

Technical Documentation Version 8.0

Automation Tools



Center for Advanced Decision Support for
Water and Environmental Systems (CADSWES)

UNIVERSITY OF COLORADO **BOULDER**

These documents are copyrighted by the Regents of the University of Colorado. No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means electronic, mechanical, recording or otherwise without the prior written consent of The University of Colorado. All rights are reserved by The University of Colorado.

The University of Colorado makes no warranty of any kind with respect to the completeness or accuracy of this document. The University of Colorado may make improvements and/or changes in the product(s) and/or programs described within this document at any time and without notice.

Contents

1. Script Management 1

About Script Management 1

Script Management Uses 2

Managing Scripts 3

Accessing the Script Manager 3

Tour of the Script Manager 3

Working With Script Groups 4

Importing and Exporting Scripts 6

Exporting Scripts 6

Importing Scripts 7

Editing a Script 8

Using the Script Editor 10

About Action Types 11

Common Settings for Action Types 14

Setting Information 14

Datetime Information 14

Filename Information 16

RPL Reorder Set Action 16

Setting Slot Values 18

Configuring User Control on the Script Dashboard 19

Setting Slot Values using a Global Function 21

Action Type Settings Reference 23

Clear DMI Values 23

Clear Scalar Slot Values 23

Clear Table Slot Values 23

Configure MRM Index Sequential 23

Configure MRM Input DMI 24

Configure MRM Output 24

Create Slot Cache 24

Create Snapshot 25

Divider 25

Enable Dispatching 25

Enable MRM Distributed Runs 25

Enable RPL Item 26

Evaluate Expression Slots 26

Execute DMI 26

Execute MRM Run 26

Execute Run 26

Execute Script 26

Generate Output Device 27
Global Time Scroll 27
Load Goal Set 27
Load Ruleset 27
Memo 28
Open File 28
Open Global Functions Set 28
Open Objects 28
Open SCT 29
Open Slots 29
Remove RPL Set 29
Reorder RPL Set 29
Replace Initialization Rules Set From File 29
Save Model 30
Set Account Method 30
Set Accounting Period 30
Set Controller 30
Set Dataset File 31
Set Excel Dataset Run Name 31
Set Init. Rules Exec. Flag 31
Set Method 31
Set MRM Descriptor 32
Set MRM Ruleset 32
Set MRM Run Range 32
Set Run Range 33
Set Run Timestep 33
Set Scalar Slot Value 34
Set Series Slot Flags 35
Set Series Slot Values 35
Set Table Slot Value 36
Synchronize Objects 37
Synchronize Slots 37

Script Dashboard 38

2. Batch Mode and RiverWare Command Language 41

Running RiverWare From the Command Line 41

Running RiverWare in Batch Mode 42

About RiverWare Command Language 43

Syntax Key 43
Environment Variables 43
Setting Environment Variables 43
Using Environment Variables 44
Special Parameters 44

RCL Commands Reference 44

CloseWorkspace 44
ConfigureExcelDataset 45

CreateSlotCache 45
EnableInfoDiag 45
ExecuteScript 45
ExportOptAnalysisAsCSV 46
GetRunInfo 46
GetSlot 47
HDB_EnsembleID 47
HDB_ModelRunID 48
InvokeDMI 48
InvokeDssDMI 48
ListDMI 48
LoadDiagSettings 49
LoadOptSet 49
LoadRules 49
OpenGlobalSet 49
OpenWorkspace 50
Output 50
RequiredInfo 50
RequireVersion 50
ReorderGoalSet 51
SaveRplSet 51
SaveWorkspace 52
SetDiagFile 52
SetEnv 53
SetRunInfo 53
SetSlot 54
SetTrace 54
SlotList 54
SlotListDMI 54
StartController 54
SyncObj 55

Contents

Chapter 1
Script Management

Scripts allow you to organize and run sequences of actions. This allows you to automate many of the tasks involved in operating a model. This section presents key Script features and provides some examples of how they might be used.

Topics

- About Script Management, page 1
- Script Management Uses, page 2
- Managing Scripts, page 3
- Editing a Script, page 8
- Action Type Settings Reference, page 23
- Script Dashboard, page 38

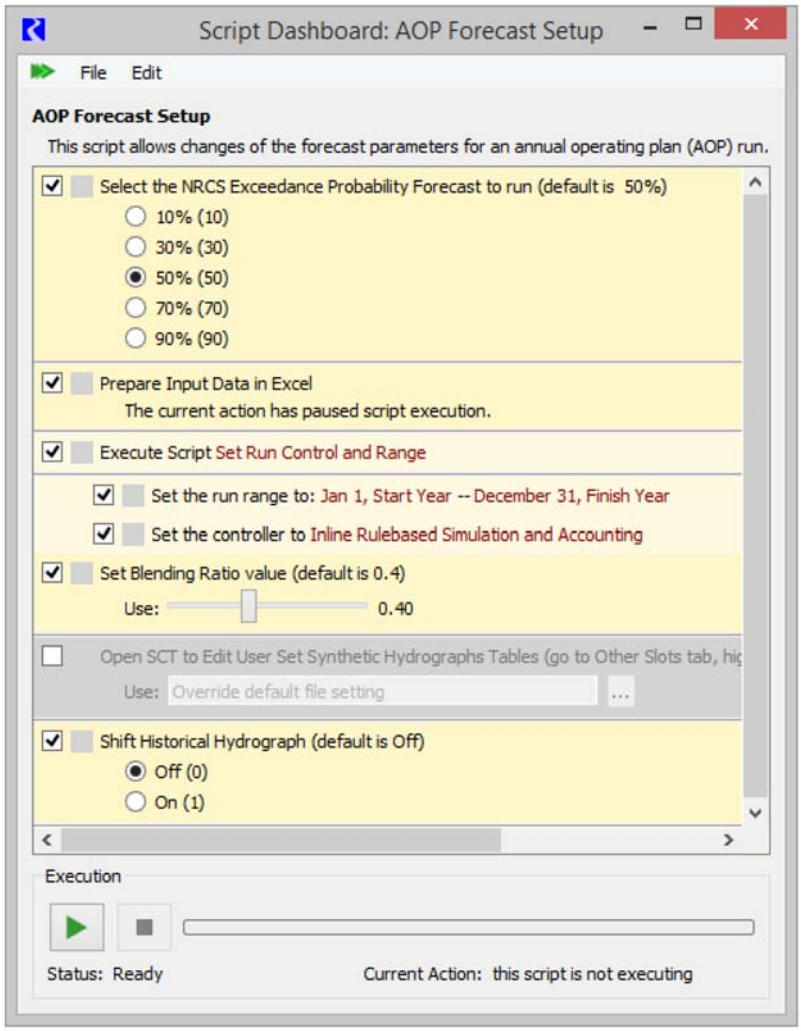
About Script Management

Like the model itself, in a typical application the script will be created and configured once and then executed many times. In our discussion of Scripts, we use the term *modeler* to refer to the person who defines and configures the script and the term *user* for the person who runs the script.

- The modeler creates scripts in the Script Manager and then configures each script in its Script Editor.
- Scripts are composed of a series of *Actions*, each of which represents a single step to be taken by the script, such as setting the run controller, setting a slot value, running a DMI or loading a ruleset.
- After an action has been added to a script in the Script Editor, the precise behavior of the action can be controlled by editing its settings.
- Any action that is not of interest to the user can configured as hidden.
- The Script Dashboard lists the associated script's visible actions and allows the user to override key configuration settings before running the script. Convenient controls such as radio buttons, sliders, and toggles, are provided for manipulating configuration settings.
- Scripts are saved to the model file but can be exported/imported to other models.
- From both the Script Editor and Script Dashboard, right-click context menus allow you to directly open the slot or object dialogs when the action refers to slots or objects.

- A script can execute another script (using the **Execute Script** action). Only one level of a script calling another script is allowed.

For example, the following script changes the run range, sets a value that represents the type of run, changes the controller, sets a date trigger referenced by rules, allows you to select a forecast percentage, changes a method selection, and clears a slot's flags.



Script Management Uses

Now that you know some of the features of scripts, why might you use them? Following are some applications or use scenarios.

- Update a model to a new time range. For example, at the beginning of the year, advance the model's run range, import new data, and set desired operation triggers.
- Prepare a model for the current day's run. In one click of a button, clear yesterday's proposed operations, import observed data, load the necessary rulesets and goal sets so the model is ready to run.

- Change an after-the-fact accounting model into an operations or planning model. Run a script to change the controller, load an alternative ruleset, modify object methods, and set slot values.
- Provide the model to stakeholders who can use the Script Manager to specify their desired inputs. They then run a script that executes the model.

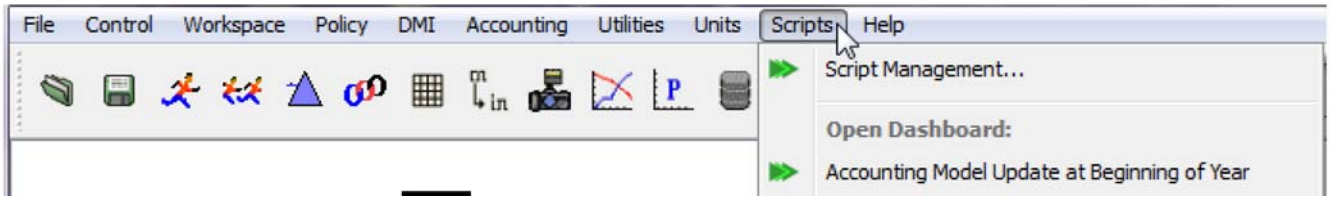
This document describes the Script Manager, Script Editor, and Script Dashboard including all of the action types and settings. See "Script Dashboard" on page 38 for details on running scripts.

Managing Scripts

The Script Manager is used to create, export/import, and delete scripts.

Accessing the Script Manager

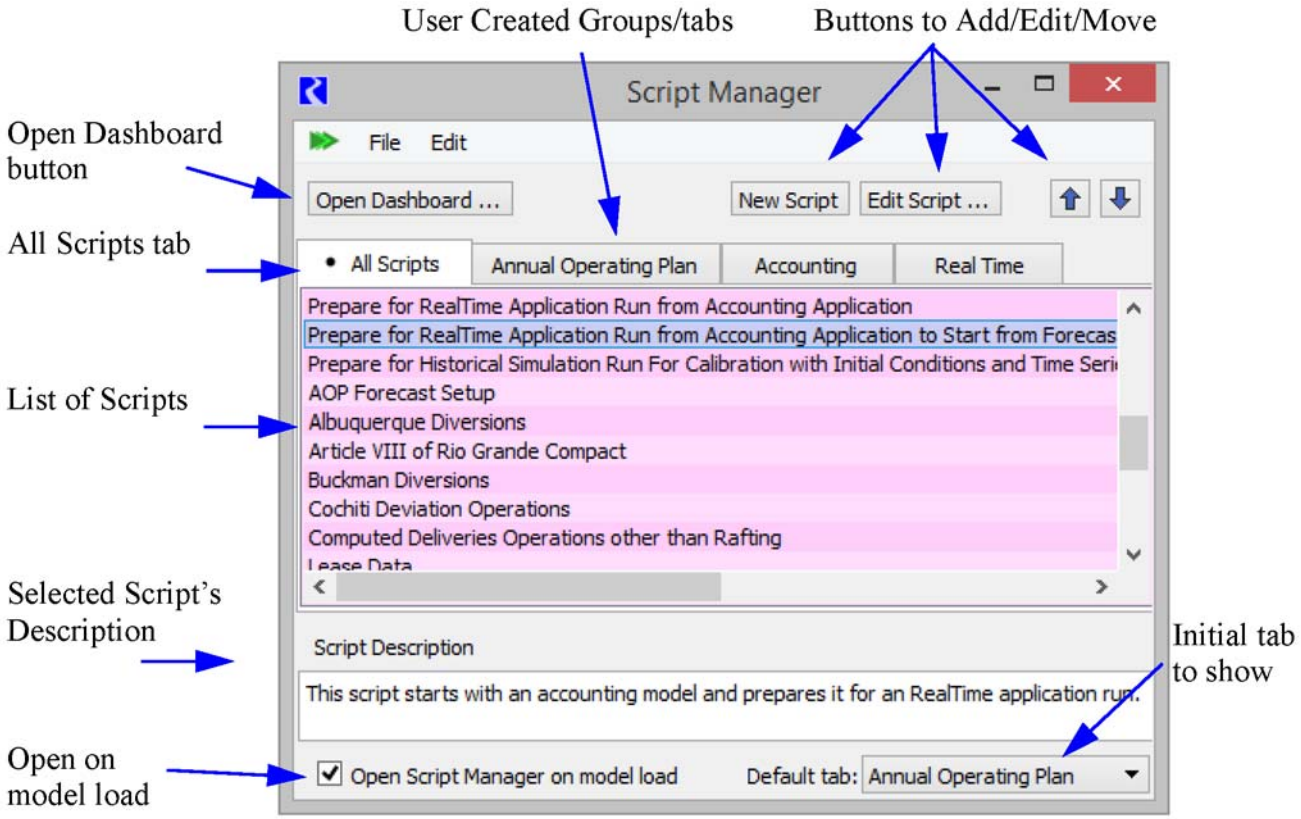
To access the Script Manager, from the workspace, use the **Scripts**, then **Script Management** menu. When there are scripts defined, the name of each script will show up in the Script Management menu. Select the name to open the Script Dashboard for that script.



Tour of the Script Manager

The Script Manager contains lists of the defined scripts in the model. These scripts are all shown on the **All Scripts** tab, as shown in Figure 1.1. In addition, you can create your own tabs or groups; see "Working With Script Groups" on page 4 for details.

Figure 1.1



Within any of the tabs, use the **Edit Script** button to open the **Script Editor**; see "Editing a Script" on page 8 for details. Select the **Open Dashboard** button or double-click a script to open the Script Dashboard; see "Script Dashboard" on page 38 for details.

The **All Scripts** tab contains the complete list of scripts, and it is the only group from which new scripts can be created or scripts can be deleted. When on the All Scripts tab, use the **New Script** button to create a script. Use the options in the **Edit** menu to delete an existing script.

