Technical Documentation Version 7.1

Batch Mode and RCL

Batch Mode and RCL Table of Contents

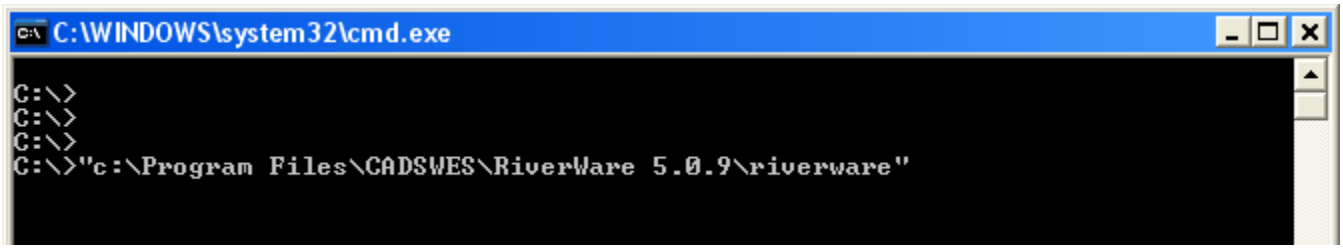
Batch Mode and RCL

1. Introduction

This section describes methodology to run RiverWare from the command line and optionally execute a RiverWare Command Language (RCL) script. This Batch Mode allows you to run RiverWare without seeing the user interface and allows for automation of many processes including advancing the run dates, running DMIs to bring in new data, running the model, generating output, and saving the model.

2. Running RiverWare from the Command Line

Running RiverWare from the command line provides an alternative to clicking on the RiverWare icon from the Windows system. In addition, RiverWare supports certain commands directly from the command line like loading models and rulesets and certain other operations. To access the command line on windows, click the Start Menu -> Run. Then type in `cmd`. A window similar to the following opens. To start RiverWare from this window, you need to type in the location of RiverWare (depending on your PATH configuration)



A complete list of command line options can be viewed by invoking Riverware with the `--help` command line argument. On Windows, in a command line window (cmd), this looks like this:

```
riverware.exe --help
```

When RiverWare is executed from a command line window, it is run as an independent process from the command shell, so exiting the command window will not automatically terminate RiverWare. The Windows task manager can be used to terminate batch runs of RiverWare.

Sometimes, the user wishes to run one or more models as automatic tasks (as Windows “Scheduled Tasks”). Setting up the runs using batch mode and the command line options can be useful in this approach. For example, a user may wish to run a series of models overnight when the machine is more available. One command line option that is useful is the --log <file> switch. This option sends any diagnostic output to the specified file.

3. Running RiverWare in Batch Mode

The RiverWare Command Language (RCL) facility allows models to be run in batch mode. While in this mode, RiverWare does not open a user interface, but instead runs in background mode.

In batch mode, the user provides RiverWare with a script file, which contains the commands RiverWare executes. Commands are entered through the RCL file, rather than through the user interface. The commands are described in detail in the following document.

The syntax to invoke RiverWare in batch mode from a command prompt is:

```
riverware --batch <script file>
```

The <script file> defines how the batch run should proceed. In the file, you specify commands to load a model, load a ruleset if necessary, start a run, and save the results. In the simplest case, you will load a model, run a simulation, save the model, and exit batch mode. The following RiverWare Command Language commands support these actions:

```
# Load the model
OpenWorkspace <Model Name>
# run the simulation
StartController
# save the model
SaveWorkspace <Model Name>.saved.mdl
# Close the opened model and exit RiverWare
CloseWorkspace
```

This is a very simple example. The full complement of key words and commands is presented in the following section.

4. RCL: The RiverWare Command Language

4.1 Key

- Angled brackets (<>) indicate a required parameter.
- Brackets ([]) indicate an optional parameter.
- A vertical bar "|" within angled brackets or brackets indicates a choice of parameters. For example, "<#RunInfo|#MRM>" indicates that either "#RunInfo" or "#MRM" must be provided as a required parameter.
- Parameters starting with "!" are tokens which should appear exactly as is.
- Parameters starting with a "#" are RiverWare names which should appear exactly as is.

