Opswise Automation Center 5.1

Command Line Interface Reference

Command Line Interface Reference

- Introduction
- Required Parameters
    - Using the Configuration File Method
    - Using Command Parameters
    - Using Environment Variables
- Optional Parameters
- Return Codes
- Using the Command Line Interface in z/OS
    - z/OS-Specific Syntax Requirements
    - Methods for Issuing Opswise Commands in z/OS
- Table of Commands

Introduction

The Opswise command line interface (CLI) is implemented as a set of commands that perform specific actions in an Opswise Controller. The CLI commands can execute on any system that has TCP/IP connectivity to the Opswise Transport. The results of the action are printed to the CLI commands standard output.

On UNIX and Windows, the CLI commands are included with the outboard components. The commands are installed in the following directory:

UNIX	`/OUTBOARD_HOME/bin`
Windows	`DRIVE:\OUTBOARD_HOME\bin`

Replace OUTBOARD_HOME with the directory in which the outboard components are installed and, for Windows, replace DRIVE with the drive they are installed on.

Add the appropriate directory for your O/S to your PATH environment variable in order to execute the commands without specifying their full path name.

On z/OS, the CLI is implemented with a single program, UAGCMDZ. The program UAGCMDZ with alias OPSUCMDZ is installed in the SUNVLOAD library.

Required Parameters

The CLI commands can execute on any system that has TCP/IP connectivity to the Opswise Transport. The following information is require for the commands to connect to the Transport and logon to an Opswise Controller:

- IP address or host name and port number for the Opswise transport
- Login ID and password for Opswise Controller with the appropriate rights

Several methods are available for providing the required information:

- Using the configuration file command switch to pass the values in a file
- Using command parameters
- Using environment variables

Config File	Command Switch	Variable	Description

`network.transports`	`-t [transporter]`	`TSPNAME`	Port number @ machine name (host name of IP address) where the Opswise transporter is running. You can view this information from the browser interface as follows: 1. From the navigation pane, select **Automation Center Resources > Connectors**. 2. Locate the transporter that was specified in the network configuration for this agent. Click on the transporter name to display its details. Among the details provided on this screen are the Host Name, IP Address, and Port for the transporter. Use the format port@host to provide this information. Example: `4803@localhost`
`network.core`	`-n [hubname]`	`HUBNAME`	Queue name for the message hub. You can view this information from the browser interface as follows: 1. From the navigation pane, select **Automation Center Resources > Connectors**. 2. Locate the QUEUE NAME for the message hub. The default is HUB01.
`security.userid`	`-u [userid]`		User ID that the CLI utility will use to log into Opswise. Must be a valid user with the appropriate permissions defined in the Opswise database using the Opswise Security module.
`security.password`	`-p [password]`		Password that the CLI utility will use to log into Opswise. Must be a valid password for this user.

Each is described further below.

Using the Configuration File Method

Shown below is the command syntax if you use a configuration file to pass the required information into your script or command.

```
[command] -c [filename] [command parameters]
```

The configuration file can have any name. It must exist in the directory from where you are issuing the commands. Shown below are the required parameters and sample values of a configuration file:

```
network.transports=4803@localhost
network.core=HUB01
security.userid=ops.admin
security.password=o
```

Using Command Parameters

Shown below is the syntax with sample values if you choose to provide the required information using command parameters:

```
-t 4803@localhost
-n HUB01
-u ops.admin
-p o
```

Using Environment Variables

You can set the following environment variables to set the core and transporter names:

```
TSPNAME
HUBNAME
```

Optional Parameters

Optional Parameter	Description
-h | -?	Help.
-o=*n*	Timeout. Value *n* is the number of seconds you want to allow before the system returns a timeout.

Return Codes

Shown below are return codes you may get in response to your command:

Return Code	Description
0	Completed successfully.
1	Configuration error (incorrect command line or configuration file options).
2	Not enough memory.

Using the Command Line Interface in z/OS

All Opswise commands are supported in the z/OS environment. They are managed and executed by the program UAGCMDZ with alias OPSCMDZ, which may execute as either a command processor or a standard z/OS batch job. OPSCMDZ is installed with the z/OS agent in library SUNVLOAD.

z/OS-Specific Syntax Requirements

The commands and syntax requirements for the z/OS environment are very similar to the other supported platforms with a few exceptions described below.