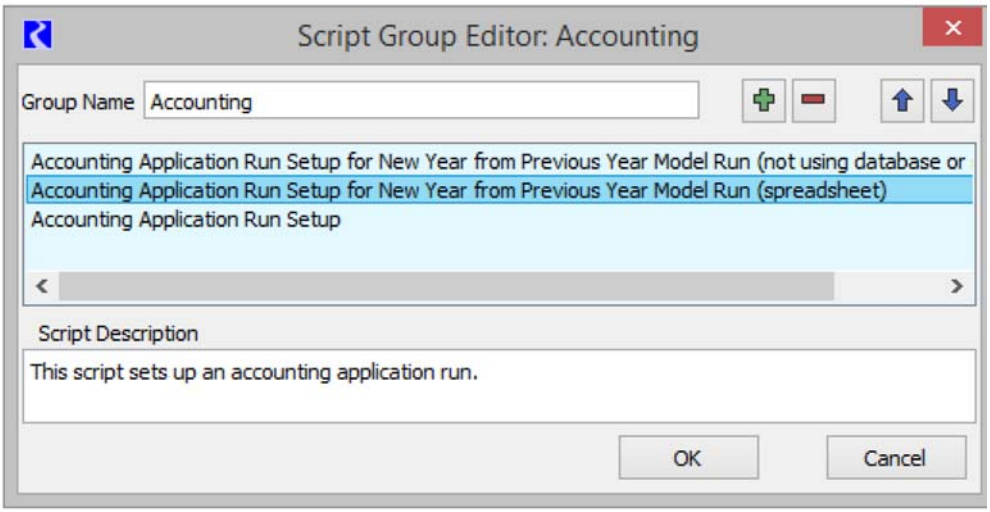
The lower portion of the Script Manager shows the selected script's description. This allows you to quickly see the description to know what the script does or why you would want to use it. The descriptions are edited as part of the script; see "Using the Script Editor" on page 10 for details.

The Script Manager can be configured to automatically open upon loading a model by checking the **Open Script Manager on model load** toggle. You can specify which script group tab is initially displayed by selecting a script from the **Default** tab menu. The **Default** tab is shown when the Script Manager window is initially opened (either upon model load or manually) within a RiverWare session.

Working With Script Groups

To create a user-defined script group, use the Script Manager's **Edit**, then **New Group** menu. To modify an existing user-defined group, select the tab and then use the **Edit**, then **Edit Group** menu item. Both of these menu items open the Script Group Editor, shown in Figure 1.2.

Figure 1.2

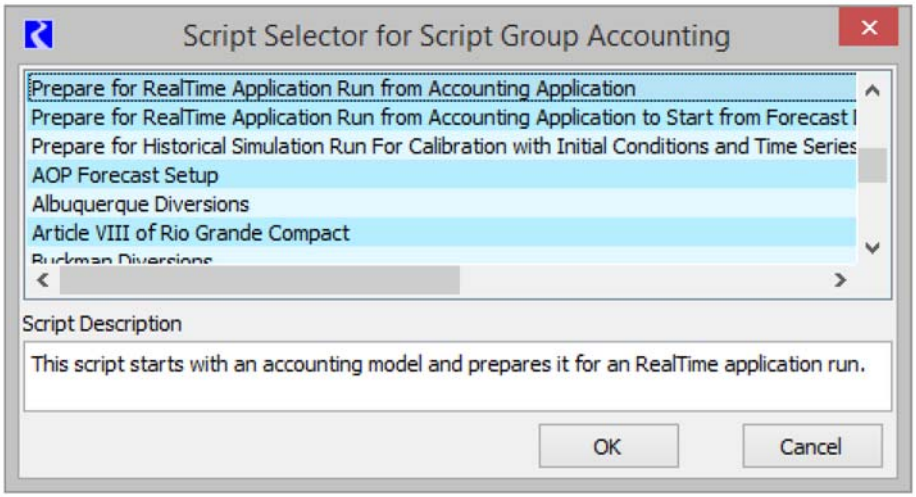


The Script Group Editor lists the scripts within the group and you can add, remove and reorder the scripts within the group.

To add a script to the group, select the green **Plus** button. This opens the Script Selector, which lists those scripts not already included in the script group. Select one or more scripts and select **OK** to add the scripts to the group.

Within the Script Manager, you can rearrange the tab groups by dragging a tab to a new position.

> **Note:** A script group does not *own* the scripts listed as within the group. Instead, the group is simply a set of references to existing scripts. Thus, a script can be shared by multiple groups, and the effort of developing a new script need not be duplicated in order to immediately include and use a script in other groups.

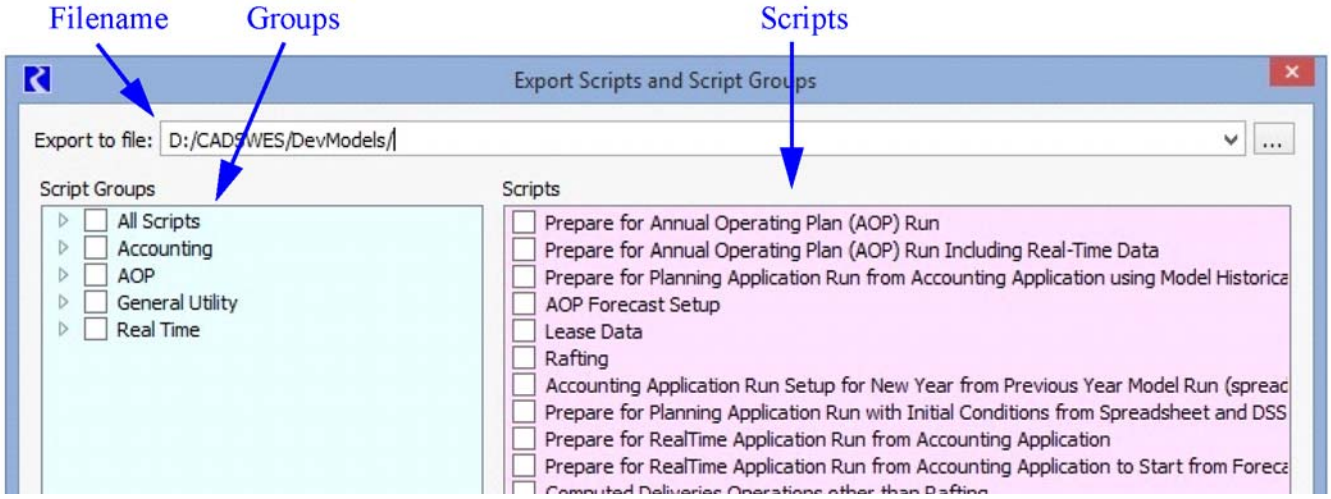


Importing and Exporting Scripts

One or more Scripts or Script Groups can be exported from the model and then imported into another model. This section describes this process.

Exporting Scripts

From the Script Manager, use the **File**, then **Export Selected Script** to export one or more scripts or groups to a file. This opens the Export Scripts and Script Groups.



First, specify a file for export. Either select from the menu, type in a file name or select the **More** button to choose a file on the system.

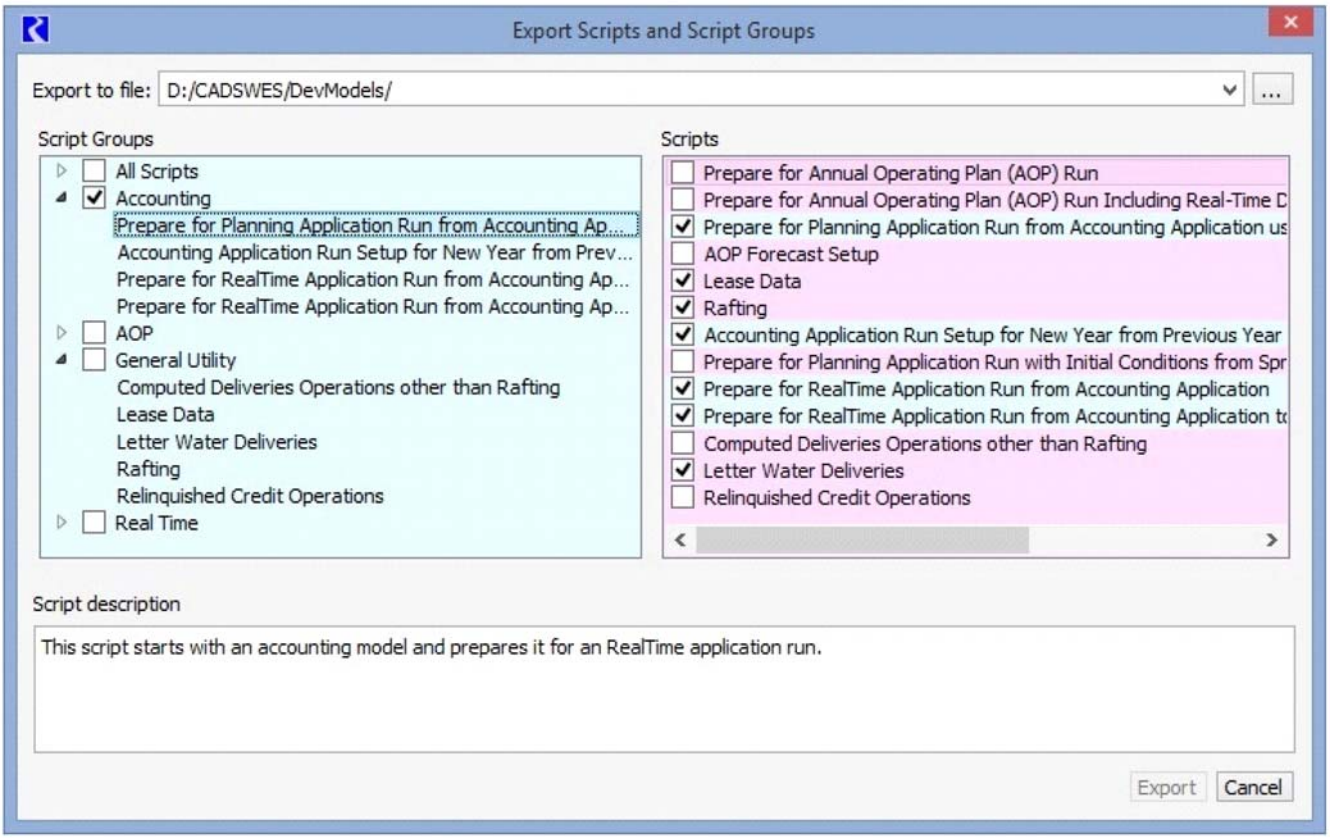
Then specify the scripts or groups to export.

The dialog contains two sections, one on the left, listing Script Groups and one on the right, listing all Scripts. The lower panel shows the description of the highlighted (not checked) script.

To export one or more Script Groups and the scripts referenced by the group, select the checkboxes in the left panel. All constituent scripts will be exported. Use the tree view arrows to see which scripts are part of the group. Also, as you select a group, the referenced scripts will also be checked and the background color will change in the Scripts panel on the right, as shown in Figure 1.3.

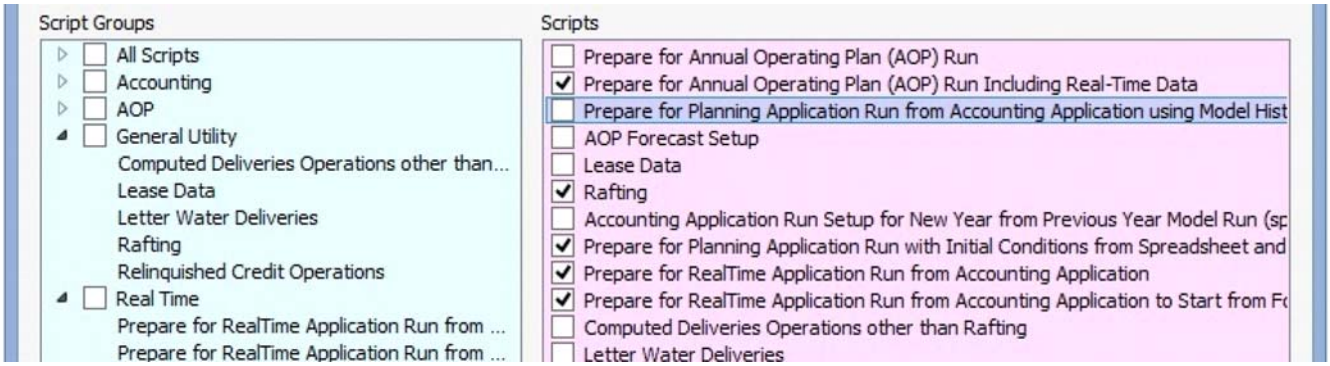
> **Note:** You cannot deselect scripts that have been selected for export that are part of a group. The following screenshot shows that one group containing four scripts (blue) will be exported as well as three additional scripts (pink).
>
> **Note:** To export all of the scripts in the model, simply select the **All Scripts** checkbox.

Figure 1.3



To select one or more scripts to export that are not part of a group, select them from the right panel. In Figure 1.4, five scripts will be exported.

Figure 1.4

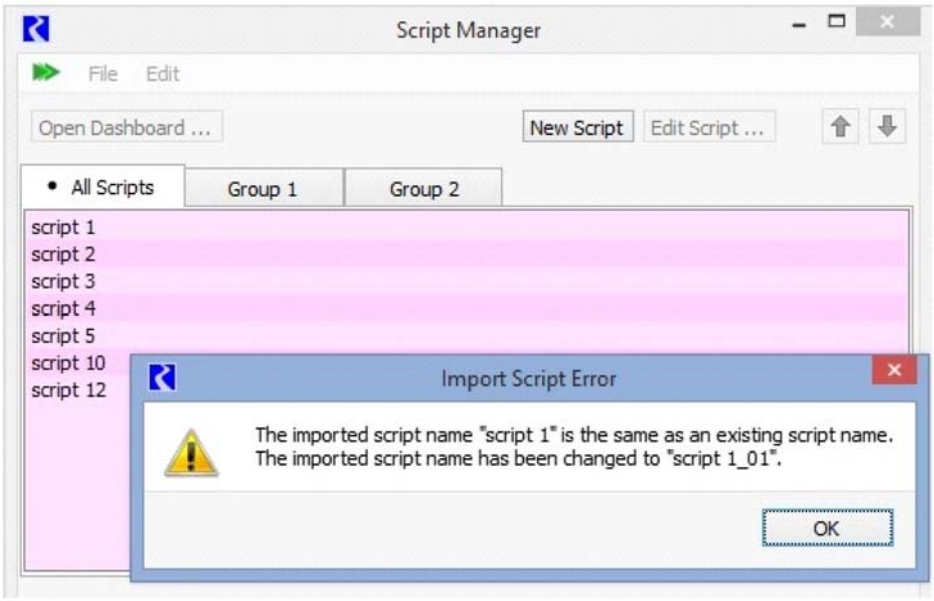


Once you have checked the desired groups/scripts, make sure to specify a file name at the top of the dialog. Then select **Export** to export the file.

Importing Scripts

Use the **File**, then I**mport Script** to import previously exported scripts. Choose the file name and select **Open**.

