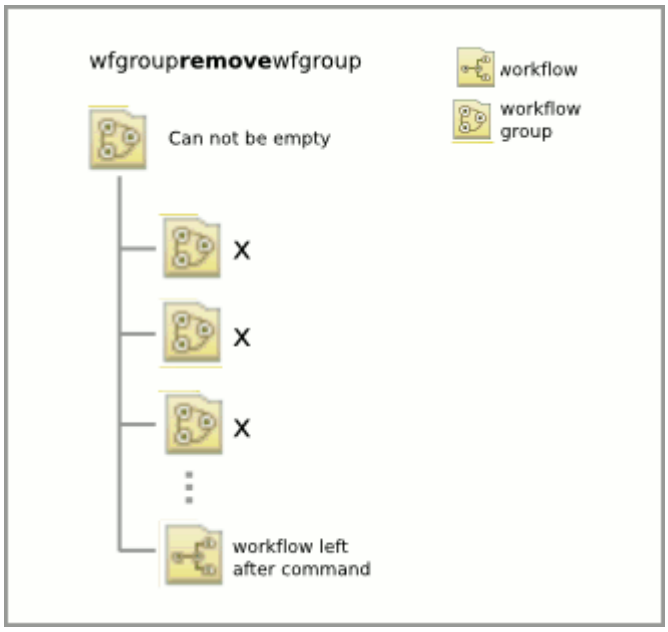
With this command you compare a single pattern match expression, or several, with the full workflow group name, `<folder>.<workflowgroupconfigurationname>.<workflowgroupname>`, of all the workflow groups.

For further information about pattern match expression see 3. Textual Pattern Matches.

Note!

With `wfgroupremovewfgroup` you cannot remove all the workflow groups from a workflow group, as this will result in an invalid workflow group configuration. In that case, the command aborts and an error message informs you about the abort cause.

As the workflow group should not be emptied, the command enables you to remove all the workflow groups from the parent workflow group only if the workflow group also contains a workflow. This way, after removing all the workflow groups, the parent workflow group is not empty. See the figure below, The wfgroupremovewfgroup Command.



The wfgroupremovewfgroup Command

Return Codes

Listed below are the different return codes for the `wfgroupremovewfgroup` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the number of arguments is incorrect.
2	Will be returned if the group is not found.
3	Will be returned if the workflow(s) you want to remove cannot be found.
4	Will be returned if there is no connection to Mgmt_Utils.
5	Will be returned if the group is locked.
6	Will be returned if the updating of group data failed.
7	Will be returned if the configuration lock could not be released.
8	Will be returned if the group is empty (all members have been removed).

2.2.57 wfimport

```
usage: wfimport [-keepOld [yes|no]] <workflow configuration> <export file> [workflow configuration password]
```

This command updates the specified workflow configuration by importing workflows that are defined in the export file.

Example - wfimport

`The workflow configuration Default.disk_collection` workflow configuration is updated with with imported data from the file `wf_disk_collection.csv`.

```
$ wfimport Default.disk_collection wf_disk_collection.csv
```

Note!

`wfimport` and the right-click command `Import Table` are different. If the workflow configuration consists of several workflows, and `wf_disk_collection.csv` contains a single workflow configuration data, only one workflow is included in the Workflow Table after `wfimport`. To avoid losing your configurations when importing, use the right-click command `Import Table` in the Workflow Table.

For further information see `Import Table` in the Desktop User's Guide.

Option/Parameter	
[-keepOld [yes|no]]	Specify `yes` to preserve workflow table data that is not updated by the export file. Specify `no` to have such data removed form the workflow table.
<workflow configuration>	The workflow configuration that you want to be updated by the export file.
<export file>	The export file name. For further information about the export file see 2.2.46 wfexport. **Note!** • `wfimport` imports the file formats: CSV (Comma Separated Value), SSV (Semicolon Separated Value), and TSV (Tab Separated Value). • Text strings within each value are delimited by a quotation mark ("). • Exported fields that contain profiles, are assigned with a unique string identifier. The ID and Name fields are exported, as we
[workflow configuration password]	To import a password protected workflow configuration, specify a password.

