Cisco Workload Automation Command Line Program Guide

Version 6.3

First Published: August, 2015
Last Updated: September 23, 2016

Contents

1

Introduction

Overview

A hidden aspect of CWA is its command line program. The *sacmd.cmd* (Windows) or *sacmd.sh* (Unix/Linux) program provides access to CWA through the MS-DOS command prompt. The Unix version of CWA uses a similar program called **tesmcmd**. The command line program is generally done from the CWA Web client but you can also use the *sacmd.cmd* (Windows) or *sacmd.sh* (Unix/Linux) program on a master machine. You can automate CWA job control functions by including commands in scripts or by embedding job control functions in the code running your company processes.

This chapter covers these topics:

Installing the Command Line Program

Instructions for installing using the Windows installation or from the Linux/Unix command line are below:

Windows Installation

To install the Command Line program for Windows:

1. Go to cisco.com and on the Cisco Workload Automation Product Downloads page, locate and download the *Cisco Workload Automation CommandLine.msi* file for either 32-bit or 64-bit.
2. Run the .msi file.
3. On the Welcome screen, click **Next**.

The **Destination Folder** panel displays.

4. Click **Next** to install to the default folder location or click **Change** to choose a different location.

   The **Ready to Install the Program** panel displays.

5. Click **Install** to begin the installation.

   When the setup is complete, the **Setup Completed** panel displays.

6. Click **Finish** to complete the installation.

Linux/Unix Installation

To install the Command Line program from Console mode:

1. Open a command prompt window by selecting from the **Start** menu, **Programs>Command Prompt**.
2. Enter `$ ./install.bin -i console`.
3. Press ENTER to continue the installation.

   The **Choose Install Folder** screen displays.

4. Press ENTER to select the default location.

   -or-

   Enter an absolute path to the appropriate location, then press ENTER.

5. At the `IS THIS CORRECT?` prompt, enter `Y` if the path for the install folder is correct or `N` if it is incorrect.
6. Press **ENTER**.

   The **Pre-Installation Summary** screen displays.

7. After reviewing the installation information, press ENTER to begin the installation.

   The **Installing** screen displays.

8. When the installation is complete, press ENTER to exit the installer.

Activating the Command Line Program

Before you can use the Command Line Program, you must connect to the Client Manager DSP.

To connect to the Client Manager DSP:

1. Verify that the Master and Client Manager are running.
2. Locate the path to the `sacmd` command and enter the following information:

```
sacmd -cmdspurl http://hostname:8080/api/tes-6.0 -user username -pass password
```

Using Job Aliases

The command-line interface to CWA allows you to refer to jobs by their job ID number or the job alias. Job aliases are set automatically to the unique job ID number whenever you create a job definition. You can edit the job alias to a name of your own choosing on the **Options** tab in the **Job Definition** dialog; however, the job alias must be unique and the **Job Alias** field cannot be left blank. Many commands allow you to refer to a job or group using the alias name or job ID, but not the full job or group name.

Job aliases must be between one and eight characters in length. Do not use spaces in job aliases.

Note: You can use the listrule command to discover the job alias, as well as other job rule data, for your jobs and job groups.

Scripts and Batch Files

The *sacmd.cmd* (Windows) or *sacmd.sh* (Unix/Linux) program includes the **`file`** command that reads commands from a file, letting you batch-process a group of commands.

You can add comments to your file by preceding the comment with two forward slashes (**//**) or the sharp sign (**#**). You can use spaces or tabs in front of the comment character (**//** or **#**), but otherwise it must be the first character in a line.

Lines that contain only spaces or tabs are ignored. You can add clarity to your script or batch file by using blank lines to separate different sections of your file.

Running the Command Line from CWA

The **Command** file can be defined in a job definition along with any necessary command parameters. Commands can be run from CWA as jobs if the user:

- is a valid CWA user. (A valid CWA user can login to the CWA Web client. A user who is only a runtime user is not a valid CWA user.)
- either uses Windows passwords.

  -or-

  runs the agent as a user.
- has the CWA access right to a particular command.

Command files are defined from the **Program** tab of the **Job Definition** dialog. The command is entered in the **Command** field and any parameters for the command are entered in the **Parameters** field.

Information for accessing the DSP should precede the command line command in the **Command Parameters** field.

Command Modes

There are two modes for entering commands from the command line program, Single and Multiple Command.

Note: Before using sacmd <command>, you can set -persist to save the given URL, user and password in an encrypted file in the user's home directory, so that next time sacmd is started, the URL, user and password will not have to be specified. For further information on command arguments, refer to the help by running sacmd -help from the bin directory of the Command Line Program home.

Single Command Mode

Using this mode, type **`SACmd`** before each command. After entering the command, you return to the MS-DOS prompt.

To enter a single command:

1. Open a command prompt window by selecting **Start>Programs>Accessories>Command Prompt**.
2. Change directories by entering the following text (if you used the default location for files during installation) after the prompt:

```
cd /d C:\Program Files\TIDAL\TESCmdLine\bin
```

Press the ENTER key and your prompt becomes the following:

```
C:\Program Files\TIDAL\TESCmdLine\bin>
```

3. Enter **SACmd** and then the command and press the ENTER key. The command executes and you are returned to the prompt.

```
C:\Program Files\TIDAL\TESCmdLine\bin>
```

For example,

```
C:\Program Files\TIDAL\TESCmdLine\bin\>SACmd status -i 8
```

Multiple Command Mode

Using the multiple command mode, you can enter CWA commands without typing **SACmd** each time.

To enter multiple command mode:

1. Open a command prompt window by selecting **Start>Programs>Accessories>Command Prompt**.
2. Change directories by entering the following (if you used the default location for files during installation) after the **C:\>** prompt:

```
cd /d C:\Program Files\TIDAL\TESCmdLine\bin
```

Press the **Enter** key and your prompt becomes the following:

```
C:\Program Files\TIDAL\TESCmdLine\bin>
```

3. Type **SACmd** and press the ENTER key again.

The prompt changes to **SACmd>** and remains **SACmd>** until you exit. You are now in program's multiple command mode. You can enter CWA commands without preceding them with **SACmd**. For example, entering a command from this mode looks like this:

```
SACmd>status -i 8
```

To exit, enter **EXIT** and press the ENTER key.

Command Line Interface

Note: Verify the Master and Client Manager are up and running before using the Command Line program.

Note: Before using sacmd <command>, you can set -persist to save the given URL, user and password in an encrypted file in the user's home directory, so that next time saCmd is started, the URL, user and password will not have to be specified. For further information on on command arguments, refer to the help by running sacmd -help from the bin directory of the Command Line Program home.

Note: Once you have entered a SACmd session, you can enter help for an understanding of available commands. For example, SACmd>help.

To start the CWA command line interface:

1. Enter **SACmd** at an MSDOS command prompt.

The general format for using **SACmd** is as follows:

```
SACmd command -option1 Argument1 -option2 "Argument 2"
```

where you provide the command, the options and the arguments. For example:

```
SACmd addrule -n alfa -G
```

where **addrule** is the command, **n** and **G** are options, **alfa** is the argument for **n**, and **G** has no argument.

Command Line Syntax

Command syntax is listed for each command.

- Any argument not enclosed in brackets is required.
- When options are enclosed in square brackets, that is, **[**and**]**, they are optional and not required.
- If options are separated with a vertical bar, that is, **|**, either one or the other option must be used.

The argument for an option may be required or optional, depending on the command. If you do use an argument, it must follow the option it belongs to.

Options and arguments are case sensitive, but the command is not. If an argument has a space, enclose the argument in quotes. For example,

```
SACmd addrule -G -a clock -h "Windows Agent"
```

In the above example, **"Windows Agent"** is the argument to the **-h** option. **Windows Agent** was enclosed in quotes because it contains a space.

The order of the arguments is usually not important. If arguments do need to be given in a specific order, it is stated in the text.

Warning: Always enclose arguments containing spaces in quotes, or the command will not execute successfully. For example, if the Program Files folder is included in a path statement as an argument, the entire path must be enclosed by quotes.

Text Conventions

Each listed command may have several sections to explain different aspects of the command.

- **Syntax** – Displays how to enter the command. The syntax section uses certain punctuation conventions to denote characteristics of the command options as explained in the prior Command Line Syntax, page 12 section.
- **Options** – Explains each parameter for the command.
- **Operation** – Amplifies aspects of command behavior that may not be readily apparent.
- **Examples** – Displays some ways to use the command.

Job Status ID Cross Reference

Usually, when specifying job status in a command, you either specify the job status itself, or you specify its job status ID. For example:

```
SACmd jobset -i 3245 -s "Completed Abnormally"
```

Caution: When specifying the job status, any status containing two or more words must be in quotes. All arguments including spaces must be enclosed in quotes, or your command will not execute successfully.

It may be easier to specify a job status using a numerical value when you know the status that the numerical value corresponds to. For example, the above command can be typed as:

```
SACmd jobset -i 3245 -s 103
```

The following table cross-references each job status as shown in the CWA Web client with their internal job status number. You can also obtain this information using the **`liststat`** command.

Job Status	Job Status ID
Scheduled	0
Waiting on Dependencies	1
Waiting on Operator	2
Held	3
Timed Out for Day	5
Agent Unavailable	7
Agent Disabled	8
Agent Outage	9
Waiting on Group	10
Waiting on Children	11
Cancel Pending	12
Waiting on Resource	49
Launched	50
Active	51
Stopped	52
Deferred	53
Error Occurred	66
Completed Normally (Gathering Output)	97
Completed Abnormally (Gathering Output)	98
Externally Defined (Gathering Output)	99
Completed	100
Completed Normally	101
Completed Abnormally	103
Skipped	104
Orphaned	105
Aborted	106
Externally Defined	107
Timed Out	108
Cancelled	109
Preorphan	110
Cancelled Normally	112

Command Summary

The tables in the following sections list all available commands and their descriptions:

Master Commands

Command Name	Description
compile	Compiles the production schedule.
status	Displays the master's status.
resume	Resumes the production schedule after being paused.
pause	Pauses the production schedule. No waiting jobs will run, even if their dependencies are met.
jobadd	Inserts a job with the specified job rule **`jobid`** into the production schedule. Note that this command refers to the ID of the job rule and not to the ID of the job occurrence, as in the other commands.
jobcancel	Cancels or aborts the job with the specified **`jobid`**.
jobremove	Removes a job from the production schedule.
jobhold	Holds a waiting job or stops an **`active`** job with the specified **`jobid`**.
jobgo	Overrides the dependencies of the job with the specified **`jobid`**, and immediately runs the job.
jobrelease	Releases a job that is in the **`Waiting on Operator`** status, or resumes a job that has been waiting or has **`Stopped`**.
jobrerun	Reruns the specified **`jobid`**.
jobset	Sets the completion status of the specified job.
submit	Inserts the specified job into production.