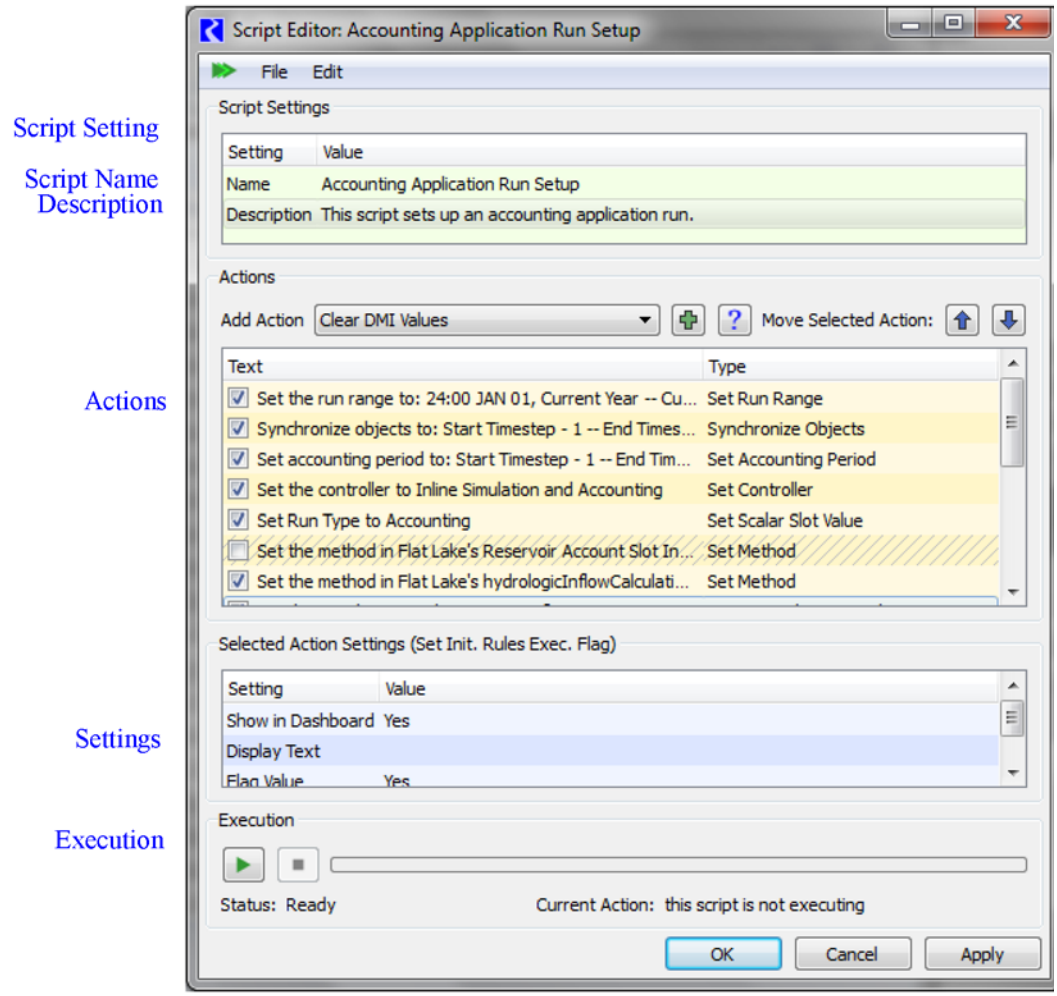
If there is a name duplication between a Script or Script Group being imported and an already existing Script or Script Group, the name being imported is automatically changed by appending "_nn" to the existing name, where nn is a two digit integer that is increased from 01 until a unique name is encountered. This scheme preserves imported-item name information without requiring detailed renaming actions during the import operation. You are notified that automatic renaming has occurred, in case you want to edit the names of the affected items afterwards.



Editing a Script

The Script Editor is where you define the Actions that the script performs. In addition, you define how the script will appear in the Script Dashboard and what values can be edited by the script user. To open the Script Editor, select a script in the Script Manager and select **Edit**. Figure 1.5 identifies the sections of the Script Editor.

Figure 1.5



Script Settings

This set of items applies to the entire script and includes the following configuration:

- Script Name. The unique name of the script appears here and in the list of scripts in the Script Manager.
- Description Editor. Enter a description, if desired, that will appear in the Script Manager.

Actions panel

This is an ordered sequence of actions, each of which has one of the defined action types. Each item has either a default or user defined label and its type. E.g., Text: “Set the run range to:” Type: “Set Run Range”. Each action can be disabled (or enabled) using the checkbox.

Selected Action Settings panel

Each action type has two or more associated parameters, each of which has an associated setting. For example, an action of type Set Method has two standard parameters, Show in Manager and Display Text, and three specific parameters, Object Name, Category, and Method.

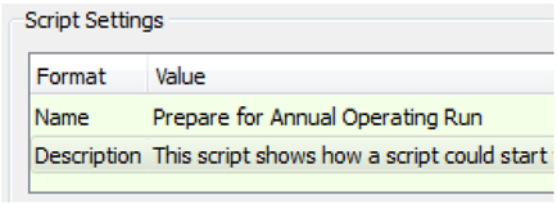
Execution panel

When creating and debugging your script, it may be more convenient to run the script from here than opening the Script Dashboard. Use the **Start** button to execute the script. The ability to run the script from the Editor is strictly for convenience while creating and editing the script. The interface and feedback for running a script is much better on the Script Dashboard; see "Script Dashboard" on page 38 for details.

Using the Script Editor

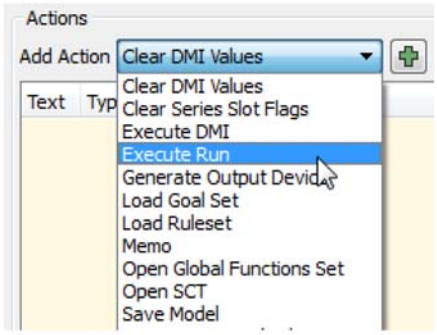
This section describes how to use the Script Editor to define the script. This assumes you have created a new Script from the Script Manager. The Script Editor opens automatically for new scripts or select **Edit** to open an existing script.

1. Enter a meaningful name in the Name field.
2. Enter a description if desired.



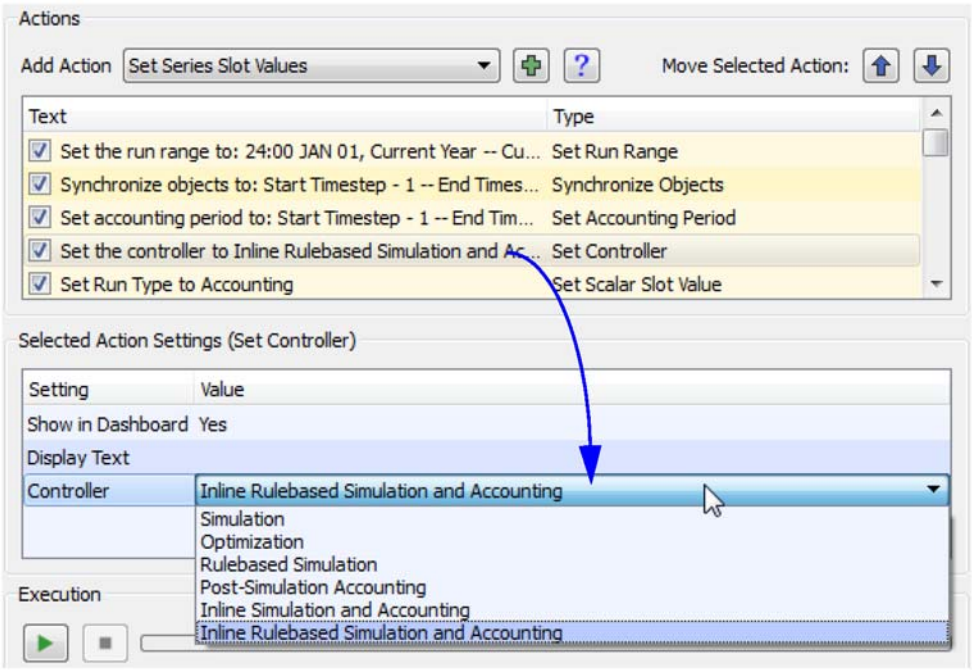
3. Add an Action to the script; see "About Action Types" on page 11 for a complete list of the available actions.

   Select **Help**  for a quick description of each action. Use the **Add Action** menu and select the desired action. When you select the action it will automatically add it to the script and select it.



4. Configure that action. With the action selected, the Settings panel shows the available parameters. For example, in Figure 1.6, the Set Controller action is selected, so now the controller can be chosen from the list in the settings panel.

Figure 1.6



Each action has two or more settings that can be configured. For many of the settings, you provide a numeric value but can also specify that it is editable in the Script Manager. For example, you may use the Set Scalar Slot Value action to set a model trigger. You can specify that the value should default to 0 in the script but allow the user to override the default and use a slider bar in the Script Manager to change the value. See "Configuring User Control on the Script Dashboard" on page 19 for details on this type of configuration.

5. Repeat Step 3. and Step 4. until you have fully defined the script.

6. Use the arrows 

to rearrange the actions as necessary. They will execute in order from top to bottom.

7. Once you have the script defined. consider what text to be show on the Script Manager. Use the **Show in Manager** and **Display Text**, to modify whether the setting is shown and if so, the text that is displayed, respectively. See "Common Settings for Action Types" on page 14 for details. In addition, you can disable actions using the check boxes. This is enabled/disabled setting is shown on the dashboard but can be re-enabled there too. See "Common Settings for Action Types" on page 14 for details.

8. Select **OK** to apply the changes and close the dialog.

About Action Types

Table 1.1 describes the types of actions. Within the Script Editor, select **Help** for a quick reference of the actions. Each action type also has a link to additional information on its settings. See "Action Type Settings Reference" on page 23 for details about the settings.

Table 1.1 Descriptions of the available actions

Action Type	Description	Link
Clear DMI Values	Clear the values set by a DMI (by using the DMI invocation manager functionality)	Clear DMI Values, page 23
Clear Scalar Slot Values	Clear the values on a set of Scalar Slots (i.e. set them to NaN).	Clear Scalar Slot Values, page 23
Clear Table Slot Values	Clear certain values on a set of Table slots (i.e. set them to Nan).	Clear Table Slot Values, page 23
Configure MRM Index Sequential	For the specified MRM configuration, modify the configuration in terms of Index Sequential.	Configure MRM Index Sequential, page 23
Configure MRM Input DMI	For the specified MRM configuration, modify the configuration in terms of Input DMIs, particularly the repeat count.	Configure MRM Input DMI, page 24
Configure MRM Output	For the specified MRM configuration, modify the configuration in terms of the Output.	Configure MRM Output, page 24
Create Slot Cache	Create a slot cache of workspace series slot values.	Create Slot Cache, page 24
Create Snapshot	Create a Snapshot of the specified slots.	Create Snapshot, page 25
Divider	Create a different colored row in the Script Editor and Script Dashboard to visually separate actions.	Divider, page 25
Enable Dispatching	Enable or disable dispatching of the specified objects.	Enable Dispatching, page 25
Enable MRM Distributed Runs	For the specified MRM configuration, specify whether to distribute concurrent runs to multiple processors on the same machine.	Enable MRM Distributed Runs, page 25
Enable RPL Item	Enable or disable a RPL item (policy group, rule, method, or goal)	Enable RPL Item, page 26
Evaluate Expression Slots	Evaluate a set of expression slots.	Evaluate Expression Slots, page 26
Execute DMI	Invoke a DMI or DMI group.	Execute DMI, page 26
Execute MRM Run	Execute the specified MRM configuration.	Execute MRM Run, page 26
Execute Run	Run the model.	Execute Run, page 26
Execute Script	Execute a script from within a script.	Execute Script, page 26
Generate Output Device	Generate the specified Output Device. This could be a file based device like a Model Report or a graphical device like a Plot Page.	Generate Output Device, page 27
Global Time Scroll	Scroll time and series displays to the specified symbolic date/time.	Global Time Scroll, page 27
Load Goal Set	Open and load a goal set (for Optimization runs).	Load Goal Set, page 27
Load Ruleset	Open and load a ruleset (for Rulebased Simulation runs)	Load Ruleset, page 27
Memo	Pause script execution and optionally open a confirmation dialog with your user specified message. While script execution is paused, you can interact with RiverWare or other applications in ways not directly supported by scripts. Select the script **Start** button to resume the script execution.	Memo, page 28

Table 1.1 (Continued) Descriptions of the available actions

Action Type	Description	Link
Open File	Open the specified external file, for example an Excel file or Word document.	Open File, page 28
Open Global Functions Set	Open a global function set.	Open Global Functions Set, page 28
Open Object	Open the specified objects.	Open Objects, page 28
Open SCT	Open an SCT with the specified filename.	Open SCT, page 29
Open Slots	Open the specified slots.	Open Slots, page 29
Remove RPL Set	Remove a selected RPL set from the model. An RBS Ruleset or Optimization Goal set is unloaded and closed, a Global Function Set is closed, and the Initialization Rules Set is cleared.	Remove RPL Set, page 29
Reorder RPL Set	Reorder the items (rules, goals, statements) in a RPL set according to priority or index defined on the specified table slot.	Reorder RPL Set, page 29
Replace Initialization Rules Set from File	Replace the existing Initialization Rules Set from a file.	Replace Initialization Rules Set From File, page 29
Save Model	Save the model to a file.	Save Model, page 30
Set Account Method	Set the selected method in one of an account's method categories.	Set Account Method, page 30
Set Accounting Period	Set the time range of the accounting system (i.e., the Begin Accounting Period and End Accounting Period dates).	Set Accounting Period, page 30
Set Controller	Set the current controller.	Set Controller, page 30
Set Dataset File	Set the DSS file for a DSS dataset, or the Excel workbook for an Excel dataset.	Set Dataset File, page 31
Set Excel Dataset Run Name	For the specified Database DMI Excel Dataset, set the run name type and/or specify a new single run name.	Set Excel Dataset Run Name, page 31
Set Init. Rules Exec. Flag	Set the "Execute Initialization Rules" run control flag. This flag controls whether or not the initialization rules are executed as part of each run.	Set Init. Rules Exec. Flag, page 31
Set Method	Set the selected method in the specified category for a set of objects.	Set Method, page 31
Set MRM Descriptor	For the specified MRM configuration, set the specified keyword, value pair MRM Descriptor.	Set MRM Descriptor, page 32
Set MRM Ruleset	In a given MRM configuration, set a new file path to a ruleset.	Set MRM Ruleset, page 32
Set MRM Run Range	Set the time range of the run (i.e. the Run Start and End dates) for the specified MRM configuration.	Set MRM Run Range, page 32
Set Run Range	Set the time range of the run (i.e. the Run Start and End dates).	Set Run Range, page 33
Set Run Timestep	Set the timestep size of the run (E.g 1 Hour, 1 Day, 1 Month, 1 Year). Options are included to specify how to synchronize and optionally aggregate data.	Set Run Timestep, page 33
Set Scalar Slot Value	Set one or more Scalar Slots values.	Set Scalar Slot Value, page 34

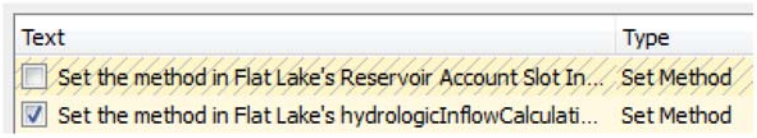
Table 1.1 (Continued) Descriptions of the available actions

Action Type	Description	Link
Set Series Slot Flags	Set the specified flag (input or output) on a set of series slots within the specified time range.	Set Series Slot Flags, page 35
Set Series Slot Values	Set Series Slot's values within a time range.	Set Series Slot Values, page 35
Set Table Slot Value	Set a single Table Slot's value at the specified row and column.	Set Table Slot Value, page 36
Synchronize Objects	Set the time range for the specified objects.	Synchronize Objects, page 37
Synchronize Slots	Set the time range for the specified slots.	Synchronize Slots, page 37

Common Settings for Action Types

An action is controlled by user settings. Actions of all types have the following settings.