Command Line Options

When OPSCMDZ is executed as a batch program, command line options are specified with the step PARM keyword. For example:

```
 //STEP01   EXEC PGM=OPSCMDZ,
 //  PARM='ops-agent-status -c dd:CMDOPTS agent-type=windows'
```

Configuration File Syntax

If you use the Opswise -c flag that specifies the configuration file, you must specify the file name using one of the formats below.

ddname Format

```
-c dd:ddname
```

where *ddname* references a DD statement that exists in your batch JCL.

Data Set Name Format

```
-c "//'dsname'"
```

where *dsname* is a fully-qualified data set name which may be a partitioned data set with a member name.

Line Numbers in Configuration File

Do not place line numbers in columns 73-80. The entire 80 byte record is read and processed as input.

Methods for Issuing Opswise Commands in z/OS

Three methods are available for running OPSCMDZ and executing Opswise commands:

- Using a batch job
- Under a TSO session
- Using a batch TSO Terminal Monitor Program (TMP)

Each of these methods is described below. In each example where data sets are specified, we use:

UNV.SUNVLOAD	Library containing the CLI programs UAGCMDZ and OPSCMDZ alias.
USER.PARM	Data set containing command options.
USER.REXX	Data set containing user REXX EXECs.

Running Opswise Commands in a Batch Job

The CLI program OPSCMDZ executes a single Opswise command as a JCL batch job step. If multiple commands are to be executed, run each one as an individual job step.

The following example illustrates running OPSCMDZ as a batch job to request the status of all Windows agents.

```
//jobname  JOB (acctg-info),'your name',MSGCLASS=X,MSGLEVEL=(1,1),
//             CLASS=A,NOTIFY=&SYSUID
//*
//STEP01   EXEC PGM=OPSCMDZ,
// PARM='ops-agent-status -c dd:CMDOPTS agent-type=windows'
//STEPLIB  DD DISP=SHR,DSN=UNV.SUNVLOAD
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//CMDOPTS  DD DSN=USER.PARM(CMDOPTS),DISP=SHR
//CEEDUMP  DD SYSOUT=*
```

Here is a break-down of this example:

- The step PARM value specifies the CLI command and it options.
- The job step can include only one Opswise command.
- The STEPLIB points to the z/OS Agent load library.
- The command output will be in SYSPRINT.
- CMDOPTS is a DDNAME that references the location of the command parameters.

Issuing Commands under TSO

You can also issue Opswise commands under a TSO session.

The sample command string below shows an Opswise command issued from an ISPF Command Shell prompt:

```
Enter TSO or Workstation commands below:
===> opscmdz ops-agent-status -c "//'USER.PARM(CMDOPTS)'" agent-type=windows
```

The above command will return data similar to the sample below:

```
Agentname                        AgentType     Status
agent-sys00101                   Windows       Active
agent-sys00201                   Windows       Offline
opscmd-complete
```

Issuing Commands as a Batch TSO

The following two examples show how to issue an Opswise command in a batch TSO.

Example One

```
//CMDZBAT  JOB (acctg-info),'your name',MSGCLASS=X,MSGLEVEL=(1,1),
//              CLASS=A,NOTIFY=&SYSUID,TIME=5
//*
//STEP01   EXEC PGM=IKJEFT01,DYNAMNBR=200,REGION=40M
//STEPLIB  DD DISP=SHR,DSN=UNV.SUNVLOAD
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//CMDOPTS  DD DSN=USER.PARM(CMDOPTS),DISP=SHR
//SYSTSIN DD *
opscmdz ops-agent-status -c dd:CMDOPTS agent-type=windows
/*
```

Output will be in SYSTSPRT.

Example Two

```
//REXXAGNT JOB (acctg-info),'your name',MSGCLASS=X,MSGLEVEL=(1,1),
//              CLASS=A,NOTIFY=&SYSUID,TIME=5
//*
//STEP01   EXEC PGM=IKJEFT01,DYNAMNBR=200,REGION=40M
//STEPLIB  DD DISP=SHR,DSN=UNV.SUNVLOAD
//SYSEXEC  DD DSN=USER.REXX,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//CMDOPTS  DD DSN=USER.PARM(CMDOPTS),DISP=SHR
//SYSTSIN DD *
  %OUTTRAP1
/*
```