- A “#” sign at the beginning of the line represents a comment. Anything occurring after the “#” will be ignored.
- Parameters which contain embedded white space characters must be enclosed in curly braces{ }, or quotes “ ”.
- Anywhere there is a file referenced below, the user can specify the path using either a full reference or using environment variables (see next section). On Windows, path names should use “/” or '\\'. Also, the file may reference a shared network: RCL accepts '\' but it has to be escaped with '\', so the RCL syntax is:

  \\\\ComputerName\\ShareName\\...

4.2 Environment Variables

Environment variable are “system variables” that allow portability of models across machines and platforms.

4.2.1 Setting Environment Variables

If you wish to set environment variables and then reference them in the RCL script, they can be set using the TCL command “set env” as follows:

```
set env(variable) value
```

Then the variable can be accessed within the RCL script using the $env(variable) syntax. These variables are only available within the script, RiverWare has already inherited the environment before this is set and are then not available. To set variables to be available to RiverWare, use the **`SetEnv`** command described **HERE (Section 4.4.23)**. It has the syntax:

```
SetEnv <variable> <value>
```

These variables are then available in RiverWare but not available within the script. Most likely, you will want to set both variables within the script. Consider the following example:

```
# for use in the script:
set env(DMI_DIR) C:/DMI/data
# for use by RiverWare:
SetEnv DMI_DIR $env(DMI_DIR)
# open the model using the TCL environment variable syntax:
OpenWorkspace $env(DMI_DIR)/model.mdl.gz
```

Then within RiverWare, you can reference DMI_DIR using the $DMI_DIR syntax in DMI’s, MRM or other places.

4.2.2 Using Environment Variables

Environment variables set outside of the RCL script can be referenced using the form:

- $VARIABLE
- $(VARIABLE)
- ${VARIABLE}

VARIABLE is a letter followed by zero or more letters, digits or underscores. If the variable is followed by a letter, digit or underscore it must be “quoted” with () or {}.

4.3 Special Parameters

- `{date time}`

  A date and time of the format {month/day/year hour:minute}, where the year includes the century and the hour is 24-hour based. Note that a date and time, which includes embedded white space, must be enclosed in braces.

- `{delta time}`

  A timestep or time duration of the format {count units}, where units are “HOURS,” “DAYS,” “WEEKS,” and “MONTHS.” Note that a delta time, which includes embedded white space, must be enclosed in braces.

4.4 Commands

Following are the RCL commands in alphabetical order:

4.4.1 CloseWorkspace

Syntax: `CloseWorkspace`

Clears the workspace.

4.4.2 CreateSlotCache

Syntax: `CreateSlotCache`

Creates a slot cache that contains the current values of all series slots on the workspace, in the range of the current controller. The cache is described **HERE (Workspace.pdf, Section 5.9)**.

Note, the slot cache is under development. Please contact riverware-support@colorado.edu for more information and the current status of this feature.

4.4.3 EnableInfoDiag

Syntax: `EnableInfoDiag <1|0>`

Enable Informational Diagnostics in the diagnostics manager dialog. When the argument is 1, diagnostics will be enabled. When the argument is zero, then the diagnostics will be disabled.

4.4.4 ExecuteScript

Syntax: `ExecuteScript <Script Name>`

Executes the specified script as defined in the Script Manager. For more information on RiverWare Script Management, click **HERE (ScriptManagement.pdf, Section 1)**.