Individual Job Occurrence Commands

Command Name	Description
grpupd	Updates inherit attributes for jobs in the specified group. You can obtain the group's job run ID by using the **jobmon** command.
jobadd	Inserts a job with the specified job rule **jobid** into the production schedule. Note that this command refers to the ID of the job rule and not to the ID of the job occurrence, as in the other commands.
jobcancel	Cancels or aborts the job with the specified **jobid**.
jobremove	Removes a job from the production schedule.
jobhold	Holds a waiting job or stops an **active** job with the specified **jobid**.
jobgo	Overrides the dependencies of the job with the specified **jobid**, and immediately runs the job.
jobmod	Modifies a job occurrence. You can obtain the job run ID by using the **jobmon** command.
jobrelease	Releases a job that is in the **Waiting on Operator** status, or resumes a job that has been waiting or has **Stopped**.
jobrerun	Reruns the specified **jobid**.
jobset	Sets the completion status of the specified job.
submit	Inserts the specified job into production.

Production Schedule Commands

Command Name	Description
jobmon	Provides a list of all job occurrences and their job ID in the database for the specified day.
listrule	Lists all job and job group rules in the database.

Dependency Commands

Command Name	Description
jobdep	Displays all file or job dependencies for a job or job group.
depdel	Deletes job and file dependencies.
depadd	Adds job and file dependencies.

Job and Job Group Definitions Commands

Command Name	Description
submit	Submits a job or group definition into the production schedule.
modrule	Modifies a job rule created with **addrule**.
delrule	Deletes a job or job group definition (job rule).
inactrule	Inactivates a job or job group definition.
addrule	Adds a job or job group to the database.
liststat	Lists all statuses and their corresponding numbers for input into commands.

Command Name	Description
calendar	Lists all calendars in the database.
alerts	Lists all the alerts presently in the production schedule.
alertset	Manually sets the status of an alert.
varset	Manually sets the value of a pre-existing variable.
hosts	Lists all the available CWA hosts on the network.

Batch Processing Command

Command Name	Description
file	Runs a batch of commands listed in an input ascii file.

Help Command

Command Name	Description
help	Invokes the help text.

Command Summary

2

Command Descriptions

This chapter contains command descriptions for the Command Line program.

addrule

Warning: Always enclose arguments containing spaces in quotes, or the command will not execute successfully. For example, if the Program Files folder is included in a path statement as an argument, the entire path must be enclosed by quotes.

The **addrule** command defines a new job rule (job definition). It *must* be followed by the **modrule** command to further define other job parameters.

After **addrule** has been issued, use the *submit* command to add the job to the production schedule.

Syntax

SACmd addrule -J|-G -n job_name|-a job_alias [-p parent_alias]
[-C command] [-E calendar_name] [-o offset] [-h agent_name|-L agent_list] [-k intrvl counts] [-I interval] [-P "parameters"] [-Q y|n] [-U <runtime_user_name>]

Note: The -C command only works with jobs; it will not work with job groups.

addrule -options

Option	Description
-J	Makes the rule a job definition. (Requires the **-C** command to work.) Either this option or the **-G** option is required.
-G	Makes the rule a job group definition. Either this option or the -J option is required.
-n job_name	The name of the job, up to 50 characters. If the job name is specified only, the job alias is set to the job id. Either this option or the **-n job_alias** option is required.
-a job_alias	Defines the job alias string, up to 8 characters. This string provides a unique ID representing the job. If the job alias is not unique, does not accept the rule. If only the job alias is specified, the job name is set to the job alias name. Either this option or the **-n job_name** option is required.
-o offset	The calendar offset in days. You can choose to subtract or add an offset to the specified calendar date. For example, if your calendar specifies 1/19/00 as a date to run your job, and you choose an offset of **1**, then the job runs on 1/20/00.
-p parent_alias	The alias name of the job's job group. Specifying this field adds the job to the job group defined by this alias.
-h agent_name	The name of the agent on which the command is run. If specifying an agent list, use with the **-L** option. If this option is not specified, the configured default agent is assumed. This option is required when using the **-L** option. ***NOTE***: Be careful to use the specific name of your agent, not the name of the agent machine.
-k intrvl_counts	Interval count for repeating jobs. Use to set the number of times to repeat the job at the interval specified by the **-I** (uppercase i) option.
-I interval	(Uppercase i) Interval for repeating jobs. Use to set the interval in minutes at which to run repeating jobs (ranging from **0-1440**). This option should be used with the **-k** and **-Q** options. For example, to rerun the same job occurrence every five minutes, 10 times, **-I 5 -k 10 -Q Y** to rerun a new job occurrence every five minutes, 10 times, **-I 5 -k 10 -Q N**

Option	Description
-C command	The path and filename of the command run by the job. If specifying a job group, this option is ignored. You must include the path to the command. For example, -C “c:\program files\reports.bat” Unix Master: In Unix, if you are specifying a directory path to a Windows machine, you must use two backslashes instead of the normal single backslash of a Windows directory path. Unix won’t recognize the single backslash. For example, in Unix, to specify the reports.bat command file on a Windows machine, **-C “\\Computer1\\program files\\reports.bat”** For example,in Windows, to specify the reports.bat command file from the programs shared directory located on computer1, -C “\\Computer1\program files\reports.bat”
-E calendar_name	The name of the calendar to associate with the job or job group.
-L	If present with the **-h** option, selects the agent list name instead of the agent name. If omitted, defaults to the agent name.
-P “parameters”	Command parameters. Use this option to specify the parameters to pass to the command file. When you specify parameters, separate them with spaces. Remember if an argument includes spaces, to enclose the entire argument in quotes. For example, to specify **OFF** as your first parameter, and **98** as your second parameter: **-P “OFF 98”** When creating an SAP ABAP job, use the syntax below to provide the parameters: ABAP=<program_name>, VARIANT=<variant_name>[, VARMASK=<runtime_variant>, VARVALS=<variant_values>] where VARMASK and VARVALS are optional parameters. To create jobs with multiple steps, you can provide multiple ABAP parameters as shown here: ABAP=<program_name>, VARIANT=<variant_name>[, VARMASK=<runtime_variant>, VARVALS=<variant_values>] ABAP=<program_name>, VARIANT=<variant_name> where ABAP is a separator for different steps.
-Q Y|N	Specify whether to allow the job to repeat (**Y**) or not (**N**). Any parameter other than **Y** or **N** is interpreted as **N**.
-U runtime_user_name	This option is mandatory when creating an SAP job. Provide a valid user name which can be used as a runtime user for the job.

Examples

- This example creates a new job with the alias name **clock** using minimum initial parameters:

  SACmd addrule -J -a clock -C c:\winnt\clock.exe -h "Win Agent"

  The minimum set of parameters are the name or alias, command (jobs only), agent or agent-list name and the **-J** or **-G** option.

- The following example adds the calendar definition, job name and parent name to the above command.

SACmd addrule -J -a clock -C c:\winnt\clock.exe -h "Win Agent" -E Daily -n clock_2 -p TIME

Note that `Win Agent` was placed in quotes. This is because of the space character in the string.

- Define a similar job as above but now define an agent list instead of an agent:

  SACmd addrule -J -a clock -C c:\winnt\clock.exe -h ListAgent2 -L

- Create a new group with the name alfa using the minimum number of arguments:

  SACmd addrule -n alfa -G

- Add other options to the last example, such as name, offset, calendar and parent alias:

  SACmd addrule -n alfa -G -E Daily -a alfa23 -p BETA -o 5

- This example shows the steps you need to take to create, modify and submit a new job to the master:

  SACmd addrule -J -a sleep -n sleep -C c:\master\sleep.exe -h "Win Agent"

  SACmd modrule -a sleep -E Holidays -P 8 -u jamesb -y 55

  SACmd submit -a sleep

- This example creates an SAP ABAP job with a single step:

  addrule -J -n "SAPSACMD_Test14" -a BZ123468 -U tidal -h sap_conn_new -C SAP -P ABAP=BTCTEST,VARIANT=variant1

  This example creates an SAP ABAP job with two steps:

  addrule -J -n "SAPSACMD_Test15" -a BZ123469 -U tidal -h sap_conn_new -C SAP -P ABAP=BTCTEST,VARIANT=variant1, ABAP=BTCTEST2,VARIANT=variant2

agent

The `agent` command adds, modifies, deletes and enables or disables a connection to an agent provided you have the correct security privileges to add or edit a connection.

Syntax

SACmd agent -C [ADD|MODIFY|DELETE|LIST] -n name

ADD [-n name -t [Windows|Unix|zOS|OVMS] -a hostname -p portnumber -e Y|N

MODIFY [-n name -e Y|N]

MODIFY [-n name -a hostname -p port]

DELETE [-n name]

LIST [no parameters needed] see also host or hosts

agent -options

Job Status	Job Status ID
-name	Specifies the name of the connection.
-e Y|N	Specifies whether the connection is enabled (`Y`) or disabled (`N`).
-t Windows|Unix| zOS|Open VMS	Type of connection to add an agent for Windows, Unix, z/OS or Open VMS.
-p port number	Possible port numbers range from `1-65535`. Default is `5912`.
-a hostname	The machine name or IP address of the machine. Defaults to name parameter.