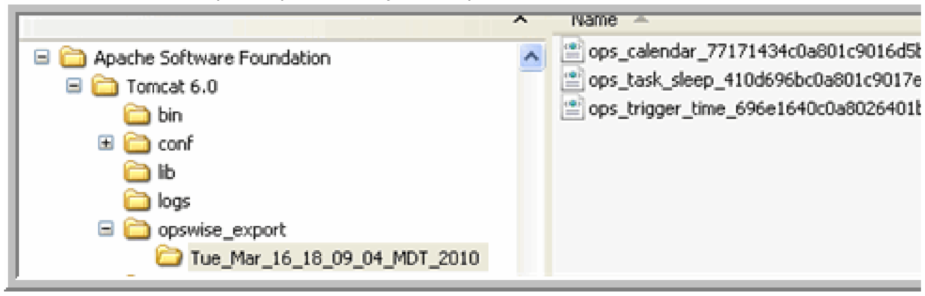
The following REXX EXEC must be located in USER.REXX(OUTTRAP1):

```
/***************************** REXX **********************************/
/* Using OUTTRAP to                                                  */
/*          (1) Obtain z/OS agent status                             */
/*          (2) Test status from the command response                */
/*          (3) Launch a task if the agent status is Active          */
/*********************************************************************/
x = OUTTRAP('OPS.')
opscmdz "ops-agent-status -c dd:CMDOPTS agent-type=z/OS"
SAY 'RC is:' RC
SAY OPS.0 'records were read.'
launch = 'NO'
DO i = 1 to OPS.0 WHILE launch = 'NO'
 IF SUBSTR(OPS.i,61,6) = 'Active' THEN
  DO
   launch = 'YES'
   opscmdz "ops-task-launch -c dd:CMDOPTS task-name=DUMPT"
  END
END
DO i = 1 to OPS.0
  SAY OPS.i
END
y = OUTTRAP('OFF')
```

Table of Commands

The following table provides a quick reference for all supported commands, including command syntax and description. Commands are listed in alphabetical order.

Command Syntax	Description
`ops-agent-status [agent-name= agent-type= options=]`	Lists the status of one or more agents. Command switches: • agent-name = Optional. The name or partial name of one or more agents. Wildcards supported. • agent-type = Not case sensitive. One of the following: WINDOWS, UNIX, LINUX, LINUX/UNIX, ZOS or Z/OS. • options=v Display verbose results. Example: `ops-agent-status -c config.txt agent-type=windows`
`ops-connector-status [connector-name= options=]`	Lists the status of one or more connectors. Command switches: • connector-name = Optional. The name or partial name of one or more connectors. This must be the name from the Connector Name field on the Connectors list. Wildcards supported. • options=v Display verbose results. Example: `ops-connector-status -c config.txt connector-name=*HUB01`

`ops-export-bulk [none]`	Performs a bulk export of all records in your Opswise database. This command creates a separate XML file for each record type in the following directory: `\$tomcat dir$\webapps\opswise\WEB-INF\plugins\com.opswise \backup\unload` Example: `ops-export-bulk -c config.txt`
`ops-export-trigger [trigger-type= trigger-name=]`	Performs an export of the specified trigger(s) and all the associated record(s). For example, if you export one trigger, the utility also exports the calendar used in the trigger and all tasks launched by the trigger. The utility creates a separate XML file for each record type and generates the output directory and filenames using a date and timestamp, plus an internal record identifier generated by the system. The exported directory and its data is created in the following directory: `\$tomcat dir$\opswise_export` The above directory is configurable in **Configuration > Properties > Export Path**. Shown below is a sample output directory and exported data:  Apache Software Foundation / Tomcat 6.0 / bin / conf / lib / logs / opswise_export / Tue_Mar_16_18_09_04_MDT_2010; Name: ops_calendar_77171434c0a801c9016d5b, ops_task_sleep_410d696bc0a801c9017e, ops_trigger_time_696e1640c0a8026401b Command switches: • trigger-name = Wildcards supported. The name of one or more triggers. • trigger-type = Optional; not case sensitive. If you used a wildcard in the trigger-name parameter, you can use this parameter to narrow down the selection to a specific type of trigger. Allowable values are CRON, TIME, FILE_TRIGGER, TEMPORARY, TASK_MONITOR, MANUAL, APPLICATION_MONITOR. The following example exports all triggers whose name begins with "SF": `ops-export-trigger -c c.cfg trigger-name=SF*`
`ops-import-bulk [none]`	Imports into your Opswise database the contents of the following directory: `\$tomcat dir$\webapps\opswise\WEB-INF\plugins\com.opswise \backup\unload` The data being imported must have been exported using the ops-export-bulk command. Example: `ops-import-bulk -c config.txt`