Return Codes

Listed below are the different return codes for the `wfimport` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the number of arguments is incorrect.
1	Will be returned If login credentials are incorrect.
1	Will be returned if configuration permission is denied.
2	Will be returned if the import file does not exist.
3	Will be returned if the import file cannot be read (read permission).
2	Will be returned if the import file directory does not exist.
5	Will be returned if the import file has an incorrect file suffix.
6	Will be returned if the configuration name is incorrect
7	Will be returned if the configuration does not exist.
8	Will be returned if the configuration is already locked.
9	Will be returned if an encryption passphrase is needed.
11	Will be returned if the configuration could not be loaded.
12	Will be returned if the import fails (see the logs).

2.2.58 wflist

```
usage: wflist <pattern match expression for workflow names>
[ -invalid | -valid | -active | -inactive ] [ -not-member-of-wfgroup ] [ -long [ -activationMode]] [ -loop [ N
]]
```

This command lists workflows. If you use the command without any options, all workflows will be listed.

With this command you compare a single pattern match expression, or several, with the full workflow name, <folder>.<workflowconfigurationname>.<workflowname>, of all the workflows.

The command accepts wild cards, such as '*' and '?'. For further information see 3. Textual Pattern Matches.

The command accepts the following options:

Option	Description
[-invalid]	Lists all invalid workflows.
[-valid]	Lists all valid workflows.
[-active]	Lists all active workflows.
[-inactive]	Lists all inactive workflows.
[-not-member-of-wfgroup]	Lists all the workflows that are not included in a workflow group.
[-long]	Lists the workflow state and information about the latest start (aborted or not started). If the workflow is running `long` reveals how long it has been running. `long` will also display the current throughput.
[-long [-activationMode]]	Lists the workflow state and information about the latest start as well as the workflow's activation mode. If the workflow is enabled it will be shown as ENABLED.
[-loop [N]]	Sets a loop mode that runs the `wflist` command forever every N seconds (N is by default set to 10) To terminate loop use Ctrl-C. **Note!** When combining [-active] and [-long], the Execution Context and the execution time for the workflow will be listed.

Return Codes

Listed below are the different return codes for the `wflist` command:

Code	Description
0	Will be returned if the command was successful.
1	Will be returned if the number of arguments is incorrect.

2.2.59 wfstart

```
usage: wfstart <pattern match expression for workflow names>... [-w <timeout s>] [-b]
```

This command executes one or more workflows.

> **Note!**
>
> If the workflow(s) is/are in building state when running the `wfstart` command, "Configuration is building." will be displayed and the workflow(s) will not start unless you have used any of the -w or -b options described below.

With this command you compare a single pattern match expression, or several, with the full workflow name, `<folder>.<workflowconfigurationname>.<workflowname>`, of all the workflows.

The command accepts standard wild cards, such as '*' and '?'. For further information see 3. Textual Pattern Matches.

> **Example - Activate all workflows in the folder myFolder starting with the letter D.**
>
> ```
> MZ>> mzsh  wfstart myFolder.D*
> ```

Only workflows in `Idle` or `Waiting` state will be started. If the workflow is in another state, an error message will be shown stating why it did not start.

Option	Description
[-w]	Use this option to wait for workflow completion, that is wait for whichever comes first of either a timeout or received exit code declaring the status of the workflow; completed, aborted etc. For further information about exit codes, see Appendix 1 . If the workflow is in building state, "Configuration is building." will be displayed, and the workflow will not be started until finished building. A timeout may then occur if the building takes too long. **Note!** The [-w] option does only allow *one* workflow to be started at the time.
[-b]	Use this option (block) to wait for the return code that indicates that workflow has for example completed, aborted or another code. For further information about exit codes, see Appendix 1 . If the workflow is in building state, "Configuration is building." will be displayed, and the workflow will not be started until finished building. **Note!** The [-b] option only allows *one* workflow to be started at the time. The [-w] option has precedence over the [-b] option. If both are used at the same time the [-w] will be active.

Note!

If the connection to the Platform is down the `mzsh` command will continue to run. The command will however not receive any exit codes until the Platform is up again. Timeouts will still occur. If the workflow is aborted or finished while the Platform was down, the correct exit code will be returned as soon as the Platform is up again.

If a connection problem with the EC occurs during the workflow execution causing the workflow to become unreachable the `mzsh` command will wait for the workflow to either complete or abort. If workflows do not reconnect and are automatically reachable again the workflow must be handled manually. The standard MediationZone alarm functionality can be used to detect unreachable workflows.