Examples

- The following example creates and enables a new Windows agent called `Win04` on the Saturn host machine using port number `5914`:

  SACmd agent -C ADD -n Win04 -t Windows -a Saturn -p 5914 -e Y

- The following example changes the port number of the `Win04` agent on the Saturn host machine to `5915`.

  SACmd agent -C MODIFY -n Win04 -a Saturn -p 5915

- This example disables the `Win04` agent.

  SACmd agent -C MODIFY -n Win04 -e N

alerts

The `alerts` command displays all the operator alerts presently in the production schedule in table format. The columns included are:

- ID – The alert ID (needed for the alertset command).
- Job Number – The job number ID of the job that issued the alert.
- Type – The kind of alert issued.
- Level – The severity level of the alert, either Critical, Warning, Error or Information.
- Status – The status of the alert, either Open(1), Acknowledged(2) or Closed(3).
- Description – The alert message as defined in the Operator Alert Action used to issue the alert.
- Response – Operator notes taken in response to the alert.
- Time – The time the Operator Alert was closed.
- User – The runtime user of the job that closed the alert.

Syntax

SACmd alert[s] [-f from_date] [-t to_date] [-b] [-s]

alert[s] -options

Job Status	Job Status ID
-f from_date	Specifies the date to start displaying alerts. If omitted, the default is today. The format for the date is YYYYMMDD. Do not use any punctuation when entering the date.
-t to_date	Specifies the date to stop displaying alerts. Alerts that fall on the day specified are included. If omitted, the default is today. The format for the date is YYYYMMDD. Do not use any punctuation when entering the date.
-b	Suppress the header which otherwise identifies the contents of each column.
-s	Display only job number, job name, alert message, response and user.

Examples

- The following example displays all the alerts, suppresses the header and displays only the job number, job name, alert message, response and user.

  SACmd alert -b -s

- The following example displays all alerts.

  SACmd alerts

alertset

Warning: Always enclose arguments containing spaces in quotes, or the command will not execute successfully. For example, if the Program Files folder is included in a path statement as an argument, the entire path must be enclosed by quotes.

The **alertset** command lets you manually set the status of an alert specified by the alert ID. To obtain the job number alert ID use the **alerts** command.

Syntax

SACmd alertset [-s status] [-i alert_ID]

alertset -options

Option	Description
-s status	Sets the status of the alert specified with the **-i** option to one of the following: **OPEN** (or **1**), **ACK** (or **2**) or **CLOSED** (or **3**).
-i alert_ID	Sets the alert associated with this alert ID to the status specified with the **-s** option.

Operation

You can change an alert's status to any of the available states. Available states include **OPEN**, **ACK** for acknowledged and **CLOSED**. The alert status can be referenced by the alphabetical or numerical designation.

Examples

- The following example sets the status of the alert associated with job number **254** to **ACKNOWLEDGED**.

SACmd alertset -i 254 -s ACK

- The following example sets the status of the alert associated with job number **254** to **CLOSED**.

  SACmd alertset -i 254 -s 3

calendar

A set of dates specifying when a job will run. Calendars are constructed from the **Calendars** pane. Each calendar has a name and can be selected for scheduling jobs in the job and job group definition dialogs. You can also create calendar groups. For more information on calendars, see *Chapter 9: Monitoring* CWA *Production* in the *Reference Guide*.

This command displays a list of calendars accessible to the current user.

Syntax

calendar

calrecalc

The **calrecalc** command recalculates all calendar dates.

Syntax

SACmd calrecalc

calrecalc -options

The **calrecalc** command has no options.

Operation

This command recompiles the dates for all calendars. The calendars are calculated for the current year and the next three years.

compile

Warning: Always enclose arguments containing spaces in quotes, or the command will not execute successfully. For example, if the Program Files folder is included in a path statement as an argument, the entire path must be enclosed by quotes.

The **compile** command compiles the production schedule for the dates specified.

Syntax

SACmd compile [-f begin_date][-t end_date]

compile -options

Option	Description
-f begin_date	Jobs designated to run on the **begin_date** and after will be compiled.
-t end_date	Jobs designated to run on and before the **end_date** will be compiled.

Operation

If only the **-f** option is supplied, the range of compile dates is from the specified **begin_date** to today. If only the **-t** option is supplied, the range of compile dates is from today to the **end_date** specified. When no option is supplied, the compilation is for today. Dates must always be in the format **YYYYMMDD**.

Within the date field, you can also include the time. When entering the time, you must always enter it in the format defined in the Windows Regional Settings Control Panel. Dates and times are displayed as configured by the date/time settings in this control panel.

Examples

- The following example compiles today's production schedule only.

  SACmd compile

- The following example compiles the production schedule from December 5, 2010 to today.

  SACmd compile -f 20101205

- The following example compiles the production schedule from today to January 5, 2010.

  SACmd compile -t 2010 105

- The following example compiles the production schedule for December 6 to December 10, 2010.

  SACmd compile -f 20101206 -t 20101210

delrule

Warning: Always enclose arguments containing spaces in quotes, or the command will not execute successfully. For example, if the folder Program Files is included in a path statement as an argument, the entire path must be enclosed by quotes.

The **delrule** command deletes a job or job group definition. You can either specify the alias or the ID of the job or job group.

Syntax

delrule -a alias|-i ID

delrule -options

Option	Description
-a alias	Specifies the alias name of the job or job group to delete. Either this option or the **-i ID** option is required.
-i ID	Specifies the identification number of the job to delete. Either this option or the **-a alias** option is required.

Operation

You can use the **listrule** command to obtain job rule IDs.

Examples

- The following command deletes the job definition with the alias name **invent**.

SACmd delrule -a invent

- The following command deletes the job definition number `3267`.

  SACmd delrule -i 3267

depadd

Warning: Always enclose arguments containing spaces in quotes, or the command will not execute successfully. For example, if the Program Files folder is included in a path statement as an argument, the entire path must be enclosed by quotes.

The **`depadd`** command adds a new job dependency or file dependency.

Syntax

SACmd depadd -a alias -t FILE|JOB|VARIABLE
[-d alias -o operator -s status -l occur -z offset -w offset -g]|
[-h maser_name iv variable_name -o operator -u value]
[-f file_name -y file_type -e file_extent -h agent_name]

depadd -options

Option	Description
-a alias	Specifies the alias of the job receiving the new dependency. This option is required.
-t FILE|JOB| VARIABLE	Specifies the type of dependency: for a file dependency, specify **`FILE`**; for a job dependency, specify **`JOB`**: for a variable dependency, specify **`VARIABLE`**. This option is required.

Job Dependency Options

Option	Description
-d alias	Specifies the alias name of the job or job group to depend upon. Use this along with the -o and -s options to specify a dependency being met when a job equals or does not equal a particular status.
-o operator	The operator in string or numeric format. You can choose: **Equals**, or 1 (default) **NotEquals**, or 2 Use this along with the **-d** and **-s** options to specify a dependency being met when a job equals or does not equal a particular status.
-s status	The status in string or numeric format. You can choose: **Active**, or 51 **"Completed Abnormally"**, or 103 **"Completed Normally"**, or 101 (default) **"Error Occurred"**, or 66 **"Externally Defined"**, or 107. Use this along with the -d and -o options to specify a dependency being met when a job equals or does not equal a particular status. Note that the statuses containing a space must be in quotes.
-l occurrence	Creates a dependency on the First Occurrence or Last Occurrence of the dependency job, or you can set it to Match Occurrence if you want the job to repeat along with the repeating dependency job. First Occurrence: **First**, or 1 Last Occurrence: **Last**, or 2 Match Occurrence: **Match**, or 3
-z offset	If the first occurrence or last occurrence is specified in logic, this value is a numerical value indicating either a positive offset from the first occurrence, or a negative offset from the last occurrence.
-w offset	When first occurrence or last occurrence of the **-l** (lower case L) option is selected, you can specify an offset from these occurrences. The offset is always positive. For example, if the last occurrence is specified, there are a total of 5 occurrences, and you specify 3 for this offset, the dependency is met on the second occurrence.
-g	Specify this option to ignore the dependency if the dependency job is not in the production schedule.

File Dependency Options

Option	Description
-f file_name	The name of the file that the job is dependent on. If the agent specified for the file dependency is an Windows agent and the file dependency type is [**Size stable**, **Size has changed in**], then file name may include wildcards (that is *,?,etc.) in either the name or path portions of the file name. If this is the case, the following warning is displayed "Using wildcards with a large result set may delay the resolution of file dependencies for other jobs. Is this okay? [**Y,N**]". If you respond "**N**", the dependency is not written to the database. The use of wildcards only apply to the above described agent and file dependency types.
-h agent_name	The agent that the dependency file exists.
-y dep_type	The type of file dependency. Use the numeric or string values in the following dependency types: File exists: **Exists**, or **200** File doesn't exist: **NotExists**, or **201** Size is less than or equal to `extent`: **SizEle**, or **202** Size is greater than or equal to `extent`: **SizEge**, or **203** Size stable for `extent`: **Stable**, or **204** Size has changed in `extent`: **Changed**, or **205**
-e extent	Use this field to specify the extent value for the last four options of the dependency types listed above. For **Size <=** or **Size >= file size**, this value is in bytes. For **Size stable for** or **Size has changed in**, this value is in minutes.

Variable Dependency Options

Option	Description
-h master_name	The remote master on which the variable exists.
-v variable_name	The name of the local or remote variable.
-o operator	The operator in string or numeric format. You can choose: **Equals**, or 1 (default) **NotEquals**, or 2 Use this along the -d and -s options to specify a dependency being met when a job equals or does not equal a particular status.
-u variable_value	The value of the variable required to meet the dependency.

Examples

- Add a dependency for a repeating job with the alias **payroll** to depend on the repeating job with the alias name **invent**. The dependency is met when the dependency job's status equals **Completed Normally**. Match the occurrence with the repeating dependency job and do not use an occurrence offset. Do not ignore the dependency if the job is not in the schedule.

  SACmd depadd -a payroll -t JOB -d invent -s 101 -o 1 -l 3 -z 0 -g N

- Make the job with the alias **payroll** dependent on the file **file1.doc** existing on the Windows agent.