`ops-import-trigger [import-file=]`	Performs an import into your Opswise database the trigger records contained in the specified file. Command switch: • import-file=Required. The filename created when you ran the export-trigger command. The file name is created automatically. You must supply the filename but the command is hardcoded to look in the following directory: `\$tomcat dir$\webapps\opswise\WEB-INF\plugins\com.opswise \backup\unload`
`ops-manual-setcompleted [task-instance=]`	For a Manual task, sets its status to Success. Command switch: • task-instance= An internal Opswise identifier. Example: `$ ./ops-manual-setcompleted -c c.cfg task-instance=8fda93dcd861e5e4005bf48e2cad6206`
`ops-manual-setstarted [task-instance=]`	For a Manual task, changes its status from Action Required to Started. This allows you to acknowledge the Manual task and indicate that the manual procedures have been started. Command switch: • task-instance= An internal Opswise identifier. Example: `$ ./ops-manual-setstarted -c c.cfg task-instance=8fda93dcd861e5e4005bf48e2cad6206`
`ops-task-cancel [task-instance=]`	Cancels the specified Task instance. Click here for a description of behavior and restrictions. Command switches: task-instance= An internal Opswise identifier.
`ops-task-forcefinish [task-instance=]`	Force finishes the specified Task instance. Click here for a description of behavior and restrictions. Command switches: • task-instance= An internal Opswise identifier.
`ops-task-hold [task-instance=]`	Places the specified Task instance on hold. Click here for a description of behavior and restrictions. Command switches: • task-instance= An internal Opswise identifier. Example: `$ ./ops-task-hold -c c.cfg task-instance=90079020d861e5e40128cbb3cdbe1cf3`

`ops-task-launch [task-type= task-name= task-variables=]`	Launches the specified task(s). Wildcards supported. Command switches: • task-type = Not case sensitive. One of the following: SLEEP, WNDOWS, UNIX, ZOS, FILE_MONITOR, MANUAL, EMAIL, FTP, SQL, FTP_FILE_MONITOR, TASK_MONITOR, STORED_PROCEDURE, WORKFLOW, SAP, SYSTEM_MONITOR, INDESCA, APPLICATION_CONTROL. • task-name = Required. A valid task name. • task-variables = Optional. Any variables specified in the task that need a value to run properly. All **variable=value** pairs must be specified within one set of curly braces, as shown in this example: `task-variables={variable1=first value variable2=second value}`
`ops-task-list [task-name= task-type=]`	Displays all the specified tasks. Command switches: • task-name = Required. Wildcards supported. The name or partial name of one or more tasks. • task-type = Not case sensitive. One of the following: SLEEP, WNDOWS, UNIX, ZOS, FILE_MONITOR, MANUAL, EMAIL, FTP, SQL, FTP_FILE_MONITOR, TASK_MONITOR, STORED_PROCEDURE, WORKFLOW, SAP, SYSTEM_MONITOR, INDESCA,APPLICATION_CONTROL.
`ops-task-release [task-instance=]`	Releases the specified Task instance from hold. Click here for a description of behavior and restrictions. Command switches: • task-instance= An internal Opswise identifier. Example: `$ ./ops-task-release -c c.cfg task-instance=90079020d861e5e40128cbb3cdbe1cf3`
`ops-task-rerun [task-instance=]`	Reruns the specified Task instance. Click here for a description of behavior and restrictions. Command switches: • task-instance= An internal Opswise identifier.
`ops-task-setpriority [task-instance= priority=]`	For agent-based tasks (Windows, Linux/Unix, or z/OS) in a status of Started, changes the priority on the specified task instance. Click here for a description of behavior and restrictions. Command switches: • task-instance= An internal Opswise identifier. • priority= One of the following: high, medium, low.
`ops-task-skip [task-instance=]`	Skips the specified Task instance. Click here for a description of behavior and restrictions. Command switches: • task-instance= An internal Opswise identifier. Example: `$ ./ops-task-skip -c c.cfg task-instance=90079026d861e5e400bba81913a4fdd0`