- **Show In Dashboard.** This is a Yes/No value that controls whether the action is listed in the Script Dashboard.
- **Display Text.** This brief text identifies the action. This text is, for example, shown in the dashboard. If the field is empty, default text will be automatically generated based on the action type.
- **Enabled/Disabled.** Each action has a check box next to it which determines whether it is enabled or disabled in the script editor. This setting is also used to indicate whether the action starts as enabled or disabled in the script dashboard, although it can be overridden in the dashboard.



Setting Information

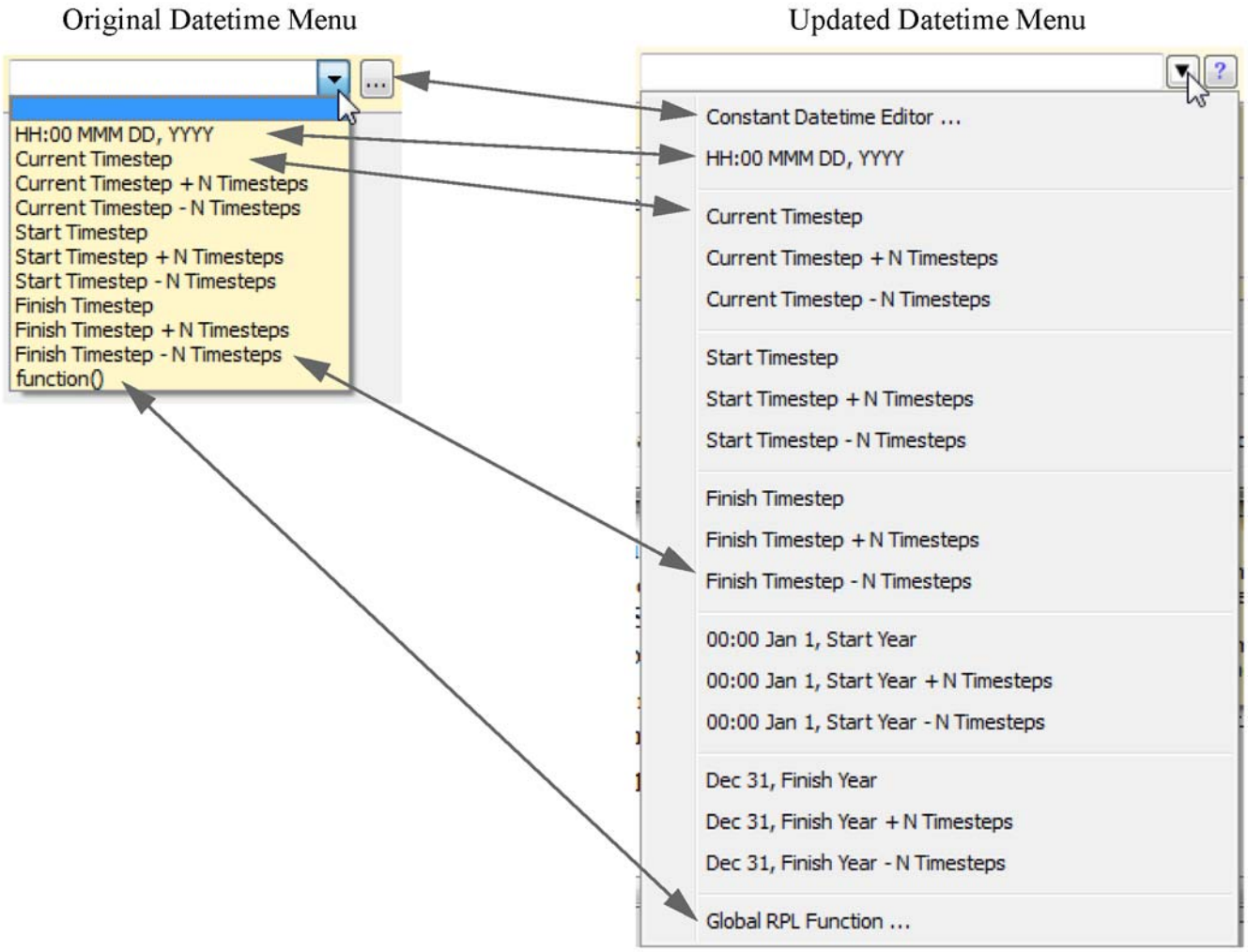
The following sections provide more information on certain settings or on settings that are shared by many actions that have additional controls.

Datetime Information

Datetimes in a script can be specified either as absolute values or a symbolic values. There are two slightly different interfaces, but both have essentially the same functionality. Depending on the action and whether it is the editor or dashboard, one of the following interfaces will be used.

> **Note:** The current menu is being transitioned to the updated menu. Common functionality has been highlighted in Figure 1.7.

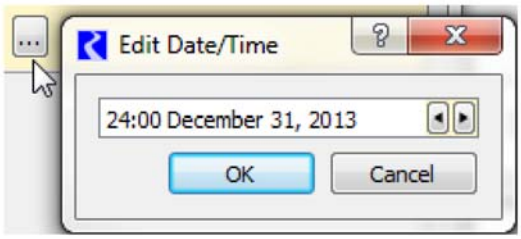
Figure 1.7



Following is additional information on each of these ways to specify the datetime.

- **Choose a Datetime**. Use the datetime spinner by selecting the **More** button or the **Constant Datetime Editor** option. This opens the Date/Time selector.

  Note: This uses the model timestep size.



- **Type a Datetime**. Enter a date by typing. Use the HH:00 MMM DD, YYYY as a guide: Hour (H), Month (M), Day (D), Year (Y).

- **Specify the Datetime Symbolically**: Use one of the options shown and specify any of the pieces necessary like integers “N”, Hour (H), Month (M), Day (D), Year (Y). On the updated menu, select **Help** for details on the symbolical datetimes. Basically, any fully specified datetime can be used. See “DATETIME” in *RiverWare Policy Language (RPL)* for details.

  Note: No @ or quotes “ ” are necessary when specifying the datetimes. In this context, “Current Timestep” means the machine/operating system time, rounded up to a controller timestep. So if you are running the script at noon on the 4th of July, 2014 in a model with a 1 Day timestep, “Current Timestep” would return 24:00 July 4th, 2014.

- **Use a Global Function**: To allow additional flexibility specifying datetimes, you can specify a RPL function from an opened Global Function Set; see “Global RPL Functions” in *RiverWare Policy Language (RPL)* for details. You can type in the name of the function (original menu) of the form function() (see note below) or choose it using a selector (updated menu). This function must return a fully specified DATETIME variable

  Note: Datetimes stored on slots cannot be set to NaN as an override on the dashboard, but can be configured to be set to NaN from the Script Editor.

  Note: Originally, you could set a datetime to a RPL function using the function() notation on the Script Editor. Functions specified this way are still supported. But for new scripts or any edits, functions should be specified by changing the “Set by” setting to Function and choosing the function from the selector.

Filename Information

Filenames can be entered manually or chosen using the file chooser. You can specify files using environment variables using the $VARIABLE syntax.

RPL Reorder Set Action

This action allows you to reorder items (rules, goals, groups, or even statements within the specified set. The new order is specified on a table slot. Following is an example that shows the use of this action. Figure 1.8 shows the ruleset before the script is executed.

Note: Statements are shown using the **View** menu.

Figure 1.8

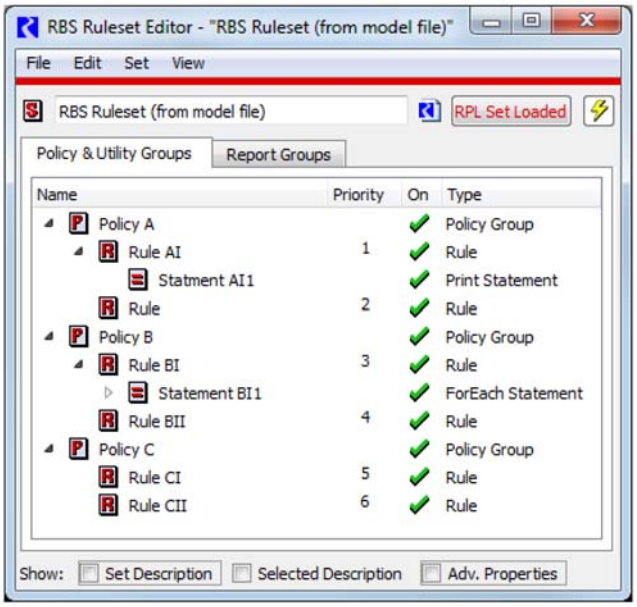


Figure 1.9 shows the specified table that describes the desired ordering. The row labels must match the names of the items in the set.

Figure 1.9

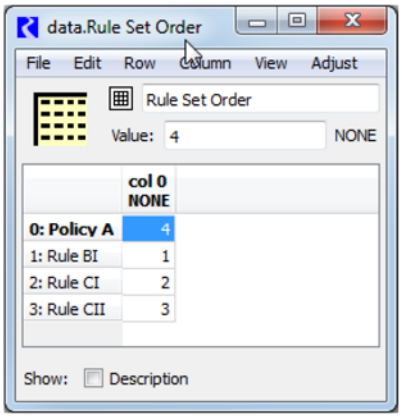


Figure 1.10 shows a text description of this reordering.

Figure 1.10

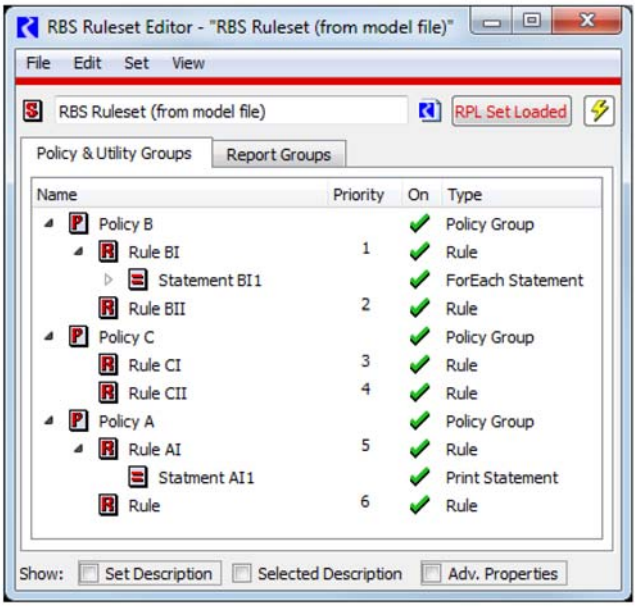
```
Before
  1. Policy A: desired order =  4, index in set = 2
  2. Rule BI: desired order =  1, index in set = 3
  3. Rule CI: desired order =  2, index in set = 5
  4. Rule CII: desired order =  3, index in set = 6

Reordering:
  Rule BI: leaving in place.
  Rule CI: leaving in place.
  Rule CII: leaving in place.
  Policy A: moving to after Rule CII

After
  1. Rule BI: desired order =  1, index in set = 1
  2. Rule CI: desired order =  2, index in set = 3
  3. Rule CII: desired order =  3, index in set = 4
  4. Policy A: desired order =  4, index in set = 6
```

Figure 1.11 shows the ruleset after reordering.

Figure 1.11



Setting Slot Values

This section contains important information on the following actions that set slot values in the model:

- "Set Scalar Slot Value" on page 34
- "Set Series Slot Values" on page 35
- "Set Table Slot Value" on page 36

"Configuring User Control on the Script Dashboard"describes how to configure user controls on the dashboard. Then, see "Setting Slot Values using a Global Function" on page 21 for more information on setting numeric values using a global function.

Configuring User Control on the Script Dashboard

This section applies when the "Set by" setting is using "Values" For actions that set values in the model (Set Scalar Slot Value, Set Series Slot Values, Set Table Slot Value), you can either specify the value to set or give a default, but allow the user of the script to override that value. This section describes configuring the action so that the user can modify the value using either a value editor, radio buttons or a slider. These are controlled by the settings shown in Figure 1.12.

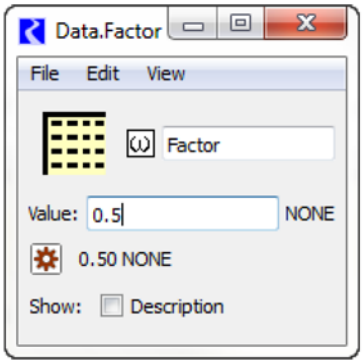
Figure 1.12



The following sections describe the three "Yes" cases. In each case, an example will be shown where a script is setting a scalar slot called Data.Factor to a value between 0.1 and 1.5. Perhaps this factor is then referenced by a rule to compute demands or inflows. For reference, Figure 1.13 shows this scalar slot.

For all of the Yes selections for Allow Editing, you can also choose to Show Current Value. When Yes, the current value (for a single scalar slot only) is shown on the Dashboard below the editing controls.

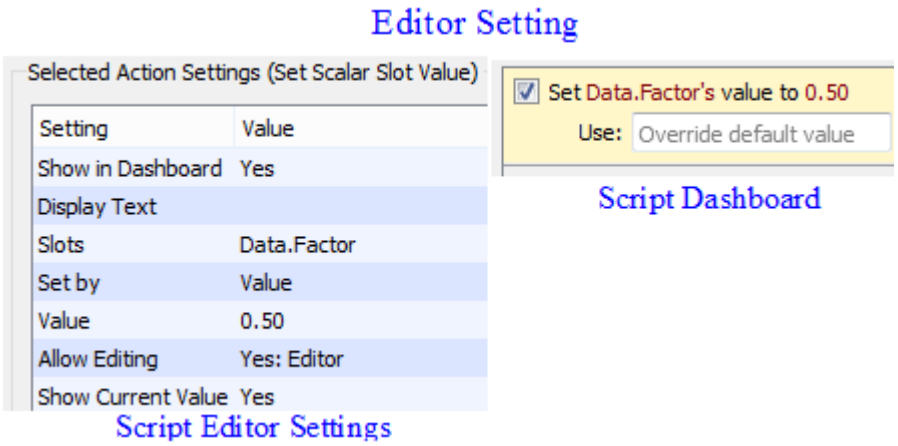
Figure 1.13



Editor

When the Allow Editing setting is Yes: Editor, an editable field is shown on the Script Manager. The default value will be used unless the user enters a different value in the field. Figure 1.14 shows the settings to get this control.

