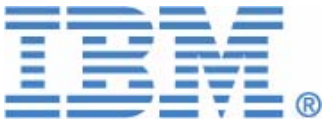
Rational. Statemate

Command Line Reference Guide

Rational Statemate Command Line Refrence Guide



Before using the information in this manual, be sure to read the “Notices” section of the Help or the PDF file available from **Help > List of Books**.

This edition applies to IBM® Rational® Statemate® 4.6 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Command Line Overview 1

Setting Environment Variables 1

Abbreviations and Shortcuts 2

Getting Command Help 3

Commands 5

Backup a Workarea 5

Check-In Operation 5

Check Out Operation 6

Dump Configuration 6

Execute Configuration 6

Execute Commands from File 7

Global Rename 7

Import into Workarea 7

Open Chart 8

Open a Chart File 8

Open GDS 8

Open Project 9

Print Chart 9

Run CheckModel 10

Run Code Generator 10

Run Documentor 11

Run Embedded Rapid Prototyper 11

Run Simulation 12

Command Line Overview

The IBM® Rational® Statemate® command line interface permits diirect entry of commands on the command line. This feature allows you to run the Rational Statemate tools in *batch mode*.

Note

The commands and flags are not case sensitive; you can enter either upper or lower case commands.

Each command can use the optional flag `-terminate`. When this flag is specified, Rational Statemate is terminated when the command has completed its run.

There is no dependency between commands. Each can be executed independently.

Setting Environment Variables

It is possible to set two environment variables to make use of the command line simple:

```
STM_CLI_FLAGS and STM_USER_CLI_FLAGS
```

The value of these variables is added to the command invocation `statemate`.

For example:

By default, in `run_stmm.bat`, the environment variable `STM_CLI_FLAGS` is set to:

```
-appname stm
```

Therefore, you do not need to supply an `appname` with every command.

In addition, you can add your own environment setting to some commonly used flags such as `tyourown` workarea:

```
setenv STM_USER_CLI_FLAGS "-wa /net/lily/disk1/wa21"
```

Abbreviations and Shortcuts

The following table lists abbreviations that can be used in the command line interface.

Command Purpose	Abbreviation
Statemate	stm
Project	prj
Databank	db
Workarea	wa
Simulation	sim
Command	cmd
Checkmodel	chm
Documentor	doc
Printchart	pch
PROFILE‘	PROF
CCODEGEN	CGEN
ADACODEGEN	ADAGEN
TERMINATE	TERM
OUTPUT	O
CHECKIN	CI
CHECKOUT	CO
EXECUTE_CONFIG	EXECUTE
DUMP_CONFIG	DUMP
TYPE	FILETYPE
FILE	FILEPATH
VERSION	VER
RELEASE	REL
DELETE	DEL
WITH_GENERICS	WITH_GEN
WITH_DESCENDANTS	WITH_DESC

Getting Command Help

`Run_stmm -help` should provide help on the possible command options.

Commands

Backup a Workarea

Syntax:

```
Run_stmm -workarea <path to workarea> -backup <path to backup directory>
```

Semantics:

Executes the backup utility. The specified workarea is backed up to the specified backup directory. No questions are asked by Rational Statemate (about existing files in the directory) and all the files in the backup directory are overwritten.

Check-In Operation

Syntax:

```
run_stmm -workarea <path to workarea> -checkin -type <file/chart type>
[-lock/-rel/-del] -name <file/chart name> [-all] [-with_ descendants]
```

Semantics:

Checks in a chart or file from the workarea into the databank.

When no mode is specified (`lock/rel/del`), the default check-in remains locked.

When `-all` is specified, all of the workarea content is checked in.

Check Out Operation

Syntax:

```
run_stmm -workarea <path to workarea> -checkout [-lock/-rel/-del]
[-with_generics] [-with_descendants] [-version <version number>] -type
<file/chart type> -name <file/chart name>
```

Semantics:

Checks out a chart or file from the databank into the workarea.

The default check out type should be without lock. When no version is specified, the latest version is checked out. It is possible to specify with generics and/or with descendants.

Dump Configuration

Syntax:

```
run_stmm -workarea <path to workarea> -dump <configuration file name>
[-version <version number>]
```

Semantics:

Dumps the specified configuration file. When no version is specified, the latest version is used.

Execute Configuration

Syntax:

```
run_stmm -workarea <path to workarea> -execute_config
<configuration file name>
```

Semantics:

Executes the specified configuration file.

Execute Commands from File

Syntax:

```
run_stmm-input <file name>
```

Semantics:

Executes all the commands found in the specified file. The format of the commands should be as specified in this document.This option enables long arguments lists and is a work-around for restricted command line interpreters such as DOS.

Global Rename

Syntax:

```
run_stmm -workarea <path to workarea> -rename <string1> -with <string2>
```

Semantics:

Renames occurrences of string1 with string2 in the entire workarea.

Import into Workarea

Syntax:

```
run_stmm -workarea <path to workarea> -import -type <file/chart type>
[-file <file name>] [-project <project name>] [-databank <databank path>]
[-version <version number>] -name <file/chartname>
```

Semantics:

Imports a chart or file from a file, a project or a databank. One of the flags, file, project, or databank, must be specified. When no version is specified, the latest version is imported.

Open Chart

Syntax:

```
run_stmm -workarea <path to workarea> -chart <chart names>
```

Semantics:

Opens Rational Statemate on the specified workarea and the graphic editor on the specified charts.