`ops-task-status [task-name= task-type= task-status= options=]`	Displays the status of all task instance(s) associated with the specified task. Command switches: • task-name = Required. Wildcards supported. The name or partial name of one or more tasks. • task-type = Optional; not case sensitive. One of the following: SLEEP, WNDOWS, UNIX, ZOS, FILE_MONITOR, MANUAL, EMAIL, FTP, SQL, FTP_FILE_MONITOR, TASK_MONITOR, STORED_PROCEDURE, WORKFLOW, SAP, SYSTEM_MONITOR, INDESCA, APPLICATION_CONTROL. • task-status = Optional; not case sensitive. Options: Any valid task status. • options=v Return verbose results Example: `$ ./ops-task-status -c c.cfg task-name=mantask`
`ops-trigger-disable  [trigger-name= trigger-type=]`	Disables the specified trigger(s). Command switches: • trigger-name = Wildcards supported. The name or partial name of one or more triggers. • trigger-type = Optional; not case-sensitive. If you used a wildcard in the trigger-name parameter, you can use this parameter to narrow down the selection to a specific type of trigger. Allowable values are CRON, TIME, FILE_TRIGGER, TEMPORARY, TASK_MONITOR, MANUAL, APPLICATION_MONITOR.
`ops-trigger-enable [trigger-name= trigger-type=]`	Enables the specified trigger(s). Command switches: • trigger-name = Wildcards supported. The name or partial name of one or more triggers. • trigger-type = Optional; not case sensitive. If you used a wildcard in the trigger-name parameter, you can use this parameter to narrow down the selection to a specific type of trigger. Allowable values are CRON, TIME, FILE_TRIGGER, TEMPORARY, TASK_MONITOR, MANUAL, APPLICATION_MONITOR.
`ops-trigger-now [trigger-name= trigger-type= trigger-variables=]`	Marks all conditions satisfied in the specified trigger(s) and launches its associated tasks. Command switches: • trigger-name = Required. Wildcards supported. The name or partial name of one or more triggers. • trigger-type = Optional; not case sensitive. If you used a wildcard in the trigger-name parameter, you can use this parameter to narrow down the selection to a specific type of trigger. Allowable values are CRON, TIME, FILE_TRIGGER, TEMPORARY, TASK_MONITOR, MANUAL, APPLICATION_MONITOR. • trigger-variables = Optional. Any variables specified in the task(s) being triggered that need a value to run properly. All variable value pairs must be specified within one set of curly braces, as shown in this example: `trigger-variables={variable1=first value variable2=second value}`
`ops-trigger-status [trigger-name= trigger-type= trigger-status= options=]`	Lists the status of the specified trigger(s). Possible statuses are **Enabled** or **Disabled**. Command switches: • trigger-name = Wildcards supported. The name or partial name of one or more triggers. • trigger-type = Optional; not case sensitive. One of the following: CRON, TIME, FILE_TRIGGER, TEMPORARY, TASK_MONITOR, MANUAL, APPLICATION_MONITOR. • trigger-status = Optional; not case sensitive. One of the following: ENABLED, DISABLED. • options=v Return verbose results.

`ops-variable-list [variable-name= variable-scope= task-name= trigger-name= options=]`	Lists the specified variable(s). Command switches: • variable-name = Wildcards supported. The name or partial name of one or more variables. • variable-scope = One of the following: global (default), local (task & trigger), task, and trigger. Defines the type of variable being listed. A global variable is created independently by selecting **Variables** from the Opswise navigation pane. A trigger variable is attached to a trigger; a task variable is attached to a task. Local can be either a task or trigger variable. • trigger-name = If you specify trigger in the scope parameter, you must specify a single trigger name. • task-name = If you specify task in the scope parameter, you must specify a single task na • options=v Return verbose results. Example: `$ ./ops-variable-list -c c.cfg variable-name=abc variable-scope=global`
`ops-variable-set [variable-name= variable-scope= variable-value= variable-description= task-name= trigger-name= create=]`	Sets the specified variable. Command switches: • variable-name = The name of a specific variable. The name must begin with an alphabetic character and can consist of: alphas (a-z, A-Z), numerics 0-9, _ (underscore). White spaces are not permitted; names are not case-sensitive. • variable-value = A string that will become the value of the specified variable. UTF-8 character set supported. • variable-description = Optional; provides a description of the variable. • variable-scope = global (default), task or trigger. In cases where there may be more than one occurrence of the specified variable, this parameter defines which one should be set. A global variable is created independently by selecting **Variables** from the Opswise navigation pane. A trigger variable is attached to a trigger; a task variable is attached to a task. • create = yes or no. A switch indicating whether or not a new variable may be created for this request. • trigger-name = If you specify trigger in the scope parameter, you must specify a single trigger name. • task-name = If you specify task in the scope parameter, you must specify a single task na Example: `$ ./ops-variable-set -c c.cfg variable-name=myvar1 variable-value=mydata1 create=yes`