  SACmd depadd -a payroll -t FILE -f c:\file1.doc -h "Windows Agent" -y 200 -e 0

- Make the job with the alias **payroll** dependent on all files in C:\ whose sizes have been stable for one hour from the present time.

  SACmd depadd -a payroll -t FILE -f C:*.* -h "Windows Agent" -y 204 -e 60

depdel

Warning: Always enclose arguments containing spaces in quotes, or the command will not execute successfully. For example, if the Program Files folder is included in a path statement as an argument, the entire path must be enclosed by quotes.

The **depdel** command deletes job dependencies and file dependencies. It then replaces that job's pre-done instances in the production schedule. Refer to the **submit** command.

Syntax

SACmd depdel -a alias [-t FILE|JOB|VARIABLE]
[-d alias -o operator -s status -l occur -z offset -w offset -g]|
[-f file_name -y file_type -x file_extent -h agent_name] [-r Y|N] [-e yyyymmdd]
[-h master_name -v variable_name -o operator -u value]

depdel -options

Caution: Options to the depdel command restrict which dependencies are deleted. If no options are specified, all dependencies for the specified job are deleted.

Option	Description
-a alias	Specifies the alias of the job or job group with the dependency to delete. This option is required.
-t FILE|JOB| VARIABLE	Specifies the type of dependency to delete: for a file dependency, specify **FILE**; for a job dependency, specify **JOB**; for a variable dependency, specify **VARIABLE**.
-d alias	Specify this option to delete only job dependencies with the alias name of the job or job group that the above alias depends upon.
-y dep_type	Specify this option to delete only the file dependencies with the following file dependency type (in numeric or string format): File exists: **Exists**, or **200** File doesn't exist: **NotExists**, or **201** Size is less than or equal to extent: **SizEle**, or **202** Size is greater than or equal to extent: **SizEge**, or **203** Size stable for extent: **Stable**, or **204** Size has changed in extent: **Changed**, or **205**
-x extent	Specify this option to delete only the file dependencies with the following extent value. For **Size <=** or **Size >= file** size, this value is in bytes. For **Size stable for** or **Size has changed in**, this value is in minutes.
-h agent or master_name	Specify this option to delete file or variable dependencies where the dependency exists on the agent indicated.

Option	Description
-s status	Specify this option to delete only job dependencies containing the status indicated. The status is stated in character or numeric format. The status can be: **Active**, or **51** **"Completed Abnormally"**, or **103** **"Completed Normally"**, or **101** **"Error Occurred"**, or **66** **"Externally Defined"**, or **107**.
-r Y | N	Specify this option to submit repeating jobs immediately. If omitted, this defaults to **N**.
-e yyyymmdd	Specifies a date in the production schedule in which the job is to be submitted. If omitted, this defaults to the first production date as specified by its calendar.
-v variable_name	Specify this option to delete only the varible dependencies that refer to the varible name indicated.
-u value	Specify this option to delete only the variable value dependencies that refer to the value indicated.
-y dep_type	Specify this option to delete only the file dependencies with the following file dependency type (in numeric or string format): File exists: **Exists**, or **200** File doesn't exist: **NotExists**, or **201** Size is less than or equal to extent: **SizEle**, or **202** Size is greater than or equal to extent: **SizEge**, or **203** Size stable for extent: **Stable**, or **204** Size has changed in extent: **Changed**, or **205**
-x extent	Specify this option to delete only the file dependencies with the following extent value. For **Size <=** or **Size >= file size**, this value is in bytes. For **Size stable for** or **Size has changed in**, this value is in minutes.
-h agent or master_name	Specify this option to delete file or variable dependencies where the dependency exists on the agent indicated.
-s status	Specify this option to delete only job dependencies containing the status indicated. The status is stated in character or numeric format. The status can be: **Active**, or **51** **"Completed Abnormally"**, or **103** **"Completed Normally"**, or **101** **"Error Occurred"**, or **66** **"Externally Defined"**, or **107**.
-r Y | N	Specify this option to submit repeating jobs immediately. If omitted, this defaults to **N**.
-e yyyymmdd	Specifies a date in the production schedule in which the job is to be submitted. If omitted, this defaults to the first production date as specified by its calendar.
-v variable_name	Specify this option to delete only the varible dependencies that refer to the varible name indicated.

Examples

- The following example deletes all dependencies for the job referred by **JobAlias**.

  SACmd depdel -a JobAlias

- The following example deletes all file dependencies running on the agent **Windows Agent** for job alias `payroll`.

  SACmd depdel -a payroll -t FILE -h "Windows Agent"

- The following command deletes all job dependencies with the alias `holiday` that job alias `payroll` depends on.

  SACmd depdel -a payroll -d holiday -t JOB

file

Warning: Always enclose arguments containing spaces in quotes, or the command will not execute successfully. For example, if the Program Files folder is included in a path statement as an argument, the entire path must be enclosed by quotes.

The `file` command lets you pipe an input *ascii* text file containing CWA commands for batch processing.

Syntax

SACmd file file_name

Note: Specify the full, absolute directory path to your file.

Operation

The file should contain a list of individual commands, with each command separated by a carriage return (one command per line).

For example:

 # this is just a test script

listrule

jobmon +iaE

jobadd -a payroll

To add comments to the text file, precede the comment with two forward slashes (//) or with the sharp (#) sign. The `file` command ignores any line beginning with two forward slashes or the sharp sign. Any line composed entirely of spaces or tabs is also ignored.

If a command in the list produces an error, that command is skipped, but the remaining lines are processed.

Example

The following example inputs the file *cmdlist.txt* to the `file` command. `cmdlist.txt` contains a list of CWA commands to process.

SACmd file cmdlist.txt

grpupd

The `grpupd` command updates inherit attributes for jobs in the specified group. You can obtain the group's job run ID by using the `jobmon` command.

Syntax

SACmd grpupd -i ID -a

grpupd -options

Option	Description
-i ID	Specifies the job run ID of the Job Group.
-a	Specifies to update agent information for jobs that are defined to inherit agent from the group.

help

The **help** command invokes help text explaining the syntax and options of each command.

Syntax

help [cmd_name]

help -options

Option	Description
cmd_name	The name of the command that you want help on.

Operation

When **help** is invoked on a command, only one page is displayed at a time. To move to the next page, press the ENTER key. To quit help, press any alphanumeric key followed by the ENTER key.

help invoked with no command name displays all available commands. If **SACmd** is substituted for **cmd_name**, general information about the *sacmd.cmd* (Windows) or *sacmd.sh* (Unix/Linux) program is displayed. Invoking **help** on the **help** command provides an explanation on using the **help** command.

Note: You can execute **SACmd** with the **-help** option for an understanding of available arguments. For example,
>"C:\Program Files (x86)\TIDAL\TESCmdLine\bin\sacmd.cmd" -help

Note: Once you have entered a **SACmd** session, you can enter **help** for an understanding of available commands. For example, **SACmd>help**.

Example

The following example gets help on the **jobadd** command.

help jobadd

historyPurge

The **historyPurge** command deletes log data stored about jobs for a specified period of time. This data can range from data logged about audits, errors and alerts pertaining to jobs to data directly related to past instances of the job.

Caution: This command is *only available for tesmcmd*. This command is not available for SACmd.

Syntax

historyPurge [date1 [date2]]

historyPurge -options

Option	Description
date1	Logs from the specified date and older are deleted.
date2	Job history from the specified date and older is deleted.

Operation

Use the following format when specifying the **`date1`** and **`date2`** options:

```
YYYYMMDD
```

If only one date option is specified then both the logs data and the job history are deleted. If both the **`date1`** and **`date2`** options are specified then both the logs and the job history are deleted but different dates can be assigned for purging these two types of data.

If no option denoting a date is used than both log data and job history is deleted starting from the current date.

Example

The following example deletes the log data for all jobs that ran on February 1, 2006 and earlier and deletes the job history for all jobs that ran on February 15, 2006 and earlier.

tesmcmd historyPurge 20060101 20060215

The following example deletes both the log data and job history for all jobs that ran on February 1, 2006 and earlier.

tesmcmd historyPurge 20060101

hosts

The **`hosts`** command lists information about all CWA hosts defined in CWA.

Syntax

SACmd host[s] [-b]

hosts -options

Option	Description
-b	Suppresses the header information.

inactrule

The **`inactrule`** command inactivates, or disables, a job or job group. When a job or job group is inactivated, its occurrences (if any) are pulled from the production schedule.

Syntax

SACmd inactrule -a alias|-i ID

inactrule -options

Option	Description
-a alias	Specifies the alias name of the job or job group to disable. Either this option or **-i ID** is required.
-i ID	Specifies the ID of the job or job group to disable. Either this option or **-a alias** is required.

Examples

- The following example inactivates the job with the alias name **invent**.

  SACmd inactrule -a invent

- The following example inactivates the job with the ID **2150**.

  SACmd inactrule -i 2150

jobadd

Warning: Always enclose arguments containing spaces in quotes, or the command will not execute successfully. For example, if the Program Files folder is included in a path statement as an argument, the entire path must be enclosed by quotes.

The **jobadd** command lets you add a job or job group to the production schedule. You can add a job either by alias or by ID. You can obtain the job occurrence ID and/or alias by using the **listrule** command. Unlike the **submit** command, the job is submitted adhoc. An adhoc job isn't dependent on a calendar because a new instance is submitted manually into the schedule. Job groups can specify group parameter values if they are expressed in **NAME=VALUE** pairs.

Syntax

SACmd jobadd -a alias| -i ID [-p params]

jobadd -options

Option	Description
-a alias	Specifies the alias name of the job or job group to add. Either this option or **-i ID** is required.
-i ID	Specifies the ID of the job or job group to add. Either this option or **-a alias** is required.
-p params	Specifies override parameters. For group parameters, separate **NAME=VALUE** pairs with commas.

Examples

- This example adds a job with the alias name **invent2**.

  SACmd jobadd -a invent2

- The following example adds a job with the job ID **10000**.