Figure 1.14

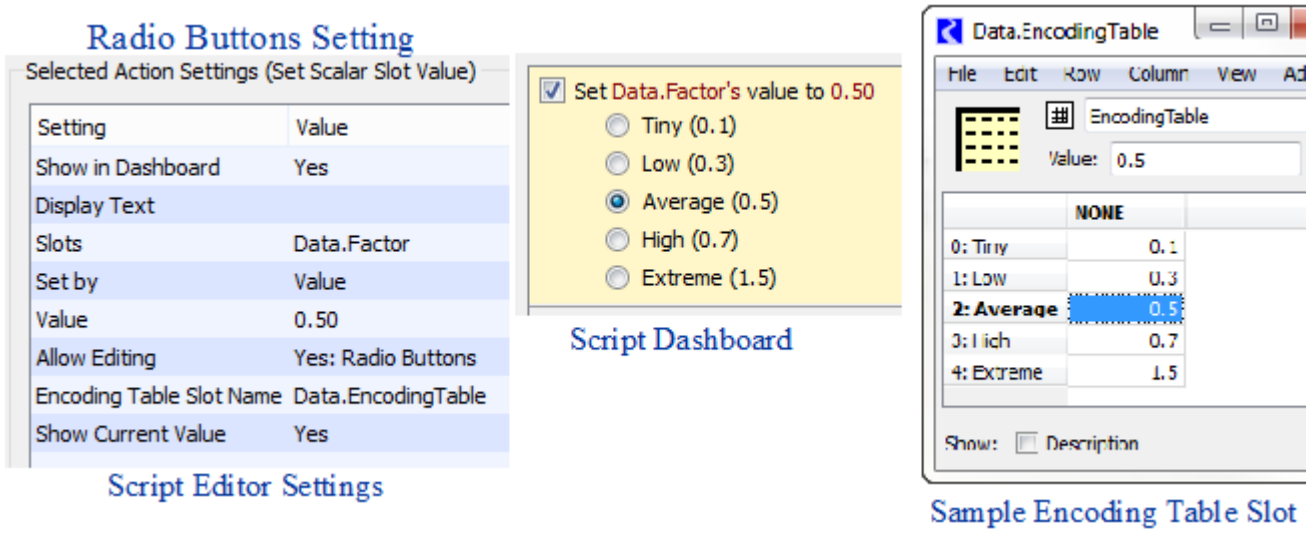


Radio Buttons

When the Allow Editing setting is Yes: Radio Buttons and an Encoding Table Slot is specified, radio buttons are shown on the Script Manager. The default value will be used unless the user selects one of the radio buttons to use that value.

The Encoding Table Slot Name setting specifies a Table Slot whose row labels and values establish a textual encoding of values. Figure 1.15 shows a sample encoding table slot. The row labels are used as the labels on the radio buttons. The values in the table are displayed on the Script Manager and will be used by the action for that button.

Figure 1.15

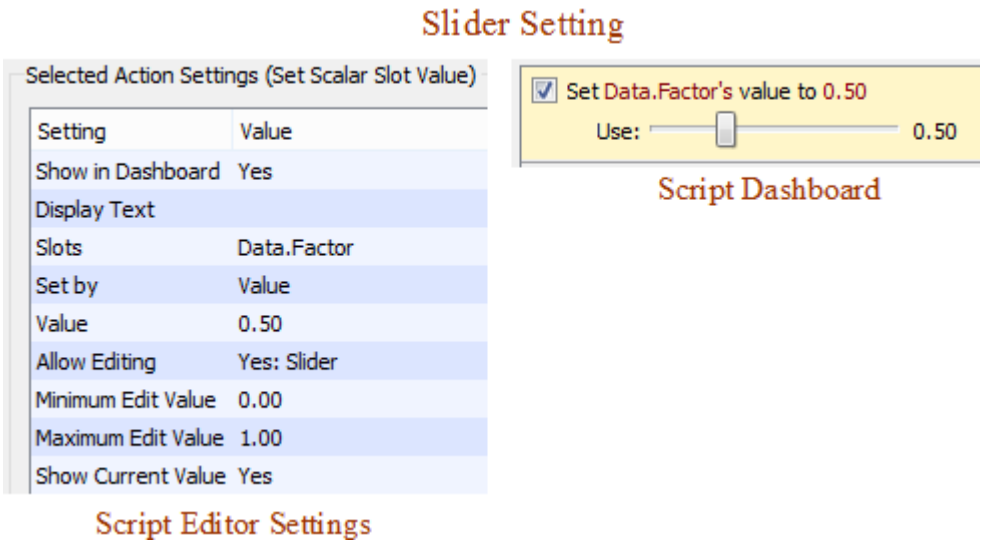


Sliders

When the Allow Editing setting is Yes: Slider and Minimum and Maximum Edit Values are defined, a slider will be shown on the Script Manager. The default value will be used unless the user moves the slider to a different value.

Figure 1.16 shows the settings to configure this control. The available slider positions are computed by dividing the interval between the minimum and maximum by 100.

Figure 1.16



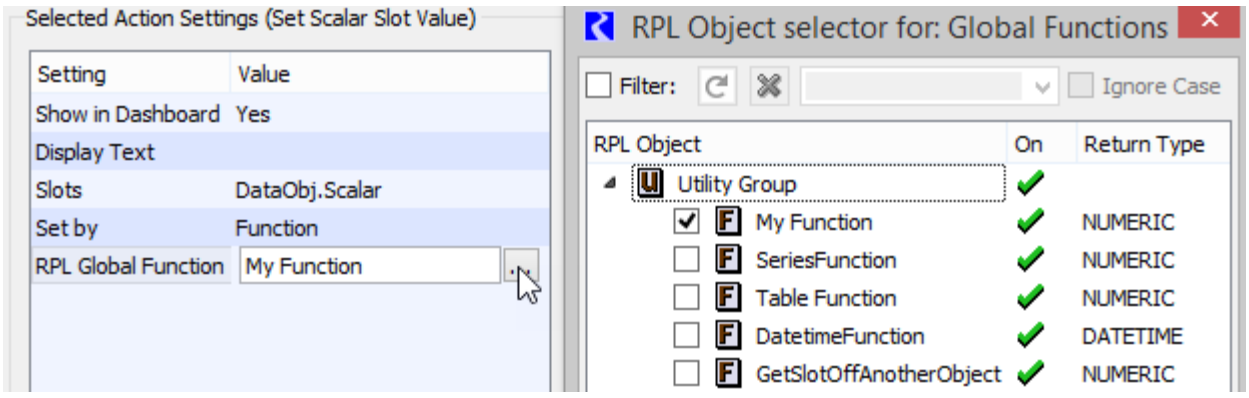
Setting Slot Values using a Global Function

The three script actions allow slot values to be set to values as described above. They also allow the slot value to be set to numeric or datetime values using a RPL Global Function.

- "Set Scalar Slot Value" on page 34
- "Set Series Slot Values" on page 35
- "Set Table Slot Value" on page 36

To use this functionality, in any of these three actions, in the "Set by" setting, choose "Function". Then choose the desired RPL Global Functions Set and the specific RPL Global Function to use as shown in Figure 1.17.

Figure 1.17 Screenshot of selecting a Global Function



Note: The Global Function must return a NUMERIC or DATETIME value. Setting Text Series Slots is not supported by this mechanism.

The global function can contain various combinations of arguments based on the action. You can use zero, one or more arguments. The arguments passed for each action are:

- "Set Scalar Slot Value": SLOT slotToSet. This action loops over each slot and calls the function passing in the slot as the argument.
- "Set Series Slot Values": SLOT slotToSet, DATETIME date: This action loops over each slot and datetime and calls the function passing in the slot and datetime as the arguments.
- "Set Table Slot Value": SLOT slotToSet, NUMERIC row, NUMERIC col. This action loops over each slot and calls the function passing in the slot, row, and column as the arguments.This action only sets the slot at the specified row and column. It does not loop over the rows and columns.

Example 1.1 Using a RPL Global Function to set multiple series slots.

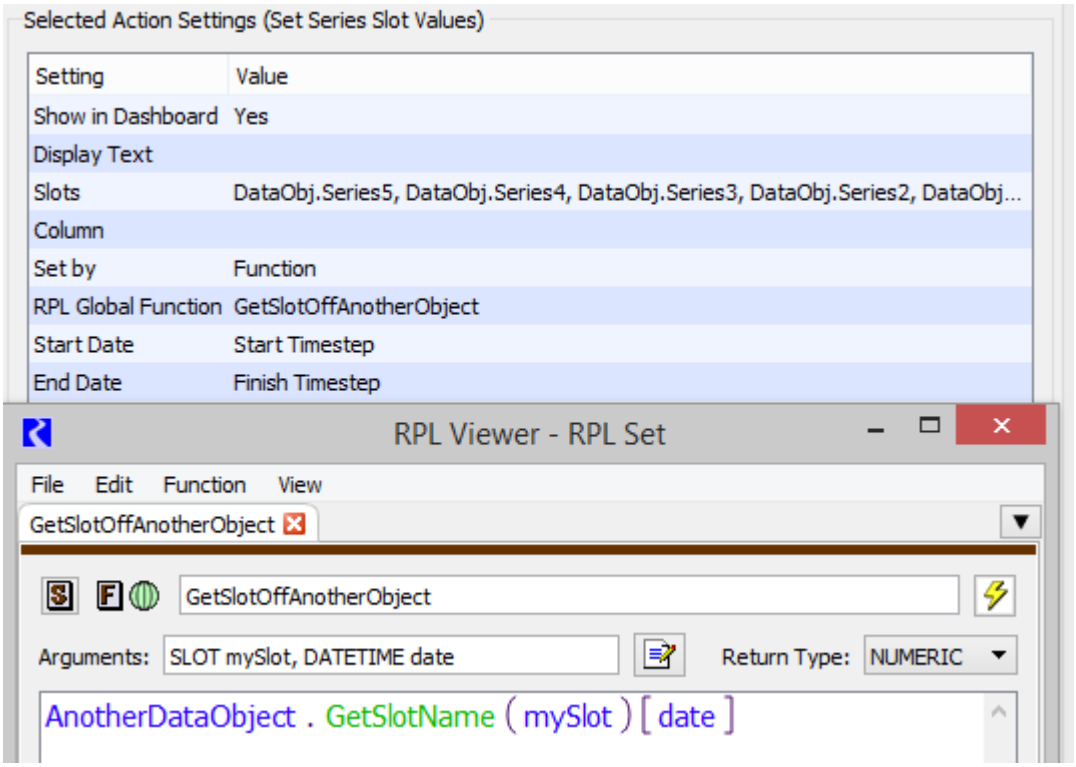
This is best explained with an example. Let's assume we want to set five series slots to a value on a data object at each timestep in the run.

Set up the script as follows:

1. In the Set Series Slot Values action, "Slots" setting, choose the 5 slots to set.
2. Choose the Global Functions Set and the desired Global Function
3. Specify the Start and End Date

In our example, the settings and RPL function is shown in Figure 1.18.

Figure 1.18 Screenshot of a Set Series Slot Values that calls a numeric RPL function



When this action is executed, it will loop over the 5 slots and all timesteps in the run. For each slot/timestep, it will call the GetSlotOffAnotherObject function and pass the slot and datetime into the function. The function computes a numeric, in this case looking up the value on another object. The action then sets that value.

Action Type Settings Reference

This section provides details about settings specific to each type of action.

• Clear DMI Values

Clear the values set by a DMI (by using the DMI invocation manager functionality).

Setting	Type	Notes
DMI Name	Text	Enter the DMI name or choose the DMI using the menu.

• Clear Scalar Slot Values

Clear the values on a set of Scalar Slots (i.e. set them to NaN).

Setting	Type	Notes
Slots	Slot Selection	Enter or choose one or more scalar slots for which you wish to clear the values.

• Clear Table Slot Values

Clear certain values on a set of Table slots (i.e. set them to Nan).

Setting	Type	Notes
Slots	Slot Selection	Enter or choose one or more table slots. The values in these slots will be cleared.
Row	Text	Specify the row as either the row number (zero based) or the row heading
Column	Text	Specify the column as either the column number (zero based) or column heading

• Configure MRM Index Sequential

For the specified MRM configuration, modify the configuration in terms of Index Sequential. This is shown on the Input tab in the MRM configuration. See “Index-sequential Runs” in *Solution Approaches* for details.

Setting	Type	Notes
MRM Configuration	Text	Enter or choose the name of an MRM configuration defined in the MRM Run Controller
Number of Runs	Integer	Specify the Number of Runs
Initial Offset	Integer	Specify the Initial Offset
Interval	Integer	Specify the Interval

Configure MRM Input DMI

For the specified MRM configuration, modify the configuration in terms of Input DMIs, particularly the repeat count. This is shown on the Input tab in the MRM configuration. See "Input" in *Solution Approaches* for details.

Setting	Type	Notes
MRM Configuration	Text	Enter or choose the name of an MRM configuration defined in the MRM Run Controller.
DMI Name	Text	Choose the Input DMI to configure. This should already be selected in the MRM configuration.
Repeat Count	Integer	Specify the Repeat Count.

Configure MRM Output

For the specified MRM configuration, modify the configuration in terms of the Output. This is shown on the Output tab in the MRM configuration. See "Output" in *Solution Approaches* for details.

Setting	Type	Notes
MRM Configuration	Text	Enter or choose the name of an MRM configuration defined in the MRM Run Controller.
Control File	File Chooser	Enter the file path. See "Filename Information" on page 16 for details.
Generate CSV Files	Yes/No	Specify whether or not to generate CSV output
Generate NetCDF Files	Yes/No	Specify whether or not to generate NetCDF output
Allow Control File Editing	Yes/No	Specify if you would like the control file to be editable on the dashboard.

Create Slot Cache

Creates a slot cache that contains the current values of all series slots on the workspace, in the range of the current controller. See "Slot Cache" in *User Interface* for details on the cache.

> **Note:** The slot cache is under development. Contact riverware-support@colorado.edu for details and the current status of this feature.

Setting	Type	Notes
None		

Create Snapshot

Create a Snapshot of the specified slots.