The chart names might be an activity chart, a state chart, or module chart names. A comma separates chart names (for example: `-chart ach`)

Open a Chart File

Syntax:

```
run_stmm<chart file names>
```

Semantics:

Enables you to invoke Rational Statemate and open the graphic editor on the specified chart file.

The file should be an ASCII file as saved in the databank. The file is loaded into a temporary workarea created in the file's directory. When exiting Rational Statemate, the workarea is deleted. Legal chart files are graphical (ach, sch, mch) and dictionary (GDS) chart files.

Open GDS

Syntax:

```
run_stmm -workarea <path to workarea> -gds <GDS names>
```

Semantics:

Opens Rational Statemate on the specified workarea and the properties editor on the specified Global Definition Set (GDS).

Open Project

Syntax:

```
un_stmm [-create] [-project <project name>] [-databank <path to databank>]
-workarea <path to workarea>
```

Semantics:

Opens the project on the given workarea. Since a workarea defines a unique project, the project name is optional. If the given project does not exist, and the flag `-create` was specified, a new project is created and opened. In this case, the data-bank flag must be specified. When `-create` is not specified, an error is issued.

Note

The `-create` flag should be the first flag to this command (if used).

Print Chart

Syntax:

```
run_stmm -workarea <path to workarea> [-printall] [-printchart <chart names>]
[-o <output file>] [-device <device name>]
```

Semantics:

Prints the specified charts. If a device name was not specified, the default postscript device is used. The chart names might be an activity chart, a state chart, or module chart names. A comma separates chart names (for example: `-printchart ach, sch, mch`).

When an output file is specified, the chart is printed into this file. When no output file is specified, the chart is printed into `chart_name.eps` under the workarea directory.

If multiple charts are specified, along with an output file, the specified file name is used and extended with the chart name. If `-printall` is specified, all the charts in the workarea are printed. In this case, -`printchart` cannot be used.

Run CheckModel

Syntax:

```
run_stmm -workarea <path to workarea> -checkmodel <profile name>
[-o <output file>]
```

Semantics:

Starts the check model on the specified profile. If the specified profile exists, the check model runs the test. If the profile does not exist, the check model pro-file manager opens on a new profile. If an output file is specified, the check model results are written into the file. If no output file is specified, the results are written to the standard output.

When running this command, you can use the "-msg" option to print the run summary to the specified output file.

Run Code Generator

Syntax:

```
run_stmm -workarea <path to workarea> [-cgen] [-adagen] <profile name>
[-command <command name>] [-outdir <output directory>] [-input <input file>]
[-o <output file>]
```

Semantics:

Starts the code generator on the specified profile. Either `cgen` or `adagen` must be specified to define the code generator to be invoked. If the specified profile does not exist, the code generator profile manager opens on a new profile. If the profile does exist, it opens the profile.

It is optional to specify a command to run. Legal commands are: `generate`, `make`, `run`, and `all`. In these cases, the specified command runs: generate code, make code, run code, or all three, one after the other. When an outdir is specified, the generated code is located in this directory. When not specified, the usual `prt/profile_name` directory is used. If an output file is specified, the code generator output is written into the file. If no output file is specified, the output is written to the standard output. If an input file is specified, it is used to execute commands in the debugger.

Run Documentor

Syntax:

```
run_stmm -workarea <path to workarea> -documentor [<template name>]
```

Document generation:

```
run_stmm -workarea <workarea_path> -documentor [-template <template_name>]
-docname <document_name> [-cmd "<parameter1=value1>...."]
[-output <filename>] [-terminate]
```

Document export (Save as..):

```
run_stmm -workarea <workarea_path> -documentor -docname <document_name>
-saveas <file_name> [-output <filename>] [-terminate]
```

Semantics:

Starts the documentor for generating a document. If a template is specified, this is used to generate the document. When no template is specified, the standard Word template is used. The document name must be supplied. Additionally, it is possible to specify values for the template parameters using the "`cmd`" flag.

Run Embedded Rapid Prototyper

Syntax:

```
run_stmm -workarea <path to workarea> -rapid <profile name>
[-command <command name>] [-o <output file>]
```

Semantics:

Starts the code generator on the specified profile. Either `cgen` or `adagen` must be specified to define the code generator to be invoked. If the specified profile does not exist, the code generator profile manager opens on a new profile. If the profile does exist, it opens the profile.

It is optional to specify a command to run. Legal commands are: `generate`, `make`, `run`, and `all`. In these cases, the specified command runs generate code, make code, run code, or all three, one after the other. When an outdir is specified, the generated code is located in this directory. When not specified, the usual `prt/profile_name` directory is used. If an output file is specified, the code generator output is written into the file. If no output file is specified, the output is written to the standard output. If an input file is specified, it is used to execute commands in the debugger.

Run Simulation

Syntax:

```
run_stmm -workarea <path to workarea> -simulation <profile name>
[-chart <chart names>] [-command <command name>] [-input <input file>]
[-o <output file>]
```

Semantics:

Starts the simulation on the specified profile. If the specified profile exists, the run window of the simulation is invoked. If the profile does not exist, the simulation profile manager opens on a new profile.

When charts are specified, they are opened in read mode and are animated by the simulation. You can use a command to run in the simulation run window. This command can be any legal command that can be run from the simulation command line. You can also specify an input file with a list of commands to be executed in the simulation. The format of the input file should be a line per command. The command's syntax is the one that is used in the simulation command line. A line beginning with the # sign is ignored by the simulation, and serve as a remark. Empty lines are legal.

The output file, when specified, is used to record all outputs of the simulation run (for example, messages and output of reports such as show changes).