SACmd jobadd -i 10000

- The following example adds a job group with the job ID **4192** and specifies values for the variables, **QUE**, **APPL** and **QUESUB**. Notice that the value pairs are separated by commas.

  SACmd job add 4192 -p "QUE=Q123, APPL=A09, QUESUB=QS2"

jobcancel

The **jobcancel** command cancels a job occurrence with the specified job ID from the production schedule. You can also specify whether canceling the job affects other dependent jobs. You can obtain the ID by running the **jobmon** command.

Syntax

SACmd jobcancel ID [YES|NO]

jobcancel -options

Option	Description
ID	The jobrun ID of the job to cancel. This option is required.
YES|NO	Specifies whether to also cancel successor jobs. If **YES**, its dependent jobs are also cancelled. If **NO**, its dependents are still dependent on this specific job. The dependent jobs will continue to wait until the dependency is met. The default is **NO**. This value must follow the ID.

Operation

When cancelled, only job occurrences that are **Waiting on Dependencies**, **Waiting on Resources** or **Waiting on Operator** end up with the status **Cancelled**. When a job occurrence is in the **Active** state, its resulting status is **Aborted**. A job's status can be determined using the **jobmon** command.

Examples

- The following example cancels the job with the job ID **12146** and releases its dependents.

  SACmd jobcancel 2146 YES

- The following example cancels the job with job ID **1857** and does not release its dependents.

  SACmd jobcancel 1857

- The following example cancels the job with job ID **2489** and does not release its dependents.

  SACmd jobcancel 2489 no

jobdep

The **jobdep** command displays all the dependencies of the specified type belonging to a job or job group.

Syntax

SACmd jobdep -i ID|-a alias -t FILE|JOB [-b]

jobdep -options

Option	Description
-i ID	The rule ID of the job whose dependencies you want to see. To obtain rule ID's use the **listrule** command. Either this option or **-a alias** is required.
-a alias	The alias of the job whose dependencies you want to see. Either this option or **-i ID** is required.
-t FILE|JOB	The type of dependencies you want to see, either all of the job's file dependencies or job dependencies. This is not case sensitive. The default is **JOB**. This option is required.
-b	Optionally suppresses the header.

Examples

- The following example displays all the file dependencies belonging to job rule ID **2346** and suppresses the header in the output.

  SACmd jobdep -b -i 2346 -t FILE

- The following example displays all the job dependencies belonging to the job with the alias **payroll**.

  SACmd jobdep -t JOB -a payroll

jobgo

The **jobgo** command overrides all dependencies for a job or job group, allowing the job or job group to launch. Obtain job run IDs by running the **jobmon** command.

Syntax

SACmd jobgo ID

jobgo -options

Option	Description
ID	The jobrun ID of the job whose dependencies you want to override. This option is required.

Operation

Only job occurrences in a **Waiting on Dependencies** status can be overridden.

jobhold

The **jobhold** command prevents a job occurrence with the specified job occurrence ID from running in the production schedule. Obtain the job occurrence ID by running the **jobmon** command.

Syntax

SACmd jobhold ID

jobhold -options

Option	Description
ID	The jobrun ID of the job you want to hold. This option is required.

Operation

On Windows agents, only job occurrences in a **`Waiting on Dependencies`** or **`Waiting on Resource`** status can be put on hold. For jobs running on other agent platforms, only jobs in a **`Waiting on Dependencies`**, **`Waiting on Resource`** or **`Active`** status can go to a **`Held`** status. When a job occurrence is in the **`Active`** status, its resulting status will be **`Stopped`**.

jobmod

The **`jobmod`** command modifies a Job occurrence. You can obtain the job run ID by using the **`jobmod`** command.

Syntax

SACmd jobmod -i ID [options]

jobmod -options

Option	Description
-i ID	Specifies the ID of the job or job group.
Command Options	
-C command_path	Specifies the path and filename of the command that is run by the Job.
-P parameters	Specifies one or more parameters to be passed to the command. For multiple parameters, you MUST separate them with spaces and then quote the string.
-e environ_file	Specifies an environment file for the command specified using the **`-C`** option. You MUST also specify the file's path.
-O output_filename	Specifies the output file name. Some jobs create an output file with a specific file name that you would like Master to recognize. Use this option to specify the name and location of the output file. When the job completes, Master WILL NOT create a default standard output file on the agent machine. Instead, you WILL have the output created by the executable you specified via the **`-C`** option.
Run Commands	
-h agent_name	Specifies the name of the agent on which to run the job. If the Agent name has a space, the name MUST be enclosed in quotes. You CANNOT use this option with the **`-g`** option.
-g agent_list	Specifies the name of the agent list on which to run the Job. If the Agent List name has a space, the name MUST be enclosed in quotes. You can NOT use this option with the **`-h`** option.

Option	Description
-u agent_name	Specifies the login account to use when running the job. Use this option when you are running a job as another user, and that user has access to data or resources required for the job that would NOT be accessible under your user account.
-u adm	Specifies adm as a UNIX user.
-m track	Specifies track method to determine the results of a Job.
-r track_cmd	Specifies the tracking command to use when Command (or 3) is selected using the `-m` option. The command must be in quotes.
-r search_text	Specifies the search string to use when NormalText (or 4) or AbnormalText (or 5) is selected using the `-m` option. The string MUST be in quotes.
Options Commands	
-y job_priority	Specifies the job priority relative to other jobs in the job's destination queue. The range is from 0 to 100, where 100 is the highest priority. The default when the job is first created is 50.

jobmon

Warning: Always enclose arguments containing spaces in quotes, or the command will not execute successfully. For example, if the Program Files folder is included in a path statement as an argument, the entire path must be enclosed by quotes.

The `jobmon` command lets you display job occurrence information. Command options allow you to filter the information displayed.

Syntax

SACmd jobmon [-d date] +display_options [filtering_options] [-b]

jobmon -options

Option	Description
-d date	The production date to display. The default is today. The format for the date is YYYYMMDD. Do not use any punctuation when entering the date.
+display_options	The list of information to display. You specify display options with no spaces, e.g. **+aivo**. The order of the options dictates the order of the columns in the display. If no options are specified, the options default to **+itnau**. You can choose from the following display options:

Option	Description
r	Job rule ID
i	Job run ID
p	Parent job group ID
j	Job type (job or job group)
c	Occurrence number
o	Job or job group owner
u	Runtime user
h	Agent The Agent the job runs on
z	Scheduled vs. unscheduled job
t	Job start time
s	Job status
v	Job duration
n	Job name
a	Job alias
q	Queue
x	Exit code
filtering_options	Specifies the records to filter for display. The options are preceded by the -sign and the value. You can specify the following filters:
-r	rule_id Job rule ID
-p	group_id Parent job group ID
-j	type Job type. You can choose: job group: group, or 1 job: Job, or 2
-o	owner Job or job group owner
-u	run_user Runtime user
-h	agent The Agent name the job runs on status job status. Refer to job status ID Cross Reference for more information.
-s	alias Job alias
-a	exit_code Exit code
-x	Suppresses the header information.

Examples

- The following command displays job aliases for owner **administrator** and user **mike**.

  SACmd jobmon +a -o administrator -u mike

- Display instance, status, start time, job queue and exit code for all instances of job alias **payroll**.

  SACmd jobmon +cstqx -a payroll

- Display instance, run ID and exit code for all group-type jobs for owner **ADMIN**.

  SACmd jobmon +cix -o ADMIN -j 1

  Same as above but display only job-type jobs.

  SACmd jobmon +cix -o ADMIN -j 2

- Display run ID, alias and start time for all jobs with a **Completed Normally** status.

  SACmd jobmon +iat -s 101

  Same as above but filter only the jobs belonging to user **michaelj**.

  SACmd jobmon +iat -s 101 -u michaelj

jobrelease

The **jobrelease** command resumes a job occurrence that was held with the jobhold command and releases a job in a **Waiting on Operator** status. You can obtain the job run ID by using the **jobmon** command.

Syntax

SACmd jobrelease ID

jobrelease -options

Option	Description
ID	The run ID of the job to release. This option is required.

Operation

Only job occurrences in a **Hold**, **Stopped** or **Waiting on Operator** status can be released or resumed. Jobs in a **Hold** status return to either a **Waiting on Dependencies** or **Waiting on Resource** status. Jobs in a **Stopped** status return to the **Active** status.

Example

The following command releases job run ID 53 currently in the **Waiting on Resource** status.

SACmd jobrelease 53

jobremove

The **jobremove** command removes job occurrences from the production schedule. The job must have a pre-launched status. Once a job reaches **Launched**, **Active** or one of the **Completed** statuses, it cannot be removed from the schedule.

Syntax

SACmd jobremove ID [ALL|ONE] [YES|NO]

jobremove -options

Option	Description
ID	The job run ID of the job to remove. This option is required.
ALL|ONE	Specifies whether to remove all occurrences of the job that have not run, including occurrences for future dates or whether to remove just the specified job run occurrence. If ALL is specified, this parameter refers to the job rule ID of the specified job run ID to remove all occurrences.If not specified, the default is **ONE** to remove just the specified job instance.
YES|NO	Specifies whether to release the job's dependents when removing the job. If **YES**, its dependents will no longer be dependent on this job. If **NO**, its dependents stay dependent on this job. The default is **NO**. This value must follow after the ID.

Operation

The ID parameter is required: the rest are optional. If all optional parameters are specified, they must be specified in the order given. Because a job can be placed in the **Stopped** or **Aborted** status while in the **Active** status using the **jobhold** or **jobcancel** commands, **jobremove** does not remove jobs with those statuses. Jobs with the **Held** or **Cancelled** status, however, can be removed

Warning: Active jobs *cannot* be stopped and restarted on Windows machines. Other platforms do allow this action.

Examples

- The following example removes all job occurrences and does not release any dependencies on it. In this case, the *job rule ID* of the specified job instance is used to identify and remove all other instances of that job rule that have not yet run.

  SACmd jobremove 123 ALL NO

- The following example removes specific job instances, but it does not release any dependencies on it. In this case, the ID refers to the run ID of the job being removed.