4.4.5 ExportOptAnalysisAsCSV

Syntax: `ExportOptAnalysisAsCSV <file name> [<section name> true|false true|false]`

Exports the Optimization Analsyis information as a comma separated values file. The arguments are as follows:

- `<file name>` is the name of the file to write.
- `<section name>` are the (optional) sections to export. Defaults to all sections. The section names are listed below and correspond to combinations of the sections **HERE (Optimization.pdf, Section 7.1.6)**:
  - None
  - SolutionSingle
  - SolutionGoalSummary
  - ConstraintNew
  - ConstraintPrior
  - ConstraintPhysical
  - VariableSlot
  - VariableOther
  - SolutionInfo
  - ConstraintPolicy
  - SolutionAndPolicy
  - ConstraintAll
  - VariableAll
- the first optional `true|false` specifies whether to skip export of constraint and variable rows associated with fully satisfied constraints or solutions. Defaults to true, i.e., do not export these rows for fully satisfied solutions. Note that if this argument is present, all previous arguments must also be present.
- the second optional `true|false` specifies whether to skip export of constraint and variable rows associated with the solution to the postponed problem, which is relevant only for runs in which a seed value was used to postpone one or more solutions. Defaults to true, i.e., do not export these rows for postponed problem solutions. Note that if this argument is present, all previous arguments must also be present.

4.4.6 GetRunInfo

Syntax: `GetRunInfo <#RunInfo|#MRM <MRM configuration>>`
`[!InitDate] [!EndDate]`
`[!Duration] [!Step] [!Controller]`

Prints single run information (#RunInfo) or multiple run information (#MRM) to standard output. If #MRM is specified, you also need to specify an MRM configuration <MRM configuration>.

- !InitDate prints the run's initial date.
- !EndDate prints the run's end date.
- !Duration prints the run's duration.
- !Step prints the run's timestep.

- !Controller prints the current controller.

If no parameters are specified, all information is printed.

4.4.7 GetSlot

Syntax: `GetSlot <Object.Slot> [<date time>]`

Returns the series or scalar slot's value, in user units, as a Tcl variable. Object.Slot should be enclosed in {} if it contains embedded white space.

<date time> is only for series slots and must be enclosed in curly braces, {}, as it contains embedded white space - for example, {05-15-2010 12:00}.

The command is used when writing batch mode scripts which make use of the Tcl scripting language. GetSlot returns the value in a Tcl variable, so sample usage includes:

```
set v [GetSlot {Object.Slot} {05-05-2010 12:00}]
SetSlot {Object.Slot} {05-05-2010 12:00} $v
```

The Tcl variable can be used in arithmetic expressions, conditionals, etc:

```
set v [expr 0.5 * $v]
if {$v > 1.0} {
  ...
} else {
  ...
}
```

Further scripting help with the TCL language is beyond the scope of this reference but can be found in online searches.

4.4.8 HDB_EnsembleID

Syntax: `HDB_EnsembleID <HDB dataset name> <Ensemble ID> ...`

Sets the Ensemble ID for an HDB dataset that is configured to have its ID selected when MRM is started.

- `<HDB dataset name>` is the name of the HDB dataset that is configured to have its ID selected when MRM is started. If the name contains embedded white space, it must be enclosed in braces.
- `<Ensemble ID>` is the Ensemble ID to set on the dataset.

IDs can be set for more than one dataset by adding addition pairs of names and IDs.

4.4.9 HDB_ModelRunID

Syntax: `HDB_ModelRunID <HDB dataset name> <Model Run ID> ...`

Sets the model run ID for an HDB dataset that is configured to have its ID selected when its DMI is invoked.

- <HDB dataset name> is the name of the HDB dataset that is configured to have its model run ID selected when its DMI is invoked. If the name contains embedded white space, it must be enclosed in braces.
- <Model Run ID> is the Model Run ID to set on the dataset.

IDs can be set for more than one dataset by adding addition pairs of names and IDs.

4.4.10 InvokeDMI

Syntax: `InvokeDMI <DMI name> [-UuserParam=value]`

Invokes a DMI.

- <DMI name> is the name of the DMI to invoke. If the name contains embedded white space, it must be enclosed in braces.
- [-UuserParam=value] sets the user parameter “userParam” to “value”.

4.4.11 InvokeDssDMI

Syntax: `InvokeDssDMI <DMI name> !DssFile <DSS file> !DssFPart <DSS F part>`

Invokes a DSS DMI.

- <DMI name> is the name of the DSS DMI to invoke. If the name contains embedded white space, it must be enclosed in braces.
- <DSS file> is the name of the DSS file. If the file name contains embedded white space, it must be enclosed in braces.
- <DSS F part> is the DSS F part. If the F part contains embedded white space, it must be enclosed in braces.