Setting	Type	Notes
Slots	Slot Selection	Enter or choose one or more slots for which you wish to take a snapshot.
Snapshot Name	Text	Specify the name of the snapshot object to create.
Replace Existing Snapshot?	Yes/No	Choose to replace a snapshot of the same name. If No, then a unique name for the snapshot will be created.

Divider

This item adds a divider to the Script Editor and Script Dashboard. It does not do anything in script execution; it is strictly for to provided a visual separation between actions. The Display Text can be used to add text to the divider.

Setting	Type	Notes
Divider Color	Color Chooser	Use the color chooser to select the background color for the divider.

Enable Dispatching

For the specified objects, enable or disable the dispatch method execution. When disabled, the object icons on the workspace and in other dialogs appear dimmed.

Setting	Type	Notes
Objects	Object Selection	Choose the desired objects using the object selector.
Enable or Disable?	Enable Dispatching / Disable Dispatching	Specify **Enable** to enable dispatching. Specify **Disable** to disable dispatching of those objects.

Enable MRM Distributed Runs

For the specified MRM configuration, specify whether to distribute concurrent runs to multiple processors on the same machine. See "Distributed Concurrent Runs" in *Solution Approaches* for details on distributed MRM.

Setting	Type	Notes
MRM Configuration	Text	Enter or choose the name of an MRM configuration defined in the MRM Run Controller.
Enable or Disable?	Enable / Disable	Specify **Enable** to run in distributed mode. Specify **Disable** to run the concurrent runs on a single processor.

Enable RPL Item

Enable or disable a RPL item (policy group, rule, method, or goal).

Setting	Type	Notes
Set	Single Selection	Choose the set to which the RPL item belongs.
Item	Single Selection	Use the RPL item chooser to select the item.
Enable or Disable?	Yes/No	Specify Yes to Enable or No to disable the specified item.

Evaluate Expression Slots

Evaluate a set of expression slots.

Setting	Type	Notes
Slots	Slot Selection	Choose the slots using the selector.

Execute DMI

Invoke a DMI or DMI group.

Setting	Type	Notes
DMI Name	Text	Enter the DMI name or choose the DMI using the menu.

Execute MRM Run

Execute the specified MRM configuration.

Setting	Type	Notes
Configuration	Text	Enter the name of an MRM configuration you wish to run.

Execute Run

Run the model.

Setting	Type	Notes
NA		

Execute Script

Execute a script from within this script.

> **Note:** The script called by this action cannot have additional Execute Script actions. Rephrased, a script cannot call a script that calls another script. Only one level of a script executing a script is allowed. Also, a script cannot call itself.

Setting	Type	Notes
Script Name	Text	Type the script name or choose the desired single script from the script selector.

Generate Output Device

Generate the specified Output Device. This could be a file based device like a Model Report or a graphical device like a Plot Page.

Setting	Type	Notes
Device Name	Text	Enter or choose the name of a output device (plot, file, report, chart)

Global Time Scroll

Scroll time and series displays to the specified symbolic date/time.

Setting	Type	Notes
Scroll to DateTime	Datetime	See “Datetime Information” on page 14 for details.
Allow Editing	Selection	Yes indicates that users should be allowed to override the start date in the Dashboard.

Load Goal Set

Open and load a goal set (for Optimization runs).

Setting	Type	Notes
File name	Text	Enter the file path. See “Filename Information” on page 16 for details.
Confirm Replacement	Yes/No	Yes indicates that if the specified set is already open, a dialog should be presented to confirm replacement of the existing set.
Show Dialog	Yes/No	When this setting is Yes, the RPL set dialog is shown when the action is executed. Otherwise, the set is loaded into the model, but not shown.

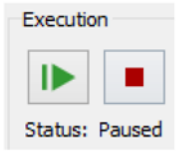
Load Ruleset

Open and load a ruleset (for Rulebased Simulation runs).

Setting	Type	Notes
File name	Text	Enter the file path. See “Filename Information” on page 16 for details.
Confirm Replacement	Yes/No	Yes indicates that if the specified set is already open, a dialog should be presented to confirm replacement of the existing set.
Show Dialog	Yes/No	When this setting is Yes, the RPL set dialog is shown when the action is executed. Otherwise, the set is loaded into the model, but not shown.

Memo

Pause script execution and optionally open a confirmation dialog with your user specified message. While script execution is paused, you can interact with RiverWare or other applications in ways not directly supported by scripts. Select the script **Continue** button to resume the script execution.

Setting	Type	Notes
Explanation	Multi-line text	Enter the desired text. This will optionally be displayed in the memo dialog and or the Dashboard dialog along with the Display Text.
Pause Execution	Yes/No	Yes indicates that script execution will pause when the memo is reached. Select the **Continue** button to resume the Script execution. 
Show Memo Dialog	Yes/No	Yes indicates that memo execution will open a dialog which presents the memo display and explanation texts.
Show Explanation in the Dashboard	Yes/No	Yes indicates that the explanation text will be displayed with the display text in the Dashboard.

Open File

Open the specified external file using the windows file association. If you would like to pause script execution after opening the file, add a Memo action, described at “Memo” on page 28, after the Open File action.

Setting	Type	Notes
File name	Text	Enter the file path. See “Filename Information” on page 16 for details. The file will be opened using the application associated with the file's extension in Windows. For example, temp.xlsx will be opened using Excel.

Open Global Functions Set

Open a global function set.

Setting	Type	Notes
File name	Text	Enter the file path. See “Filename Information” on page 16 for details.
Confirm Replacement	Yes/No	Yes indicates that if the specified set is already open, a dialog should be presented to confirm replacement of the existing set.
Show Dialog	Yes/No	When this setting is Yes, the RPL set dialog is shown when the action is executed. Otherwise, the set is opened into the model, but not shown.

Open Objects

Open the specified objects’ dialogs.

Setting	Type	Notes
Objects	Object Selection	Choose the objects to open.

Open SCT

Open a System Control Table (SCT) with the specified filename.

Setting	Type	Notes
File Name	Text	Enter the file path. See "Filename Information" on page 16 for details.

Open Slots

Open the specified slots.

Setting	Type	Notes
Slots	Slot Selection	Choose the slots using the selector.

Remove RPL Set

Remove a selected RPL set from the model. An RBS Ruleset or Optimization Goal set is unloaded and closed, a Global Function Set is closed, or the Initialization Rules Set is cleared.

> **Note:** If you have any of these sets saved in the model file and you remove them using this action, any changes will be lost! This action should only be used if you have the set saved to a file and wish to remove it from the model and reopen it from the file. Use with caution!

Setting	Type	Notes
Set	Single Selection	Choose the desired RPL set.
Confirm Removal	Yes/No	Yes indicates that a dialog should be presented to confirm removal of the set.

Reorder RPL Set

Reorder the items (rules, goals, statements) in a RPL set according to priority or index defined on the specified table slot.

Setting	Type	Notes
Set	Single Selection	Choose the desired RPL set.
Order Slot	Single Slot Selection	Choose the table slot which specifies the new RPL set ordering. See "RPL Reorder Set Action" on page 16 for details.

Replace Initialization Rules Set From File

Replace the existing Initialization Rules Set from a file. Any changes made to the Initialization Rules in the model will be lost. Use with caution!

Setting	Type	Notes
File name	Text	Enter the file path. See “Filename Information” on page 16 for details.
Confirm Replacement	Yes/No	Yes indicates that a dialog should be presented to confirm replacement of the existing Initialization Rules Set.
Show Dialog	Yes/No	When this setting is Yes, the RPL set dialog is shown when the action is executed. Otherwise, the set is replaced in the model, but not shown.

• Save Model

Save the model to a file.

Setting	Type	Notes
File Name	Text	Enter the file path. See “Filename Information” on page 16 for details.
Include Output Values	Yes/No	Yes indicates output values will be saved.
Output Values Precision	Integer	Indicates the precision to use for output values.
Confirm Overwrite	Yes/No	Yes requires interactive confirmation when saving the model to a preexisting file.

• Set Account Method

Set the selected method in one of an account’s method categories.

Setting	Type	Notes
Account Name	Text	Choose the account using the selector
Category	Text	Choose the category on the account
Method	Text	Choose the desired method

• Set Accounting Period

Set the time range of the accounting system (i.e., the Begin Accounting Period and End Accounting Period dates).

Setting	Type	Notes
Start Date	Datetime	See “Datetime Information” on page 14 for details.
End Date	Datetime	See “Datetime Information” on page 14 for details.

• Set Controller

Set the current controller.

Setting	Type	Notes
Controller	Selection	Choose the controller from the list

Set Dataset File

Set the DSS file for a specified DSS Dataset or the Excel workbook for a specified Excel Dataset.

Setting	Type	Notes
Dataset	Text	Enter or choose the name of a DSS Dataset or an Excel Dataset defined in the Dataset Manager.
File Name	Text	Enter or choose the DSS file or Excel workbook.
Allow File Editing	Selection	Yes means the file can be entered or chosen in the Dashboard. No means that the file cannot be changed in the Dashboard.

Set Excel Dataset Run Name

For the specified Database DMI Excel Dataset, set the run name type and/or specify a new single run name.

Setting	Type	Notes
Excel Dataset	Text	Enter or choose the name of an Excel Dataset defined in the Dataset Manager. See "Excel Datasets" in *Data Management Interface (DMI)* for details on Excel Datasets.
Run Name Type	Selection	Choose one of the available Run Name Types either: • Single Run Name • Run with MRM Number (i.e. Run0, Run1, ...) • Trace with MRM Trace Number (i.e. Trace1, Trace2,...)
Single Run Name	Text	This setting is shown when the above setting is set to Single Run Name. Enter a new Single Run Name or choose one from the list; see "Run Name Type" in *Data Management Interface (DMI)* for details. In the Script Dashboard you can enter a new Single Run Name or choose one from the same list.

Set Init. Rules Exec. Flag

Set the Execute Initialization Rules run control flag. This flags controls whether or not the initialization rules are executed as part of the run. The flag is independent for each major controller, Simulation, Rulebased Simulation and Optimization. For more information, see "Can I disable the Initialization Rules?" in *RiverWare Policy Language (RPL)*.

Setting	Type	Notes
Sim and RBS	Yes/No	Yes = execute initialization rules before each Simulation or Rulebased Simulation run, including those controllers that include Accounting.
Optimization	Yes/No	Yes = execute initialization rules before an Optimization run.

Set Method

Set the selected method in the specified category for a set of objects.

Setting	Type	Notes
Object Name	Text	Choose one or more the objects using the selector
Category	Text	Choose the category on the object. The list shown is for the first object.
Method	Text	Choose the desired method. If the category or method is not possible on the object, an error will be issued.

Set MRM Descriptor

In a given MRM configuration, set the specified keyword, value pair MRM Descriptor. The descriptors are shown on the **Description** tab of the MRM configuration. See "Description" in *Solution Approaches* for details.

Setting	Type	Notes
MRM Configuration	Text	Enter or choose the name of an MRM configuration defined in the MRM Run Controller
Keyword	Text	Specify the Keyword. If the Keyword does not exist in the configuration, a new one is created.
Value	Text	Specify the desired Value to use with the Keyword.

Set MRM Ruleset

In a given MRM configuration, set a new file path to a ruleset.The ruleset is shown on the **Policy** tab of the MRM configuration. See "Policy" in *Solution Approaches* for details.

Setting	Type	Notes
MRM Configuration	Text	Enter or choose the name of an MRM configuration defined in the MRM Run Controller
Ruleset	Integer	Specify which of the rulesets you wish to change. The default is the first ruleset, 1.
File Name	Text	Enter the file path. See "Filename Information" on page 16 for details.

Set MRM Run Range

Set the time range of the run (i.e. the Run Start and End dates) for the specified MRM configuration. The run range is shown on the **Run Parameters** tab of the MRM configuration. See "Concurrent Runs" in *Solution Approaches* for details.

Setting	Type	Notes
MRM Configuration	Text	Enter or choose the name of an MRM configuration defined in the MRM Run Controller
Start Date	Datetime	See "Datetime Information" on page 14 for details.

Setting	Type	Notes
End Date	Datetime	See "Datetime Information" on page 14 for details.
Allow Start Date Editing	Yes/No	Yes indicates that users should be allowed to override the start date in the Dashboard.
Allow End Date Editing	Yes/No	Yes indicates that users should be allowed to override the end date in the Dashboard.

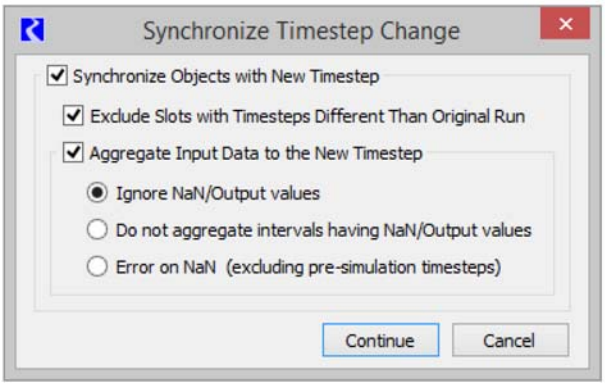
Set Run Range

Set the time range of the run (i.e., the Run Start and End dates).

Setting	Type	Notes
Start Date	Datetime	See "Datetime Information" on page 14 for details.
End Date	Datetime	See "Datetime Information" on page 14 for details.
Allow Start Date Editing	Yes/No	Yes indicates that users should be allowed to override the start date in the Dashboard.
Allow End Date Editing	Yes/No	Yes indicates that users should be allowed to override the end date in the Dashboard.

Set Run Timestep

Set the run timestep size. This action is analogous to changing the run control timestep size. The settings in the action are analogous to the options presented in the Synchronize Timestep Change dialog; see "Synchronize From Run Control" in *User Interface* for details.



Setting	Type	Notes
Timestep	Selection	Choose from 5 Minute, 15 Minute, 1 Hour, 6 Hour, 12 Hour, 1 Day, 1 Month, or 1 Year.
Synchronize Objects with New Timestep	Yes/No	Should objects be synchronized to the new timestep?

Setting	Type	Notes
Exclude Slots with Timesteps Different Than Original Run	Yes/No	Should slots with a timestep that is different than the original run be synchronized with the new timestep?
Aggregate Input Data to the New Timestep	Yes/No	When changing to a larger timestep (1 Month or 1 Year), should Input values be aggregated?
Aggregate NaN Handling	Selection	If aggregating, how should NaN and Output values in input slots be handled? See "Synchronize From Run Control" in *User Interface* for details on aggregation.

• Set Scalar Slot Value

Set one or more Scalar Slot values.