Return Codes

Listed below are the different return codes for the `wfstart` command:

Code	Description
0	Will be returned if the command was successful.
50	Will be returned if the argument count is incorrect.
51	Will be returned if the argument parse fails.
70	Will be returned if the workflow is not found.*
80	Will be returned if no user is logged in.
90	Will be returned if an unexpected error occurred.
230	Will be returned if the workflow is already running.
231	Will be returned if permission is denied (no execute permission).
232	Will be returned if the workflow does not exist.*
240	Will be returned if the workflow is invalid

* The difference between return code 70 and 232 is that 70 is returned if a workflow is not found during validation of parameters, and 232 is returned if a workflow is not found when trying to start the workflow. In theory, the workflow could be deleted during the extremely short time between the validation and the start, but in practice it will happen very rarely.

2.2.60 wfstop

```
usage: wfstop [ -immediate ] <pattern match expression for workflow names> ...
```

This command stops one or more workflows.

With this command you compare a single pattern match expression, or several, with the full workflow name, `<folder>.<workflowconfigurationname>.<workflowname>`, of all the workflows.

The command accepts wild cards, such as '*' and '?'. For further information see 3. Textual Pattern Matches.

The command does not kill all processes, but rather sends a message to all the agents to stop.

If the workflow cannot be stopped, an error message will be shown stating why it was not stopped.

The command accepts the following option:

Option	Description
[-immediate]	The workflow is stopped without waiting for a batch to finish.

Return Codes

Listed below are the different return codes for the `wfstop` command:

Code	Description
0	Will be returned if the command was successful or if the argument count is incorrect.
1	Will be returned if no user is logged in.
1->	The number of non-matching workflow names as parameters, or the number of failed workflow stops.

3. Textual Pattern Matches

In resemblance to Regular Expressions, when searching through text strings of names and other textual patterns in mzsh, there are two characters that help you filter text according to a certain criteria:

- The asterisk '*' is a wildcard for one or more characters.
- The question mark '?' is a wildcard for any single character.

Note!

If you want to use the '*' and '?' wildcards when you are not logged in, the wildcards have to either be enclosed with single or double quotation marks, or preceded with a backslash '\'.

For example:

`mzsh mzadmin/dr wfgrouplist \*` will work.

`mzsh mzadmin/dr wfgrouplist "*"` will work.

`mzsh mzadmin/dr wfgrouplist * -mode D` will not work.

The period '.' mark is not a wildcard and is treated as any character.

4. Executing Shell Commands When OS Level Access Is not Available

MediationZone has some capabilities and activities that can only be performed with OS level access, either through direct shell access or through a shared folder, for example execution of `mzsh` commands or handling of External References property files.

Using the APL Shell Script Execution Function `ScriptExec` can address most needs for shell commands that normally would require OS access.

The `ScriptExec` function is executed as a part of the normal workflow execution principles so it can be used by any user who has be granted permissions to design workflow logic and execute workflows.

This is done in 3 steps:

1. Create the *script* that contains the command and its parameters. The *script* must match any required dependencies in the filesystem.
2. Create a workflow that uses the `ScriptExec` function to call the *script*, run it, and capture the execution events.
3. Execute the workflow and analyse the debug output for any errors.

When successful, the command is executed in the same way as when regular shell command is been used. Any errors and information about the execution are shown in the workflow debug events.

Appendix 1

Exit Code	Message	Info
0	Run completed.	
50	<Prints the help command>	No parameters to the command.
51	<varying, error description>	Error parsing the parameters.
60	Timed out.	Workflow/group did not completed/aborted within the defined timeout.
70	No matching workflow.	Workflow/group not found. If wfgroupstart is used the message will be: No matching group found.
71	Only one workflow is allowed to be started when option -w is used.	Workflow is replaced with workflowgroup if wfgroupstart is used. [-w] is replaced with [-b] in case it is used instead.
80	<varying, depending on why it didn't start>	Reason why the workflow/group did not start.
90	Unexpected error: <error message + stack trace>	Unexpected error, abort command.
100	Aborted.	
101	Platform shutdown, group aborted.	
110	Unknown status of run.	Command has been out of contact with platform and lacks information about how the run completed.
230	Already active (ignoring request).	
231	Permission denied.	
232	No such configuration.	
240	Configuration is invalid.	