4.4.12 ListDMI

Syntax: `ListDMI [!Type <Input|Output>] [!Dataset <dataset type>]<output file>`

Writes the names of the DMIs which match the specified criteria to the output file.

- [!Type <Input|Output>] limits the output to either Input or Output DMIs.
- [!Dataset <dataset type>] limits the output to DMIs which use only the specified dataset type.
- <output file> is the file the DMI list is written to.

For example:

- ListDMI file

Writes the names of all DMIs to the file.

- ListDMI !Type Input file

Write the names of all Input DMIs to the file.

- ListDMI !Type Input !Dataset DSS file

Write the names of all Input DMIs which use only DSS datasets to the file.

4.4.13 LoadConstraints

Syntax: `LoadConstraints <file path> [!Append]`

Loads the constraint set into the workspace. By default, constraint sets which are already loaded into the workspace are cleared.

- <file path> is the path to where the constraint set resides on disk.
- !Append does not clear constraint sets which are already loaded into the workspace.

4.4.14 LoadOptSet

Syntax: `LoadOptSet <file path>`

Loads the optimization ruleset into the workspace.

<file path> is the path to where the optimization ruleset resides on disk.

4.4.15 LoadRules

Syntax: `LoadRules <file path>`

Loads a ruleset into the workspace.

- <file path> is the path to where the ruleset resides on disk.

4.4.16 OpenGlobalSet

Syntax: `OpenGlobalSet <file path>`

Opens a global function set. Note, this must be called before LoadRules if the ruleset references functions in the global set.

- <file path> is the path to where the set resides on disk.

4.4.17 OpenWorkspace

Syntax: `OpenWorkspace <file path>`

Loads a model into the workspace.

- <file path> is the path to where the model resides on disk.

4.4.18 Output

Syntax: `Output <outputDeviceName>`

Generates an output file that is specified in the Output Manager of the model. Plot pages cannot be generated.

- <outputDeviceName> is the name of the output device saved in the Output Manager of the model.

4.4.19 RequireVersion

Syntax: `RequireVersion <major version>.<minor version>[.<patch level>]`

Specifies the RiverWare version required to execute the batch script. If the RiverWare version is earlier than the required version, the command issues an error message and stops the script file's execution.

- <major version>.<minor version>[.<patch level>] are the major version, minor version and optional patch level of RiverWare necessary to execute this batch script.

4.4.20 ReorderGoalSet

Syntax: `ReorderGoalSet <order slot>`

This command allows you to reorder items (goals, groups, or certain statements) within the currently loaded optimization goal specified set. The desired order is specified by the single column order slot, a table slot, each of whose rows corresponds to the name of an item in the goal set. The values in the single column of the order slot are interpreted as integers giving the desired order of that object with respect to other set items referred to in the table. This command is similar to the **Reorder RPL Set** Script action discussed in the Script Management section **HERE (ScriptManagement.pdf, Section 3.3)**.

4.4.21 SaveWorkspace

Syntax: `SaveWorkspace [file path]`

Saves the model which is currently loaded into the workspace.

- [file path] is the path to where the model is to be saved on disk. If the file path is not specified, the OpenWorkspace file path is used.

4.4.22 SetDiagFile

Syntax: `SetDiagFile <diagnostic file>`

Sets the location of the diagnostic output file.

- <diagnostic file> is the path to where the diagnostic file will be written.

4.4.23 SetEnv

Syntax: `SetEnv <variable> <value>`

Sets an environment variable for use within the RiverWare run.

See the section **HERE (Section 4.2)** for more information on setting and using environment variables.

4.4.24 SetRunInfo

Syntax:
```
SetRunInfo <#RunInfo|#MRM <MRM configuration>>
        [!InitDate <date time>]
        [!StartDate <date time>]
        [!EndDate <date time>]
```

```
[!Duration <delta time>]
[!Controller <controller type>]
```

Sets single run information (#RunInfo) or multiple run information (#MRM). If #MRM is specified, you also need to specify an MRM configuration <MRM configuration>.

