Technical Documentation Version 7.3

RPL User Interface

Center for Advanced Decision Support for
Water and Environmental Systems (CADSWES)
UNIVERSITY OF COLORADO **BOULDER**

RPL User Interface Table of Contents

RPL Sets 1

Types of RPL Sets 1
Set Name 2
Save Location 2
Managing RPL Sets 3
- Actions specific to Rulesets, Optimization Goal Sets, and Global Function Sets .. 3
- Actions specific to Accounting Method Set, Expression Slot Set, Initialization Rules, and Iterative MRM Sets 5

Tour of a RPL Set 6
- Elements 6
- RPL Set Editor View 8

Editing RPL Sets 11
- Blocks and Groups 11
- Validity 14

Comparing RPL Sets 14
- Accessing the Comparison Tool 14
- Selecting RPL Sets to Compare 15
- Tour of the RPL Set Comparison Tool 15
- Example of using the RPL Set Comparison Tool 18

Exporting and Importing RPL sets 22
- Export 23
- Import 23

RPL Editor Dialogs 25

RPL Viewer vs RPL Editor 25
Working with RPL Dialogs 27
- Execution Constraint / Execute Block Only When 28
- Descriptions 28
- Notes 28
- Comments 29
- Executing DMIs from Blocks 29
- Stop On NaN 30

Statements 30
Editing a RPL Expression 33
- Using the palette 33
- Entering Values 34
- Undo and Redo 35

Using the History 36
Data types for Looping Variables 36
Renaming Looping Variables and Function Arguments 37
RPL Short Cuts 37
Disabling an Item in a List or a Statement 38
Open Slots and Objects from RPL dialogs 38
RPL Search and Replace Dialog 39
Accessing the dialog 39
Searching for occurrences of a string 39
Replacing matching strings with another string 41
Functions 41
Predefined Functions 41
Writing a User-Defined Function 42
Constraints on Functions 44
Time Invariant Functions and Function Value Caching 44
Selecting RPL Items 45
Developing Efficient RPL Expressions 46

Example: Creating a new RBS Ruleset 47

Initialization Rules Set 50

Global RPL Functions 53

Creating a new Global RPL Function Set 54
Opening an Existing Global Function Set 55
Using Global RPL Functions 56

RPL Printing and Formatting 56

Printing 56
Formatting - The Display Settings 57
Font 58
Colors 58
Line Breaks 59
Element Numbers 63

RPL External Documentation 64

Overview 64
File Association 65
Document Structure 66
Configuration and User Interface 67
Viewing and Editing 70

HTML **71**
MS WORD **73**
PDF **74**
Text **74**
Use Example **74**

RPL Sets and Editors

1. RPL Sets

This document describes basic rule, function, method, or constraint construction with the RPL Set Editor. The RPL Set Editor is the main window through which policy is managed. From this window:

- Rules, functions, or methods, may be: added, deleted, opened, named, prioritized and turned off or on.
- Entire RPL sets may be: opened, closed, saved, loaded, exported, imported and unloaded.

1.1 Types of RPL Sets

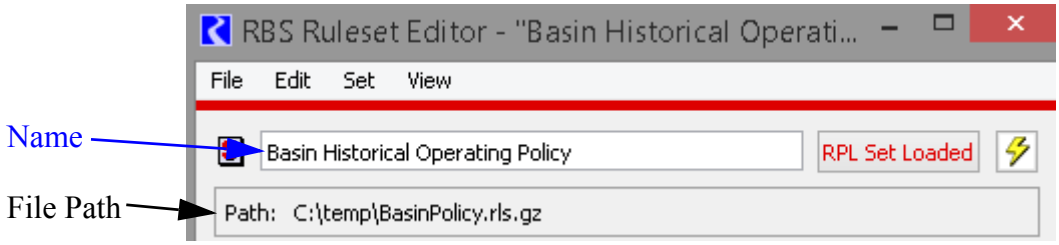
This document refers to all of the RPL sets in RiverWare as follows. The table shows information about each set that will be referred to later