Setting	Type	Notes
Slots	Slot Selection	Choose the scalar slots using the selector
Set by	Value/Function	This setting indicates how you wish to set the slot, either by specifying values or by a global RPL function. When set to "Value", the "Value" setting and Allow editing setting, below, are shown. When set to "Function", the RPL Global Function setting is shown. **Note:** Only numeric or datetime slots can be set via the "Function" option.
Value	Value	If the "Value" option is selected in the "Set by" setting, specify the value to set at each timestep specified. This value can remain or be set to NaN.
RPL Global Function	Single Selection	If the "Function" option is selected in the "Set by" setting, use the RPL function chooser to select the function.
Allow Editing	Selection	No indicates that the value can not be changed in the Dashboard, Yes: Editor indicates that it can be changed using a line editor; Yes: Radio Buttons indicate that a set of radio buttons should be presented in the Dashboard to override the value; and Yes: Slider indicates that a slider should be presented to allow changes to the value. See "Configuring User Control on the Script Dashboard" on page 19 for details.
Minimum Edit Value	Numeric (slot value)	Specify the minimum value that is allowed in the Dashboard (applies to Slider only).
Maximum Edit Value	Numeric (slot value)	Specify the maximum value that is allowed in the Dashboard (applies to Slider only)

Setting	Type	Notes
Encoding Table Slot Name	Slot Selection	The table slot row labels and values establish a textual encoding of values. Applies only to Radio Button display. See “Configuring User Control on the Script Dashboard” on page 19 for details.
Show Current Value	Yes/No	When Allow Editing is Yes, then this setting allows you to show the current value for the scalar slot on the Dashboard. This only works when one scalar slot is specified.
RPL Global Function	RPL function selector	When using the “Set by:” “Function” mode, specify the Global Function to use. Currently, this function must return a Numeric value or a Datetime. The function is passed the SLOT as an argument. Use these as desired. See “Setting Slot Values using a Global Function” on page 21 for details.

• Set Series Slot Flags

Set the specified flag (input or output) on a set of series slots within the specified time range.

Setting	Type	Notes
Slots	Slot Selection	Enter or choose one or more series slots. The flags in these slots will be set to the specified value.
Column	Value or Text	For multi-column series slots, specify the column as either the column number (zero based) or column heading. If not specified, the first column is used.
Start Date	Datetime	The first timestep for which you wish to set the flag. See “Datetime Information” on page 14 for details.
End Date	Datetime	The last timestep for which you wish to set the flag.See “Datetime Information” on page 14 for details.
Flag	Selection	The flag to set on each value in the range.

• Set Series Slot Values

Set Series Slot values within a time range.

Note: This action does not support the setting of Text Series Slots.

Setting	Type	Notes
Slots	Slot Selection	Choose the series slot using the selector
Set by	Value/Function	This setting indicates how you wish to set the slot, either by specifying values or by a global RPL function. When set to “Value”, the “Value” setting and editing setting, below, are shown. When set to “Function”, the RPL Global Function setting is shown. **Note:** Only numeric or datetime slots can be set via the “Function” option.
Column	Value or Text	For multi-column series slots, specify the column as either the column number (zero based) or column heading. If not specified, the first column is used.

Setting	Type	Notes
Value	Value	If the “Value” option is selected in the “Set by” setting, specify the value to set at each timestep specified. This value can remain or be set to NaN.
RPL Global Function	Single Selection	If the “Function” option is selected in the “Set by” setting, use the RPL function chooser to select the function.
Start Date	Datetime	See “Datetime Information” on page 14 for details.
End Date	Datetime	See “Datetime Information” on page 14 for details.
Allow Editing	Selection	No indicates that the value can not be changed in the Dashboard, Yes: Editor indicates that it can be changed using a line editor; Yes: Radio Buttons indicate that a set of radio buttons should be presented in the Dashboard to override the value; and Yes: Slider indicates that a slider should be presented to allow changes to the value.
Minimum Edit Value	Numeric (slot value)	Specify the minimum value that is allowed in the Dashboard (applies to Slider only).
Maximum Edit Value	Numeric (slot value)	Specify the maximum value that is allowed in the Dashboard (applies to Slider only).
Encoding Table Slot Name	Slot Selection	The table slot row labels and values establish a textual encoding of values. Applies to Radio Buttons only. See “Configuring User Control on the Script Dashboard” on page 19 for details.
RPL Global Function Name	RPL function selector	When using the “Function” mode, specify the Global Function to use. Currently, this function must return a Numeric or a Datetime depending on the slot’s unit type. The function is passed the SLOT and DATETIME as arguments. Use these as desired. See “Setting Slot Values using a Global Function” on page 21 for details.

• Set Table Slot Value

Set a single Table Slot value at the specified row and column.

Setting	Type	Notes
Slots	Slot Selection	Choose the table slot using the selector
Row	Text	Specify the row as either the row number (zero based) or the row heading
Column	Text	Specify the column as either the column number (zero based) or column heading
Set by	Value/Function	This setting indicates how you wish to set the slot, either by specifying values or by a global RPL function. When set to “Value”, the “Value” setting and editing setting, below, are shown. When set to “Function”, the RPL Global Function setting isshown. **Note:** Only numeric and datetime slots can be set via the “Function” option.
Value	Value	If the “Value” option is selected in the “Set by” setting, specify the value to set at each timestep specified. This value can remain or be set to NaN.
RPL Global Function	Single Selection	If the “Function” option is selected in the “Set by” setting, use the RPL function chooser to select the function.

Setting	Type	Notes
Allow Editing	Selection	No indicates that the value can not be changed in the Dashboard, Yes: Editor indicates that it can be changed using a line editor; Yes: Radio Buttons indicate that a set of radio buttons should be presented in the Dashboard to override the value; and Yes: Slider indicates that a slider should be presented to allow changes to the value.
Minimum Edit Value	Numeric (slot value)	Specify the minimum value that is allowed in the Dashboard (applies to Slider only).
Maximum Edit Value	Numeric (slot value)	Specify the maximum value that is allowed in the Dashboard (applies to Slider only)
Encoding Table Slot Name	Slot Selection	The table slot row labels and values establish a textual encoding of values. Applies to Radio Buttons only. See “Configuring User Control on the Script Dashboard” on page 19 for details.
RPL Global Function Name	RPL function selector	When using the “Function” mode, specify the Global Function to use. Currently, this function must return a Numeric or a Datetime depending on the slot’s unit type. The function is passed the SLOT, row NUMERIC, and column NUMERIC, as arguments. Use these as desired. See “Setting Slot Values using a Global Function” on page 21 for details.

• Synchronize Objects

Set the time range for the specified objects.

Setting	Type	Notes
Objects to Synchronize	Object Selection	Use the selector to choose the desired objects.
Exclude Slots with Non-run Timestep	Yes/No	Yes indicates that slots with different timesteps will be excluded from the synchronize action.
Start Date	Datetime	See “Datetime Information” on page 14 for details.
End Date	Datetime	See “Datetime Information” on page 14 for details.

• Synchronize Slots

Set the time range for the specified slots.

Setting	Type	Notes
Slots to Synchronize	Slot Selection	Use the selector to choose the desired slots.
Start Date	Datetime	See “Datetime Information” on page 14 for details.
End Date	Datetime	See “Datetime Information” on page 14 for details.

Script Dashboard

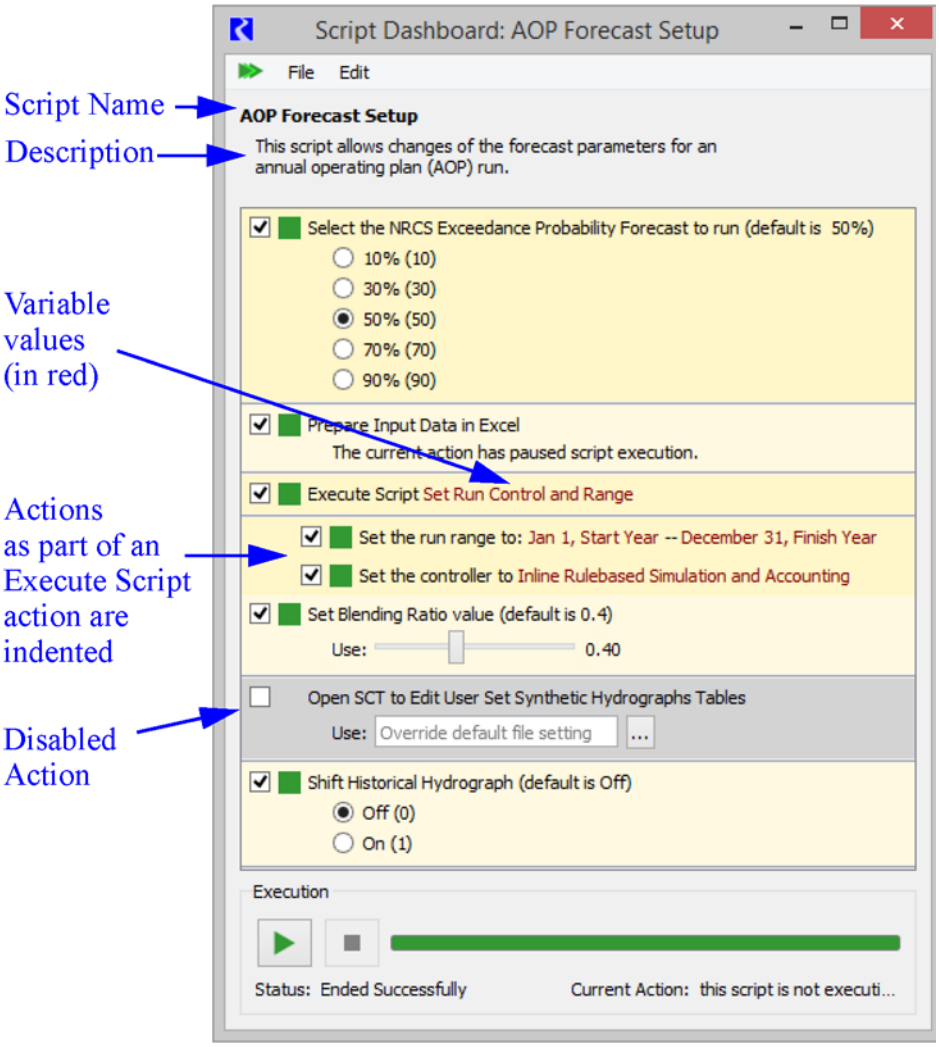
The Script Dashboard is the main interface for executing the script. This dialog presents a summary and controls for the script. All settings are configured in the Script Editor; see "Editing a Script" on page 8 for details. Figure 1.19 identifies features of this dialog.

- The script variable values are shown in red. Editable values are shown as either text edit fields, buttons, or sliders.
- Actions which are part of an Execute Script action (i.e. a subscript) are indented.
- Disable an action by selecting the check box. This turns that row grey. Use **Edit,** then **Enable All Action** to enable all actions shown in the dashboard at once; similarly **Edit,** then **Disable All Action** disables them all.
- Use the **Edit,** then **Reset Values** option to revert all values in the Script Dashboard back to the default value as defined in the Script Editor.
- To run a script, specify any desired override values (where configured) in the Script Dashboard and then select the **Run** button 

  Note: You can also execute a script from batch mode using the following RCL command:

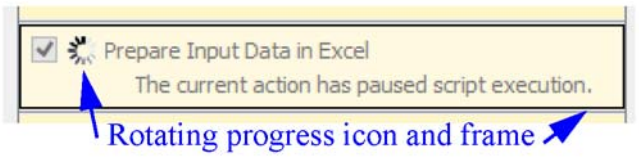
  ```
  ExecuteScript <Script Name>
  ```

Figure 1.19

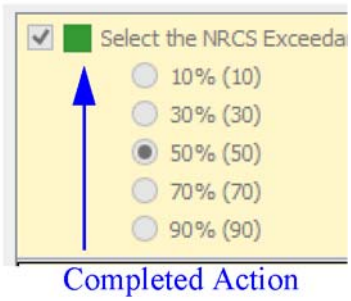


The Script Dashboard provides a progress bar at the bottom of the dialog showing the current action and the overall progress (based on the number of actions). As script actions execute, the Script Dashboard scrolls to keep the currently executing action in view.

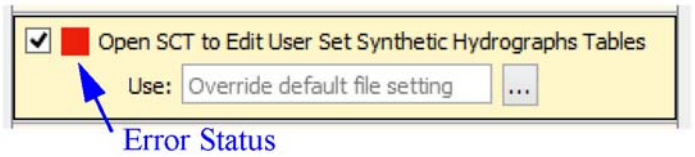
A rotating icon to the right of the **Enable/Disable** checkbox and darkened frame progress through the script to indicate the currently executing action.



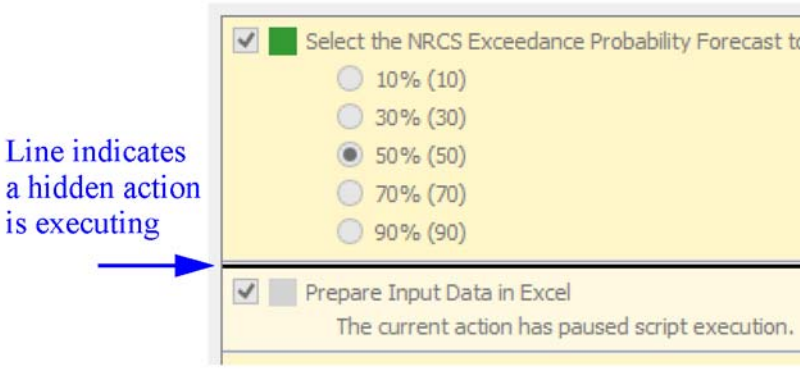
Actions that have completed successfully are indicated by a small green square to the right of the **Enable/Disable** checkbox.



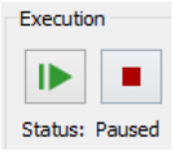
Actions resulting in an error, show a red square indicator.



During script execution, a dark horizontal line between actions indicates a hidden action (one that is not shown in the dashboard as configured in the editor) is executing.



If the script fails or a Memo with confirmation is encountered, a dialog will open alerting you. For the Memo, you must select the **Continue** button to resume execution.



Note: The script developer can specify which actions are shown on the Script Dashboard. Therefore, the dashboard may not contain all the actions executed by the script.

Chapter 2
Batch Mode and RiverWare Command Language

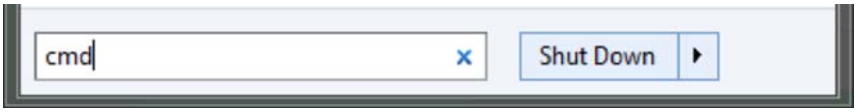
This section describes methodology to run RiverWare from the command line and optionally execute a RiverWare Command Language (RCL) script. This Batch Mode allows you to run RiverWare without seeing the user interface and allows for automation of many processes including advancing the run dates, running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

Topics

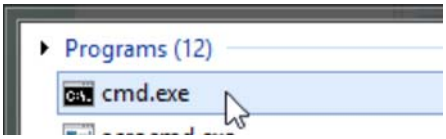
- Running RiverWare From the Command Line, page 41
- Running RiverWare in Batch Mode, page 42
- About RiverWare Command Language, page 43
- RCL Commands Reference, page 44

Running RiverWare From the Command Line

Running RiverWare from the command line provides an alternative to selecting 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, in the Windows Start menu Search field, type `cmd`.