- !InitDate sets the run's initial date to <date time> of format {MM-DD-YYYY HH:MM}.
- !StartDate sets the run's start date to <date time> of format {MM-DD-YYYY HH:MM}.

  Note, if you specify both !InitDate and !StartDate, the argument which appears second will take precedence.
- !EndDate sets the run's end date to <date time> of format {MM-DD-YYYY HH:MM}.
- !Duration sets the run's duration to <delta time> of format {MM-DD-YYYY HH:MM}.
- !Controller sets the run's controller to <controller type>. The controller type must be one of the controllers available in the Run Control dialog:

  Simulation
  Rulebased Simulation
  Optimization

  Note that controller types which contain embedded white space must be enclosed in braces.

Following are samples of the syntax:

- SetRunInfo #RunInfo !InitDate {04-21-2011 24:00}
- SetRunInfo #RunInfo !StartDate {04-22-2011 24:00}
- SetRunInfo #RunInfo !EndDate {04-26-2011 24:00}
- SetRunInfo #RunInfo !Controller "Optimization"
- SetRunInfo #MRM "Paleo Runs"

4.4.25 SetSlot

Syntax: `SetSlot <Object.Slot> [<date time>] <value>`

Sets the series or scalar slot's value to the value, assumed to be in user scale and units. Object.Slot should be enclosed in {} if it contains embedded white space.

<date time> is only for SeriesSlots and must be enclosed in {} as it contains embedded white space - for example, {05-15-2010 12:00}.

See `GetSlot <Object.Slot> [<date time>]` **HERE (Section 4.4.7)**, for more information on possible usage.

4.4.26 SetTrace

Syntax: `SetTrace <1|0>`

Enable or disable trace messages from the RCL interpreter. The trace messages are prefaced with "TRACE:" and written to standard output; they can be captured in a file with the "--log <log file>" command-line option.

4.4.27 SlotList

Syntax: `SlotList <output file>`

Writes information about all slots in the model to the output file.

- <output file> is the file the slot list is written to.

The information is written as comma-separated values to the output file; the "--help" command line option lists the output fields.

4.4.28 SlotListDMI

Syntax: `SlotListDMI <DMI name> <output file>`

Writes information about the slots to/from which the DMI imports/exports data; the information is written as comma-separated values to the output file. The information includes:

- The slot's name.
- The slot's priority (determined by its dataset association).
- The slot's begin date "mm-dd-yyyy".
- The slot's begin time "hh:mm:ss".
- The slot's end date "mm-dd-yyyy".
- The slot's end time "hh:mm:ss".
- The dataset associated with the slot .
- The dataset's type, currently DSS or HDB. If the Dataset is DSS, then the slot's DSS path "/A/B/C/D/E/F/" is also provided as another comma separated value.

4.4.29 StartController

Syntax: `StartController [!MRM <MRM configuration>]`
`[firstTrace=N]`
`[numTrace=M] ctlFile=file]`

Starts the current RiverWare controller.

- !MRM will start the Multiple Run Management controller instead of the single run controller, using the specified MRM configuration. RCL currently does not support many commands for defining multiple runs, so models should already contain the multiple run definitions.

The following commands override the values in the MRM configuration in the model file ([!MRM <MRM configuration>] must be used if you wish to specify these parameters):

- [firstTrace =N] sets the index sequential initial offset
- [numTrace=M] sets the index sequential number of runs
- [ctlFile=file] sets the output control file.

4.4.30 SyncObj

Syntax: `SyncObj [!Acct] [!ExDiffTS]`
`[!StartDate <start date>]`
`[!EndDate <end date>]`

Synchronize slots to either the run control or the specified dates.

- [!Acct] includes accounting slots.
- [!ExDiffTS] excludes slots whose timestep differs from the run control timestep.
- [!StartDate <start date>] synchronizes slots to the specified start date, rather than the run control initial date.
- [!EndDate <end date>] synchronizes slots to the specified end date, rather than the run control end date.