RPL Set	Set Color	Save Location	Link to More information
Expression Slot Set	Orange	In model file	Click **HERE (Slots.pdf, Section 4.6)**
Global Function Set	Brown	Separate file **OR** in model file	Click **HERE (Section 5)**
Initialization Rules	Teal	In model file	Click **HERE (Section 4)**
Iterative MRM Ruleset	Navy Blue	In model file	Click **HERE (MRM.pdf, Section 4.3.3)**
Object Level Accounting Method Set	Green	In model file	Click **HERE (Accounting.pdf)**
Optimization Goal Set	Purple	Separate file **OR** in model file	Click **HERE (Optimization.pdf, Section 6.7)**
Rulebased Simulation (RBS) Ruleset	Red	Separate file **OR** in model file	Click **HERE (, Section 1.3)**

When dealing with specific menus, this document will use the terminology “Set” in place of any type of set. Please note that each of the sets has a unique menu name and should be used accordingly.

1.2 Set Name

Each set can have a user specified name that is separate from the file path. The default for a newly created set is “RPL Set N”. Enter your desired name in the field shown below.



Note: Prior to RiverWare 7.0, the name and the file name were the same. When loading an old set into 7.0, the name defaults to the file name. You can change the name if desired.

1.3 Save Location

Based on the type, the set is saved in one of two locations as shown in the table above:

- In the model file or
- In an external file

Four of the sets (expression slot set, initialization rules, iterative MRM rulesets, and accounting method set) are always saved in the model file. Global functions sets, Rulesets and Goal sets may be saved in either location as specified in the Run Parameter for the appropriate controller. This is accessed from the **Run Control** dialog. When the Rulebased Simulation or Optimization controller is selected, then the **View ➥ Optimization/Rulebased Simulation Run Parameters** menu opens the parameter dialog.

The toggle **Save Loaded RPL Set with Model** controls where the set is saved. By default, this toggle is **OFF** meaning the set is saved in a file external to the model. Checking it **ON** will save the **Loaded** set with the model file.

Note: This toggle only applies to the **Loaded** set. Any other opened sets will not be saved with the model file.

All of the **Run Parameter** dialogs contain an option to **Save All Global Functions Sets with Model** toggle. By default, this toggle is **OFF** meaning that all global functions sets are saved in files external to the model. Checking it **ON** will save all open global functions sets with the model file.

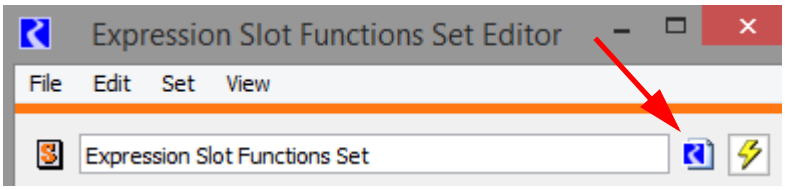
Once a set is saved with the model file, there is some risk that it may be lost if:

- the toggle above is unchecked and the model is saved
- the set is unloaded and the model is saved

- another set is loaded and the model is saved
- closing the set and the model is saved

In each case, a warning message is presented to allow you to confirm that the action is intended. A set saved with the model is automatically loaded when the model is opened. The set is then minimized. You can bring it to the front at anytime using the workspace **Policy** menu or the buttons on the bottom of the workspace.

Further, for any set that is saved with the model, a model file icon is displayed at the top right of the dialog.



Sets that are saved in an external file have the path shown below the set Name:

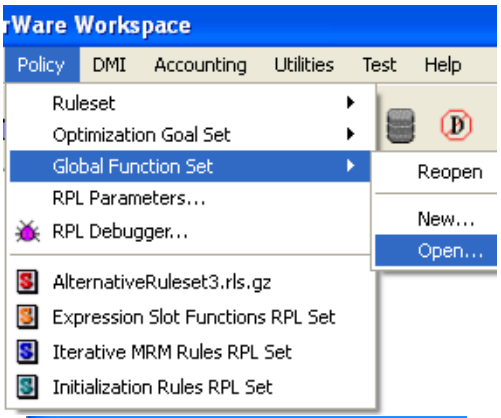
1.4 Managing RPL Sets

Following is a description of managing RPL sets. Much of this information is specific to the RPL set in question.