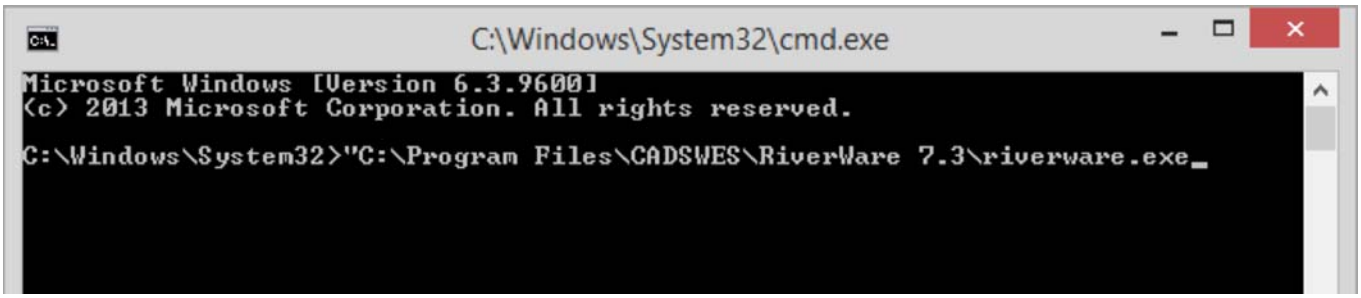


Then select the option for **cmd.exe**.



A window similar to that shown in Figure 2.1 will open. To start RiverWare from this window, enter the full RiverWare path (depending on your PATH configuration).

Figure 2.1 Starting RiverWare from the Windows Command Line



A complete list of command line options can be viewed by invoking RiverWare with the `--help` command line argument. On the Windows command line, type:

```
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.

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 following example will invoke the batch script file called `BasinOperationsScript.rcl` and will write the diagnostics output to file called `BasinOperationsDiagnostics.txt`.

```
riverware --batch "C:\MyRiverWareFiles\BasinOperationsScript.rcl" --log "C:\MyOut-
puts\BasinOperationsDiagnostics.txt"
```

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 RCL 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.

About RiverWare Command Language

This section provides general information about the RiverWare Command Language (RCL) syntax and variables.

Syntax 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 "#" 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 as follows:
  `\\\\ComputerName\\ShareName\\...`

Environment Variables

Environment variables are *system variables* that allow portability of models across machines and platforms.

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; see "SetEnv" on page 53. This command has the following 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.

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 `{}`.

Special Parameters

- `{date time}`

  A date and time of the format `{MM/DD/YYYY HH:MM}` or `{MM-DD-YYYY HH:MM}`, where the year includes the century and the hour is 24-hour based.

  Note: 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`," or "`MONTHS`."

  Note: A delta time, which includes embedded white space, must be enclosed in braces `{}`.

RCL Commands Reference

Following are the RCL commands in alphabetical order.

- **CloseWorkspace**

  ```
  CloseWorkspace
  ```

Clears the workspace.

> **Note:** The `CloseWorkspace` command does not close RiverWare. RiverWare closes once the execution of the complete batch script is complete. Multiple `OpenWorkspace` and `CloseWorkspace` commands within the same script will be executed within a single RiverWare session.

• ConfigureExcelDataset

```
ConfigureExcelDataset <Dataset Name> [workbook=...] [sheet=...]
```

This command changes the workbook and/or the sheet for an Excel Dataset used in a Database DMI.

- `<Dataset Name>` is an existing Excel dataset in the model. If the dataset does not exist in the model or is not an Excel dataset, the batch script will fail with an error message.

  > **Caution:** If the model is saved using the SaveWorkspace command, the modified Excel datasets will be saved.

Example

```
ConfigureExcelDataset {My Excel Dataset} {workbook=C:/My Data/My Workbook.xlsx}
ConfigureExcelDataset {My Excel Dataset} {sheet=MyNewSheetName}
```

• CreateSlotCache

```
CreateSlotCache
```

Creates a slot cache that contains the current values of all series slots on the workspace, in the range of the current controller. See "Slot Cache" in *User Interface* for a description of the cache.

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

• EnableInfoDiag

```
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.

• ExecuteScript

```
ExecuteScript <Script Name>
```

Executes the specified script as defined in the Script Manager. See "Script Management" on page 1 for details on RiverWare Script Management. The script name should be enclosed in curly braces `{}` if it contains embedded white space.

Example:

```
ExecuteScript {Run Operations Model}
```

• ExportOptAnalysisAsCSV

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

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

- `<file name>` is the name of the file to write. If the file path contains embedded white space, it should be enclosed in curly braces `{}`.
- `<section name>` are the (optional) sections to export. Defaults to all sections. The section names are as follows and correspond to combinations of the Optimization sections; see "Export as CSV" in *Optimization* for details.

  ```
  ConstraintAll
  ConstraintNew
  ConstraintPhysical
  ConstraintPolicy
  ConstraintPrior
  None
  SolutionAndPolicy
  SolutionGoalSummary
  SolutionInfo
  SolutionSingle
  VariableAll
  VariableOther
  VariableSlot
  ```

- 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: 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: If this argument is present, all previous arguments must also be present.

Example:

```
ExportOptAnalysisAsCSV {C:\MyOutputs\MyOptAnalysis.csv} {PolicyConstraints} {true}
```

• GetRunInfo

```
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.

Following are examples of the syntax:

```
GetRunInfo #RunInfo !InitDate
GetRunInfo #RunInfo !EndDate
```

GetSlot

```
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 curly braces `{}` 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.

HDB_EnsembleID

```
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.

HDB_ModelRunID

```
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.

InvokeDMI

```
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`".

Example:

```
InvokeDMI {Operations Data Input DMI}
```

InvokeDssDMI

```
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 `{}`.

ListDMI

```
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.

Examples:

```
ListDMI C:/MyOutputs/DMIListFile.txt
```

Writes the names of all DMIs to the file DMIListFile.txt.

```
ListDMI !Type Input C:/MyOutputs/DMIListFile.txt
```

Writes the names of all Input DMIs to the file DMIListFile.txt.

```
ListDMI !Type Input !Dataset DSS C:/MyOutputs/DMIListFile.txt
```

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

• LoadDiagSettings

```
LoadDiagSettings <file path>
```

Loads the diagnostic settings from the specified file.

- `<file path>` is the path to the diagnostics setting file.

The diagnostic settings file must have been saved from the model file beforehand. See "Loading and Saving Diagnostic Configurations" in *Debugging and Analysis* for information on saving diagnostic settings to a file. As an example, you might have preconfigured diagnostic settings for printing out detailed informational diagnostics for debugging purposes when you run in batch mode.

> **Caution:** If the model is saved using the SaveWorkspace command, it will be saved with these new diagnostic settings.

Example:

```
LoadDiagSettings C:/MyRiverWareFiles/DebugDetails_DiagSettings.txt
```

• LoadOptSet

```
LoadOptSet <file path>
```

Loads the optimization goal set into the workspace.

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

Example:

```
LoadOptSet C:/MyRiverWareFiles/MyOptGoalSet.opt
```

• LoadRules

```
LoadRules <file path>
```

Loads a ruleset into the workspace.

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

Example:

```
LoadRules C:/MyRiverWareFiles/MyRuleset.rls
```

• OpenGlobalSet

```
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.

Example:

```
OpenGlobalSet C:/MyRiverWareFiles/MyGlobalFunctions.gfs
```

• OpenWorkspace

```
OpenWorkspace <file path>
```

Loads a model into the workspace.

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

Example:

```
OpenWorkspace C:/MyRiverWareFiles/BasinOperationsModel.mdl
```

• Output

```
Output <outputDeviceName>
```

Generates an output file that is specified in the Output Manager of the model. See "Output Manager" in *Output Utilities and Data Visualization* for details about the available output devices and how they can be configured. Plot pages cannot be generated from a RCL script.

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

Examples:

```
Output OperationsForecastReport
Output FloodStorageAllocationChart
```

• RequiredInfo

```
RequiredInfo <message>
```

This command adds the specified message to the diagnostic output as a Required Information message. It is a means to allow you to specify any text in the RCL script to be added to the diagnostic output file.

- `<message>` is any user-specified text string. The text must be enclosed in braces {} if it includes spaces.

Example:

```
RequiredInfo {Executing a planning run}
```

Output in the diagnostic output file:

```
_REQINFO_:    "Executing a planning run"
```

• RequireVersion

```
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.

Example

```
RequireVersion 7.3.1
```

If the version of RiverWare is earlier than 7.3.1, the batch script will fail with an error message.

ReorderGoalSet

```
ReorderGoalSet <order slot>
```

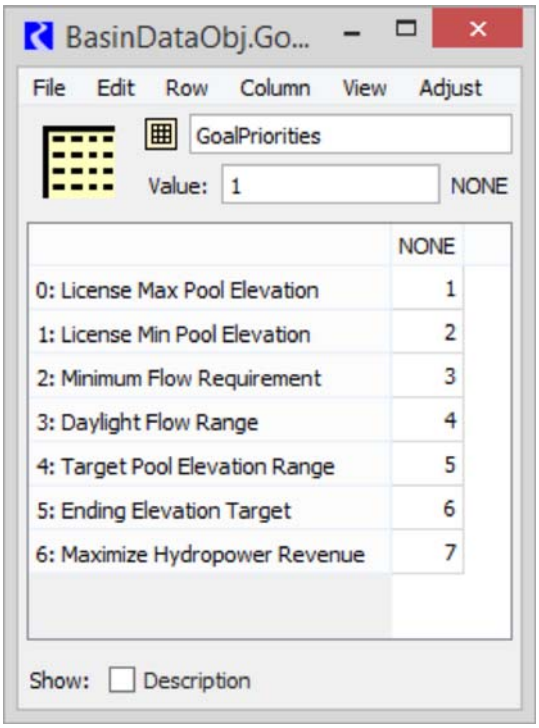
This command allows you to reorder items (goals, groups, or certain statements) within the currently loaded optimization goal specified set. The desired order is specified by the single column order slot, a table slot, each of whose rows corresponds to the name of an item in the goal set. The values 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; see "Reorder RPL Set" on page 29.

Example:

```
ReorderGoalSet BasinDataObj.GoalPriorities
```

The command refers to the table slot shown in Figure 2.2.

Figure 2.2 Table slot with priorities to reorder the goal set

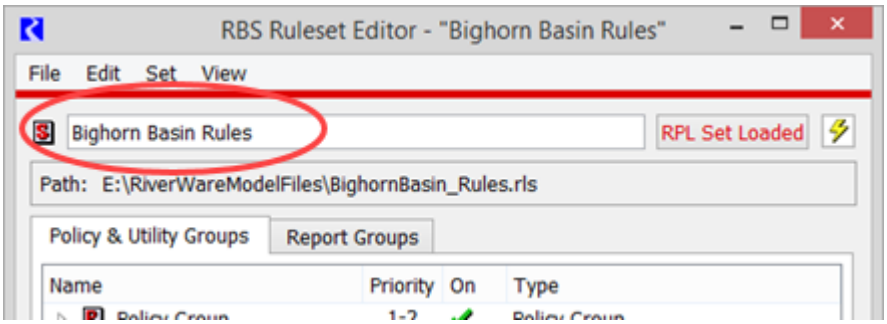


SaveRplSet

```
SaveRplSet <Set Name> <Application Type> <File Path>
```

This command allows you to save to a file a RPL set that is currently open in the model. If the specified RPL set is not open in the model, the batch script will fail with an error message.

- `<Set Name>` is the name of the RPL set to be saved. If the set name includes spaces, it must be enclosed in braces {}. This is the name displayed at the top of the RPL Set Editor, *not* the file name.



- `<Application Type>` refers to the type of RPL set. If the application type includes spaces, it must be enclosed in braces {}. The valid options are

  ```
  Global Functions
  Initialization Rules
  Optimization
  Rule Based Simulation
  ```

- `<File Path>` is the path to where the RPL set is saved. If the file already exists, the command will overwrite the existing file.

Examples:

```
SaveRplSet {Bighorn Basin Rules} {Rule Based Simulation} C:\MyFiles\Bighorn.rls
SaveRplSet {Bighorn Opt Goals} Optimization C:\MyFiles\BighornOpt.opt
```

SaveWorkspace

```
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.

Examples:

```
SaveWorkspace
```

Saves the model to the same file name from which it was opened

```
SaveWorkspace C:/MyOutputs/SavedModelFile.mdl
```

Saves the model to the file specified

SetDiagFile

```
SetDiagFile <diagnostic file>
```

Sets the location of the diagnostic output file.

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

SetEnv

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

Sets an environment variable for use within the RiverWare run.

See "Environment Variables" on page 43 for details on setting and using environment variables.

SetRunInfo

```
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` with the same `SetRunInfo` command, the argument that 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 `{Number TimeUnits}`.
- `!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: Controller types that contain embedded white space must be enclosed in braces `{}`.

Following are samples of the syntax:

```
SetRunInfo #RunInfo !InitDate {04-21-2011 24:00} !EndDate {04-26-2011 24:00}
SetRunInfo #RunInfo !StartDate {04-22-2011 24:00}
SetRunInfo #RunInfo !EndDate {04-26-2011 24:00}
SetRunInfo #RunInfo !Duration {200 Days}
SetRunInfo #RunInfo !Controller {Rulebased Simulation}
SetRunInfo #MRM "Paleo Runs"
```

SetSlot

```
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 curly braces `{}` if it contains embedded white space.

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

See "GetSlot" on page 47, for details on possible usage.

SetTrace

```
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.

SlotList

```
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.

SlotListDMI

```
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.

StartController

```
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.

SyncObj

```
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 size differs from the Run Control timestep size.
- `[!StartDate <start date>]` synchronizes slots to the specified start timestep, rather than the Run Control initial timestep.
- `[!EndDate <end date>]` synchronizes slots to the specified end timestep, rather than the Run Control finish timestep.

Examples:

```
SyncObj
```

Synchronizes all series slots to the Run Control initial timestep and finish timestep.

```
SyncObj !ExDiffTS !StartDate {04-23-2013 12:00} !EndDate {04-26-2013 12:00}
```

Synchronizes series slots to the specified start and end timesteps and excludes slots that have a timestep size that differs from the Run Control timestep size.

Chapter 2
Batch Mode and RiverWare Command Language