  SACmd jobremove 123 ONE NO

- The following example removes the specified job occurrence, and releases any dependencies upon it. In this case, the ID refers to the job run ID of the job being removed.

  SACmd jobremove 123 YES

jobrerun

The **jobrerun** command lets you manually rerun a completed job or job group. You can obtain the job run ID by using the **jobrerun** command.

Syntax

SACmd jobrerun ID [YES|NO] [-s step]

jobrerun -options

Option	Description
ID	The run ID of the job to rerun. This option is required.
YES|NO	Successor. This value can be **YES** or **NO** (or successor omitted where it defaults to **NO**). When **SUCCESSOR** is **YES** the master will rerun all the jobs dependent on this job (if any). This value must follow after the ID, and must be provided if the step parameter is being used.
-s step	The step to start from when rerunning a job that supports multiple steps or processes (for example, SAP or PeopleSoft jobs).

Operation

When you rerun a job, the job's dependencies are enforced.

Example

- The following example reruns the job with run ID **3245**, and does not rerun all jobs dependent upon it.

  SACmd jobrerun 3245 No

- The following example reruns the job with run ID **1834** starting at step **3**.

  SACmd jobrerun 1834 -s 3

jobset

Warning: Always enclose arguments containing spaces in quotes, or the command will not execute successfully. For example, if the Program Files folder is included in a path statement as an argument, the entire path must be enclosed by quotes.

The **jobset** command lets you manually set the completion status of a job. You can obtain the job run ID by using the **jobmon** command.

Syntax

SACmd jobset -i ID -s status

jobset -options

Option	Description
-i ID	The run ID of the job to rerun. This option is required.
-s status	The new completion status for the job. You have the following options: **"Completed Normally"**, or **101** **"Error Occurred"**, or **66** **"Completed Abnormally"**, or **103** **Skipped**, or **104** **"Externally Defined"**, or **107** **Orphaned**, or **105** This option is required. Note that statuses containing spaces must be in quotes.

Operation

You can set a job's completion status to the following statuses (which constitute all available completion statuses): **Completed Normally**, **Completed Abnormally**, **Error Occurred**, **Externally Defined**, **Orphaned** and **Skipped**.

Examples

- The following example sets the job with run ID **2476** to **Completed Normally.**

  SACmd jobset -s 101 -i 2476

- Either of the following examples sets the job with run ID **3245** to **Completed Abnormally.**

  SACmd jobset -i 3245 -s 103

  SACmd jobset -i 3245 -s "Completed Abnormally"

listrule

Warning: Always enclose arguments containing spaces in quotes, or the command will not execute successfully. For example, if the Program Files folder is included in a path statement as an argument, the entire path must be enclosed by quotes.

The **listrule** command lists information about job and job group rules. You can choose the information to list. You can also select which records to display.

Syntax

SACmd listrule +display_options [filtering_options] [-b]

listrule -options

Option	Description
+display_options	The list of information to display. You specify display options with no spaces, e.g. **+aivo**. The order of the options dictate the order of the columns in the display. The number of display options is limited to four; if more that four options are specified, only the first four display options are displayed. If no options are specified, the default **+itna** is used. You can choose from the following display options:
a	Job alias
i	Job ID
p	Parent job group ID
l	(Lower case L) Dependency logic
r	Tracking cmd
m	Track method
s	Save output
f	Time window, from time
x	Time window, until time
d	Calendar from date
v	Calendar until date
+display_options	The list of information to display. You specify display options with no spaces, e.g. **+aivo**. The order of the options dictate the order of the columns in the display. The number of display options is limited to four; if more that four options are specified, only the first four display options are displayed. If no options are specified, the default +itna is used. You can choose from the following display options:
a	Job alias
g	Agent List name
h	Agent name
y	Job priority
c	Concurrency
I	(Uppercase i) Interval at which to repeat the job.
k	Interval count
u	Runtime user
z	Unscheduled Allowed
e	Environment file name
P	Command Parameters separated by spaces (e.g. **-P OFF 98**)
C	Command file name
o	Owner name
n	Job name
j	Job type
E	Calendar name
g	Agent List name
h	Agent name
y	Job priority
filtering options	

Option	Description
-a	Alias name
-n	Job name
-o	Owner name
-u	Runtime user
-p	Parent ID
-J	Display jobs only
-G	Display groups only
-b	Suppresses header information.

Examples

- Display alias and calendar information for all jobs (no-groups) for user **jamesb**:

  SACmd +aE -J -u jamesb

- Display name, alias, priority and job rule ID for all groups belonging to user **jamesb**.

  SACmd +nayi -G -u jamesb

- Display name, calendar and command for every job-only with parent ID = **2345**.

  SACmd +nEC -J -p 2345

liststat

The **liststat** command lists all possible job statuses and their associated status number in two column format.

Syntax

SACmd liststat

liststat -options

The **liststat** command has no options.

modrule

Warning: Always enclose arguments containing spaces in quotes, or the command will not execute successfully. For example, if the Program Files folder is included in a path statement as an argument, the entire path must be enclosed by quotes.

The **modrule** command modifies a job or job group definition (rule). To modify the rule, either the job ID or an alias must be specified. In order to use **modrule**, it must be preceded by the **addrule** command.

Syntax

SACmd modrule -a alias | -i ID [options]

modrule -options

Option	Description
-a alias	The alias name of the job or job group. Job ID's can be obtained using the listrule command. For example, to specify the job with alias **12345**, -a 12345 If you choose to use the **-i** option to specify the rule, the **-a** option can be used to change the alias number. See below for more information. Either this option or **-i ID** is required.
-i ID	The rule ID of the job or job group. For example, -i 123 Either this option or **-a alias** is required. Note that if you choose to specify the job or job group using this option, you can change the alias using the -a option. For example, to open **Job 345** and change its alias to **456**, SACmd modrule -i 345 -a 456
General Options	
-n job_name	Job name. Specifies the name of the job. Names with spaces must be enclosed in quotes. For example, to specify the job name as **My Job**, -n "My Job"
-p parent_ID	The parent job group ID. Job group ID's can be obtained using the listrule command. For example, assign a job to a parent job group with ID 234, -p 234
-j job_type	Job Type. Specifies the job type for the job, **GROUP** for Job Group or **JOB** for Job. For example to specify a Job Group, -j GROUP
-J class_name	Job class. Specifies the job class for the job. If your queues are setup to receive jobs based on their class, you can use this option to direct your job to run from those queues with the specified class as a queue filter. For example, to specify the Reporting Class, -J Reporting Class To remove a job class previously specified, **-J**
-o owner_name	Owner name. Specifies the owner of the job. You can specify your screen name or any of your workgroup names. Names containing a space must be enclosed in quotes. For example, to set the job to belong to workgroup My Workgroup, -o "My Workgroup"
-T Y|N	Set the job to Enabled or Disabled. When **-T Y** is specified, you allow the job to enter the production schedule. When **-T N** is specified, you prevent the job from entering the production schedule, either automatically or manually.
Command Options	

Option	Description
-c command_path	Command file name. Specifies the command. You must include the path to the command. For example, -C c:\programs\reports.bat For example, to specify the reports.bat command file from the programs shared directory located on computer1, -C \\Computer1\programs\reports.bat
-P “parameters”	Command parameters. Specifies parameters to pass to the command file. When you specify parameters, they should be separated by spaces. For example, to specify **OFF** as your first parameter, and 98 as your second parameter: -P “OFF 98”
-e environ_file	Environment file name. Specifies an environment file for the command specified using the **-c** option. You must also specify the file’s path. For example, to specify the environment file **env1.txt**. -e c:\environments\env1.txt
-O output_filename	Output File Name. Some jobs create an output file with a specific file name that you would like CWA to recognize. Use this option to specify the name and location of the output file. When the job completes, CWA will not create a default standard output file on the agent machine. CWA will not create a default standard output file on the agent machine. Instead, you will have the output created by the executable you specified via the **-c** option. -O c:\outputfiles\output.txt
Schedule Options	
-k repeats	Specifies the number of times a job is to be repeated.
-I interval	Specifies the amount of time in seconds between repeats of a job.
-Q repeat Y|N	(Uppercase Q) Specifies if this is a job runs repeatedly. **Y** indicates that the same job occurrence will be rerun after 'interval' up to 'intrval_cnt'. (See above for definitions of 'interval' and 'intrvl_count'.) **N** indicates that a new job occurrence will run every 'interval' up to 'intrval_cnt'. (See above for definitions of 'interval' and 'intrvl_count'.) The value must be specified in upper case. If a value other than **Y** or **N** is specified, the job is set to run only once.
-V Y|N	Specifies whether the job will inherit the calendar (**Y**) or not (**N**). If the job has a parent job group specified using the **-p** option, specifying this command inherits the calendar settings from the parent group. After using this command, you do not need to specify the **-E**, **-U**, **-d**, and **-v** options.
-E calendar_name	Calendar name. Specifies the calendar to use for scheduling the job or job group. For example, to specify the work_day calendar, **-E work_day**
-U offset	Offset. Specifies a numerical offset that is applied to each day generated by the calendar. For example, if the calendar generates every Monday, an offset of 1 specifies every Tuesday. For example, to offset the calendar by one day, **-U 1**
-d from_date	Calendar from date. Specifies the date from which the job can be scheduled. The date must be specified in YYYYMMDD format. For example, to specify the calendar from date as Jan 6, 2005, **-d 20050106**