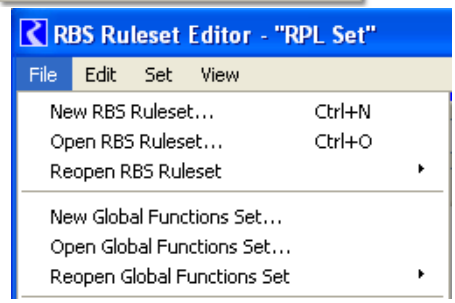
1.4.1 Actions specific to Rulesets, Optimization Goal Sets, and Global Function Sets

The following actions are specific to managing a set (either Ruleset, Optimization Goal Set, or Global Function Set) and are performed on an entire set:

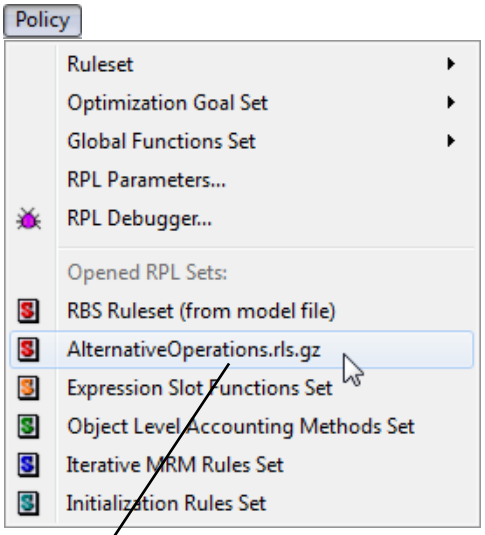
- **New**: A new RPL set is created by selecting **Policy ➥ Set ➥ New...** from the main RiverWare workspace or **File ➥ New Set...** from an already opened set.
- **Open**: An existing RPL set (saved as a file) is opened by selecting **Policy ➥ Set ➥ Open...** from the main RiverWare workspace or **File ➥ Open Set...** from an already opened set. Either of these commands invokes a file chooser in which the desired set is selected. Each set is saved in a separate file. Several sets may be opened at once. Each opened set and any windows opened from these sets are identified by a unique color band across the top of their windows.



- **Reopen**: An existing RPL set (saved as a file) can be reopened by selecting **Policy ➥ Set ➥ Reopen...** from the main RiverWare workspace or **File ➥ Reopen Set...** from an already opened set. Either option then presents a cascading menu of recently opened sets and folders.



- **Reopen and Load**: An existing RPL set (saved as a file) can be reopened and loaded by selecting **Policy ➥ Set ➥ Reopen and Load...** from the main RiverWare workspace. Then, a cascading menu of recently opened sets and folders is presented.
- **Show RPL set**: When a set is opened, the name of the set is added to the workspace **Policy** menu as shown in the screenshot to the right. Only the short name is shown in the menu, but the full path is shown in the workspace status bar. Use this menu to raise/show the desired set. When the set is saved with the model file, the menu will show it as "RPL Set". Note that the expression Slot, initialization rules, MRM and accounting methods sets (saved with the model) are always shown too. Also, the workspace has buttons to show the **loaded** RBS ruleset RBS or optimization goal set Opt. The bar is colored when there is a loaded set, grey when there isn't. Click on one of the buttons (when a set is loaded) to raise that set to the top.The workspace also shows color coded buttons for all the opened sets. Tool tips indicate their names.



- **Save**: The RPL set is saved to a file by selecting **File ➥ Save Set** in the **Set Editor** menu bar. The name and directory of the file to which the RPL set is saved is that from which it was last loaded or to which it was last saved.

- **Save As**: A new RPL set must be given a name and saved for the first time by selecting **File ➥ Save Set As...** in the **Set Editor** menu bar. This command is also used to save an open RPL set to a different filename and/or directory from which it was last loaded or saved. The **Save Set As...** command invokes a file chooser in which the RPL set's new path and filename must be specified.
- **Load**: A ruleset or goal set is loaded into a model for use during a run by clicking on the **RPL Set Not Loaded** button on the right side of the **Set Editor** menu bar. When the set is loaded, the button text changes to **RPL Set Loaded** and the colored bar along the top of the **Set Editor** becomes red. Only one ruleset at a time may be loaded for use by a model. Loading a set when another is already loaded will unload the first ruleset. Global Function Sets are not loaded, any open set applies to all other sets.
- **Unload**: A loaded set is unloaded from a model by clicking the **RPL Set Loaded** button on the right side of the **Set Editor** menu bar. When the set is unloaded, the button text changes to **RPL Set Not Loaded**, and the red bar along the top of the **Set Editor** reverts to its original color (blue, green, yellow, purple, etc.).
- **Close Window (Ctrl+W)**: An opened RPL set is closed but not removed by selecting **File ➥ Close Window** in the **Set Editor** menu bar. It can be re-shown using the Policy menu on the workspace.
- **Close (and Unload) Set**: An opened RPL set is closed by selecting **File ➥ Close (and Unload) Set** in the **Set Editor** menu bar. Closing a loaded set automatically unloads the set from the model. Closing does not automatically save changes to the set since the last save. This option is not available for those sets always saved with the model file.