Option	Description
-v to_date	Calendar until date. Specifies the date to which the job can be scheduled. The date must be specified in YYYYMMDD format. For example, to specify the calendar to date as Jan 10, 2005, **-v 20050110**
-X Y|N	Specifies whether the job inherits the time window (**Y**) or not (**N**). If the job has a parent job group specified using the **-p** option, specifying this command inherits the Time Window settings from the parent group. After using this command, you do not need to specify the **-w**, **-f**, and **-x** options.
-w time_dep	Time window. Specifies time window parameters. To use this option you must set the time window option to prevent inheritance (**-X=N**). **NotTimeDep** or **0**: Specifies no time window for the job or job group. This is the default when the job is first created. **TimeDepAnyDate** or **1**: If the job doesn't run today within the time window specified by the -f and -x options, (due to other dependencies not being met), this option lets you try again tomorrow. **TimeDepSchDate** or **2**: If the job does not run today within the time window specified by the **-f** and **-x** options, this option prevents it from running on future days. For example, to specify that job to try again tomorrow if it is unsuccessful running in today's time window, **-X=N -w TimeDepAnyDate**
-f from_time	Time window, from time. Specifies the time dependency from which the job or job group can run from the schedule. You must specify the time in the format as shown in the Regional Settings control panel. To use this option you must set the time window option to prevent inheritance (**-X=N**). For example, **-X=N -f 04:12:00 PM**
-x to_time	Time window, until time. Specifies the time dependency to which the job or job group can run from the schedule. You must specify the time in the format as shown in the Regional Settings control panel. To use this option you must set the time window option to prevent inheritance (**-X=N**). For example, **-X=N -x 05:12:00 PM**
-I interval	(Uppercase i) Interval for repeating jobs. Specifies an interval in minutes to run repeating jobs. This option should be specified along with the **-k** option. For example to repeat the job every five minutes, 10 times, **-I 5 -k 10**
-k intrvl_count	Interval count for repeating jobs. Specifies the number of times to repeat the job at the interval specified by the **-I** (uppercase i) option. See above for example.
Run Commands	
-K Y|N	Specifies whether the job will inherit the Agent (**Y**) or not (**N**). If the job has a parent job group that is specified using the **-p** option, specifying this command inherits the Agent, Agent List and Runtime user settings from the parent job group. After using this command, you do not need to specify the **-h**, **-g**, and **-u** options. For example, to inherit the Agent from the job group, **-p "My Group" -K Y**
-h agent_name	Agent name. Specifies the name of the agent on which to run the job. If the Agent name has a space, the name must be enclosed in quotes. You cannot use this option with the **-g** option. For example, to use the Windows Agent named Batman, **-h Batman**
-g agent_list	Agent list name. Specifies the name of the agent list on which to run the job. If the agent name has a space, the name must be enclosed in quotes. You cannot use this option with the **-h** option. For example, to use the agent list named "My Agent List", **-g "My Agent List"**

Option	Description
-u runtime_user	Runtime user. Specifies the login account to use when running the job. Use this option when you are running a job as another user, and that user has access to data or resources required for the job that is not accessible under your user account. For example, to specify Alex in the acmeco domain as the runtime user, **`-u acmeco\Alex`** To specify a Unix user account such as adm, **`-u adm`**
-m track	Track method. Specifies how to determine the results of a job. You can choose: **`ExitCode`** or **`1`** to let the job's exit code determine the completion status of the job. This is the default setting when the job is first created. **`External`** or **`2`** to set the completion status to External. The Operator should then determine the actual completion status from the job's output and then set it using the jobset command or using the CWA Web client. **`Command`** or **`3`** to use an external command that inspects the job's output, such as the windows Find command, to determine the completion status. The exit code of the command becomes the exit code of the job. **`NormalText`** or **`4`** to scan the job's output for a string that indicates the job completed normally. **`AbnormalText`** or **`5`** to scan the job's output for a string that indicates the job completed abnormally.
-r track_cmd	Tracking cmd. Specifies the command to use when **`Command`** (or **`3`**) is selected using the **`-m`** option. The command must be in quotes. For example, you can search the output for the string **`OK`** using the **`Find`** command by typing **`Find `OK``**. If **`OK`** is found, **`Find`** returns **`1`** which becomes the exit code of the job, that is, **`Completed Normally`**. Note that any text that needs to be placed in quotes within the command should be enclosed in accent marks, not parenthesis or apostrophes. The accent mark is normally located on the tilde character button on your keyboard. For example, to search through the output of a job for **`Completed OK`**, -m command -r Find `Completed OK `
-r search_text	Search text. Specifies the string to use when **`NormalText`** (or **`4`**) or **`AbnormalText`** (or **`5`**) is selected using the **`-m`** option. The string must be in quotes. For example, you can search the output for the string **`OK`** to determine that the job completed normally. Note that any text needing to be placed in quotes within the command should be enclosed in accent marks, not parenthesis or apostrophes. The accent mark is normally located on the tilde character button on your keyboard. For example, to search through the output of a job for **`Completed OK`**, **`-m normaltext -r 'Completed OK '`**
-Z est_duration	Estimated duration. Specifies the estimated duration in minutes for the job. The estimated duration is useful when checking if your job runs over or under the estimated time. For example, to set the estimated duration to ten minutes, -Z 10
-N min_duration	Minimum duration. Specifies the minimum duration for your job. The minimum duration can be used to trigger a job event that takes an action if the job completes before its minimum time. For example, to set the minimum duration to 5 minutes, -N 5

Option	Description
-M max_duration	Maximum duration. Specifies the maximum duration for your job. The maximum duration can be used to trigger a job event that takes an action if the job runs after its maximum time. For example, to set the maximum duration to 15 minutes, -M 15
Dependencies Commands	
-l dep_logic	(Lowercase L) Dependency logic. Specifies whether any or all dependencies must be met for the job or job group to run. You can choose: **AND** or **1** to specify all dependencies. This is the default setting when the job is first created. **OR** or **2** to specify any dependency. For example, to specify that all dependencies must be met, -1 AND
Options Commands	
-y job_priority	Job priority. Specifies the job priority relative to other jobs in the job's destination queue. The range is from **0** to **100**, where **100** is the highest priority. The default when the job is first created is **50**. For example, to set the job priority to **65**, -y 65
-c concurrency	Concurrency. Specifies what to do if another occurrence of the job is already running. **Run** or **1**: Runs the job along with the previous occurrence. This is the default setting when the job is first created. **Skip** or **2**: Skips the job altogether. The job occurrence is never run. **Defrnrm** or **3**: Runs the job after the previous occurrence completes normally. **Defer** or **4**: Runs the job after the previous occurrence completes. For example, to set the concurrency to run anyway, -c Run
-z Y|N	Specifies that a job can be added adhoc to a schedule. Without this setting, you can only schedule the job using a calendar.
-q	Requires operator release flag. When specified, holds the job in the production schedule when all dependencies are met, until the operator releases it with the jobrelease command or using the CWA Web client.
-s Y|N|R	Save output. This option, when specified, saves the output required for the **-m** and **-r** options. If **N**, then the output is discarded. (default) If **Y**, then the output is saved and added to the previous job instance's output. If **R**, then the output is saved but overwrites (or replaces) the previous job instance's output.
-so Y|N	Summary only. If job output is to be saved or appended, this option saves the output in summary form rather than full job output. This option only applies to Oracle Applications jobs. If **Y**, then the output is presented in summary form. If **N**, then the full job output is saved. (default)

Option	Description
-R retention	Retention. Specifies the length of time in days to keep job occurrence history for this particular job in the production schedule. Job history older than the retention setting is automatically purged. For example, to set the retention history to 15 days, -R 15
Runbook Commands	
-b Runbook string	Runbook text string. Specifies the text string to be included on the **Runbook** tab of the job rule. For example, a URL.
Description Commands	
-S description	Job description. Lets you write a description for the job. The description must be enclosed in quotes. For example, to add a description, -S “This is my description.”

Operation

Due to the number of options in the **modrule** command, you can break up this command into several smaller commands. You can then use the **submit** command to add the job or job group to the production schedule.

Examples

- Modify environment file, agent and calendar for job alias **payroll**.

  SACmd modrule -a payroll -e c:\envfile.doc -h "Windows Agent" -E Dailys

- Modify job name and run time user for job ID **2345**.

  SACmd modrule -i 2345 -n new job2 -u jamesb

output

The **output** command displays the output of a job.

Syntax

SACmd output -i ID

output -options

Option	Description
-i ID	Specifies the identification number of the job that will provide the output.

Operation

Displays the output of the job in uncompressed text.

pause

The **pause** command pauses the production schedule, preventing jobs, even those whose dependencies have been met, from being launched. To restart the production schedule, use the **resume** command.

Syntax

SACmd pause

pause -options

The **pause** command has no options.

qlimit

The **qlimit** command lets you manually set the limit of an existing queue.

Syntax

SACmd qlimit -n name -l limit

qlimit -options

Option	Description
-n name	The name of the queue. Note that a queue name containing a space must be enclosed in quotes, e.g., **"system queue."**
-l limit	The new value for the queue limit.

Operation

The **qlimit** command only works with existing queues. It cannot be used when creating a new queue. You can set a value for the queue only if you have the proper security privileges to edit this queue.

Note: A queue name that contains a space must be enclosed within quotation marks when used in a command.

resume

The **resume** command resumes the production schedule, allowing jobs to be launched after using **pause** to temporarily stop the production schedule. This does not apply to jobs that are **Waiting on Dependencies**.

Syntax

SACmd resume

resume -options

The **resume** command has no options.

status

The **status** command displays the status of a job or job group instance.

The condition or state of a job occurrence varies throughout its life cycle. When a job has entered the schedule and is waiting to run or is actively running, possible statuses include:

- Active
- Waiting on Children

- Launched
- Waiting on Resource
- Waiting On Dependencies
- Held
- Agent Unavailable
- Agent Disabled
- Waiting on Group
- Timed Out for Day
- Waiting on Operator

When a job completes, possible status values are:

- Completed Normally
- Completed Abnormally
- Error Occurred
- Orphaned
- Skipped
- Aborted
- Cancelled
- Timed Out

If the status of a job occurrence is **`Externally Defined`**, then CWA is waiting for an external status update.

For more information about job statuses, see *Jobs and Job Groups* in the CWA *Reference Guide*.

Syntax

SACmd status -i ID|-a alias -o Inst

status -options

Option	Description
-i ID	The occurrence ID of the job or job group. Either this option or **`-a alias -o Inst`** is required.
-a alias	The alias of the job or job group instance. Either this option or **`-i ID`** is required. If this option is used, the **`-o Inst`** option must also be used.
-o Inst	The job or job group occurrence.

Examples

- Display the status of the job with the job ID of `9`.

status -i 9

- Display the status of the second occurrence of the job alias **34**.

  status -a 34 -o 2

submit

The **submit** command replaces predone job or job group instances in the production schedule according to its calendar. If there are job or job group instances already completed, these instances are not replaced

Syntax

SACmd submit -i ID|-a alias [-r Y|N] [-e yyyymmdd]

submit -options

Option	Description
-i ID	The rule ID of the job or job group to replace. To obtain rule ID's use the listrule command. Either this option or **-a alias** is required.
-a alias	The alias of the job or job group to submit. Either this option or **-i ID** is required.
-r Y | N	Specifies whether to submit repeating jobs immediately. If omitted, this defaults to **N**.
-e yyyymmdd	Specifies a date in the production schedule in which the job is to be submitted. If omitted, this defaults to the first production date as specified by its calendar.

Examples

- The following example submits the job with the alias name **newjob2**.

  SACmd submit -a newjob2

- The following example submits a job with the rule ID **2354**.

  SACmd submit -i 2345

useradd

Warning: Always enclose arguments containing spaces in quotes, or the command will not execute successfully. For example, if the folder Program Files is included in a path statement as an argument, the entire path must be enclosed by quotes.

The **useradd** command allows you to add a new user. The user name is a required parameter.

Syntax

SACmd useradd user/group name [-d domainname | -p Windowspassword |

-f fullname | -g Y|N | -r "Y" or "N" or "R" |-s security_profile_name

useradd -options

Option	Description
-d domainname	Specifies the user domain.
-p Windows/FTP/ OS400 password	Specifies the Windows/FTP/OS400 password for the new user.
-f fullname	The user's full name. If omitted, this defaults to the user/group name.
-g Y|N	Specifies whether this user is a group (**Y**) or a user (**N**). If unspecified this defaults to user.
-r Y|N|R	Specifies what type of user. If **R**, then the user is a runtime user only. If **Y**, then the user is a superuser. If **N**, then the user is not a superuser. If unspecified then this defaults to not a superuser.
-s security_profile_name	Specifies the name of the security profile for the user. If omitted; this defaults to the Operator profile.

Operation

You can add a new user only if you have the required security privileges for adding users.

varset

Warning: Always enclose arguments containing spaces in quotes, or the command will not execute successfully. For example, if the folder Program Files is included in a path statement as an argument, the entire path must be enclosed by quotes.

The **varset** command lets you manually set the value of an already existing variable.

Syntax

SACmd varset -i <group jobrun ID> -n <variable name> [-v <variable value> | -r <TES variable>]

varset -options

Option	Description
-i id	The Job Run ID for a group whose local variable is to be updated.
-n name	The name of the variable whose value is to be reset. This option is required
-v value	The new value for the variable. This option is required if a -r TES variable value is not specified.
-r TES variable	Indicates that the new value for the variable is contained in another TES variable and the value has to be resolved first. This option is required if the option " -v" is not used.

Operation

You can set a value for any variable for which you have editing privileges. The variable should be specified in the format required by its type (**DATE**, **STRING** or **NUMBER**).

varset

3

Using the Master Command Line Utility

Overview

CWA offers a master-side admin command line utility on Windows and Unix you can use to perform certain configuration updates to a database without having the master and client manager running. This utility communicates directly with database using JDBC APIs. It can only be accessed from the base machine that the master is installed on and you do not need to provide the name of the machine in the command.

In addition to configuring updates to your database without the Master and Client Manager running, this utility offers the following support for:

- enabling/disabling fault tolerance.
- managing database users and passwords.
- setting the log levels in the Master before startup.
- enabling/disabling JMX in the Master.
- stopping/resuming of all jobs launching.

Understanding the tesm command for the Master CLI

For Cisco Workload Automation 6.2 and later, the **tesm** script has been enhanced to support the features described above. The command line is written for UNIX/Linux as well as Windows in the form of a batch file script.

Options for TESM command

The following are the options for running **tesm** on the Master command line:

cm

Syntax

cm setcmpwd pluginname oldpassword newpassword

Operation

This command sets the plugin name and password for the Client Manager.

Example

- The following command sets the plugin name and password for the Client Manager.

  cm setcmpwd tes-6.0.dsp tidal123 tidalabc

enableJmx/disableJmx

Syntax

tesm enableJmx/tesm disableJmx

Operation

This command either enables or disables JMX on the master without the master running.

Examples

- The following command enables JMX on the master.

  tesm enableJmx

- The following command disables JMX on the master.

  tesm disableJmx

ft

Syntax

tesm ft on/off

Operation

Configures fault tolerance to either on or off by updating the Sysval table in the database. If the flag is set to **on**, then it checks to verify if the backup master and fault monitor exists.

Examples

- The following command turns fault tolerance on.

  tesm ft on

- The following command turns fault tolerance off.

  tesm ft off

keystoremgr

Syntax

tesm keystoremgr -svc <service> -reload

Operation

This command carries out management tasks on the keystore(s) maintained by CWA. The following options are supported:

Examples

- **-svc <service>** specifies service name to target the command to a particular Adapter service. This is the base name of Adapter service package. For example, for "*webservice.pkg*", the base name is "webservice".

  Use the value ALL in place of <service> to broadcast the command to all active adapters. For example, **-svc ALL**

- **-reload** instructs the recipient(s) to reload keystore.

loglevel

Syntax

tesm loglevel –D logvalue

Operation

Updates the log level in the database for user to debug the master. This command will work even when the master is not running.

Examples

- The following command updates the log level in the database for debugging purposes to *Severe*.

  The following levels are available:

  - SEVERE
  - WARNING
  - INFO
  - LOW
  - MEDIUM
  - HIGH

pause

Syntax

tesm pause

Operation

Pauses the production schedule. No waiting jobs will run; even if their dependencies are met. The flag in the database is also updated. This command works when the master is not running by using JDBC to update the database to reflect this change.

Examples

- The following command pauses the production schedule.

  tesm pause

resume

Syntax

tesm resume

Operation

Resumes the production schedule after being paused. The flag in the database table sysval will be updated to reflect this status. This command will work even when the master is not running.

Examples

- The following command resumes the production schedule after being paused.

  tesm resume

setpwd

Syntax

tesm setpwd old password new password or test setpwd

Operation

Allows you to change the database password, even if the master is not running. This command updates the database directly when the master is not running to change the database password.

Examples

- The following command changes the old password (tidal123) to the new password (tidalabc).

  tesm setpwd tidal123 tidalabc

- The following command tests a password.

  test setpwd

superuseradd

Syntax

tesm superuseradd userid domainname

Operation

This command adds a superuser and domain to the master.

Examples

- The following command adds the superuser, tidaluser, and the domain, tidalsoft, to the master.

  tesm superuseradd tidaluser tidalsoft

systemvalue

Syntax

tesm systemvalue [-i ID] [-string value] [-integer value]

where:

-i ID—The ID of the sysval. Specify ALL to display values for all sysvals.

-string value—The string to use for sysval_string.

-integer value—The integer value to use for sysval_integer.

Operation

The systemvalue command lists or sets system values while the master is running. Only certain supported system values can be set this way. To get a list of system values that can be updated with this command, execute the command without any arguments. If you only specify an ID without any new values, then the current values for that sysval will be returned. If you specify an ID of ALL without any new values, then the current values for all defined sysvals will be returned, including sysvals that cannot be udpated with this command.

4

Customizing the Command Line Program

Overview

The Command Line Program (that is, sacmd.cmd (Windows) or sacmd.sh (Unix/Linux) is equipped with the following two property files that allow you to customize its behavior:

- tescmd.props, page 75
- table.props, page 76

Both files can be found in the config directory under the installation directory.

tescmd.props

The *tescmd.props* property file defines properties involving runtime environment and visual effects.

Property Name	Description
JAVA_HOME	Specifies a directory path pointing to the Java Runtime for launching the SACmd. This property is set up by the InstallShield installer of the *sacmd.cmd* (Windows) or *sacmd.sh* (Unix/Linux) program.
UseUnixId	Specifies whether to execute Unix 'id' command to gather user info. Default is false. The *sacmd.cmd* (Windows) or *sacmd.sh* (Unix/Linux) program has the ability to persist credential you enter for connecting to the Client Manager so you don't need to enter again. This is especially useful if you plan to run *sacmd.cmd* (Windows) or *sacmd.sh* (Unix/Linux) in batch mode (referred to as Single Command Mode, page 10 in this guide) within your batch script. The program uses context information unique to the login user to encrypt sensitive data so no one else can comprise its secrecy. By default, this user info is gathered using javax.security.auth.login.LoginContext API. If the runtime platform is a Unix/Linux system and, for any reason, it does not fully support this API, you can set this property to true and the *sacmd.cmd* (Windows) or *sacmd.sh* (Unix/Linux) program will gather user info by the 'id' command instead.
Prompt	Specifies the text prompt to display at startup, and upon completion of each command execution. Default is **`SACmd`**.

Property Name	Description
PageHeight	Specifies the number of lines to display per page. Default is 50.
Style	Specifies the style in which to display command results. Valid styles are: table list xml Default is table. You can change it to any of the three values depending on your need. For example, you may choose xml style if you wish to feed the command result into your application capable of processing the xml document.
javax.net.ssl.trustStore	Specifies fully qualified path of the SSL trust store to be used if you wish to connect SACmd to a SSL enabled Client Manager. This property is commented out by default.

table.props

This property file defines formatting properties for table style command results. It is effective only if value of the **`Style`** property in *tescmd.props* is set to **`table`**. For each listed SACmd, you can customize columns to display as well as title, width and alignment of each column.

For example, to change alignment of the column **`active`** of the **`agent -C LIST`** command to **`left`** (default is " center"), you can specify the following:

agent.list.active.align=left

Note: You should not modify this file directly. Editing this file will not have any effect on the SACmd. Instead, create a custom property file by following the procedure described in the header of the table.props. When starting SACmd, supply this custom property file by using the -table_props option.