1.4.2 Actions specific to Accounting Method Set, Expression Slot Set, Initialization Rules, and Iterative MRM Sets

The following actions are specific to the Object Level Accounting Method set, the Expression Slot set, Initialization Rules, and the Iterative MRM set. Each of these RPL sets is saved with the model so they do not need to be opened or saved separately.

To show the Object Level Accounting Method set (when accounting is enabled):

- From the Workspace choose the **Policy ➥ Accounting Methods RPL Set** or choose **Accounting ➥ Open Accounting Methods RPL Set**

To show the Expression Slot RPL set:

- From the Workspace choose the **Policy ➥ Expression Slot RPL Set**

To show the Initialization Rules RPL set:

- From the Workspace choose the **Policy ➥ Initialization Rules Set**

To show the Iterative MRM RPL set:

- From the Workspace choose the **Policy ➥ MRM RPL Set.** The MRM RPL set can also be opened from the MRM run control dialog. This is further described **HERE (MRM.pdf, Section 4.3.3)**.

These sets are always active in their respective context so it is not necessary to load or unload the sets. The dialog for the sets can be closed using the **File ➥ Close Set** menu. Again, each of these sets is saved with the model file so do not need to be separately saved.

The Initialization Rules Set Editor dialog does have a **File ➥ Save Initialization Rules Set As** menu item to save the initialization rules to a file. A **File ➥ Replace Initialization Rules Set from File** menu item allows the initialization rules set to be replaced by the contents of a specified file. These menu items allow an initialization rules set to be moved between models via a file. However there is only a single instance of initialization rules in a model and this set is still saved and loaded with the model file. More information on Initialization Rules can be found **HERE (Simulation.pdf, Section 5.1.2)**.

1.5 Tour of a RPL Set

In RPL sets there is an upper level construct specific to the RPL set. For example, in a ruleset, a rule is the construct; in an object level accounting method set, a method is the upper level construct. In this document, a *block* refers to the upper level construct for each of these sets and will be used in all descriptions. Following is a list of the block for each type of set:

	RPL Set	Upper level *Block*:
	Expression Slot Set	N/A
	Global Function Set	N/A
	Initialization Rules Set	Rule
	Iterative MRM Ruleset	Rule
	Object Level Accounting Method Set	Method
	Optimization Goal Set	Goal
	RBS Ruleset	Rule

Note: Expression Slot and Global Function Sets do not have the paradigm of a block. For Expression Slot Sets, the analogous upper level construct is the expression slot itself. Global Function Sets do not have the concept of a block as they contain only global utility groups and functions.

This remainder of this section describes the components of a RPL set when using the RPL set editor

1.5.1 Elements

A RPL set includes the following general elements:

- **Policy Groups:** Policy groups are containers for blocks and functions. The functions within a policy group are available only to blocks and functions within that policy group.
- **Utility Groups:** Utility groups are containers for functions. The functions within a utility group are available to blocks and functions within any policy group.

- **Blocks:** Blocks (Rules/Methods) are prioritized expressions of policy which assign slots and/or generate diagnostic messages.
- **Functions:** Functions are subroutines called from blocks or other functions, which evaluate to a value in one of the expression data types.
- **Report Groups:** Report groups provide an organizational grouping of related RPL and workspace objects.
- **Report Items:** Report items refer to blocks or functions in the RPL set or to workspace objects.

Following is a table showing all the RPL elements and their icons. Note, the icons are shown in grey but each set has its own color that is used for the letter:

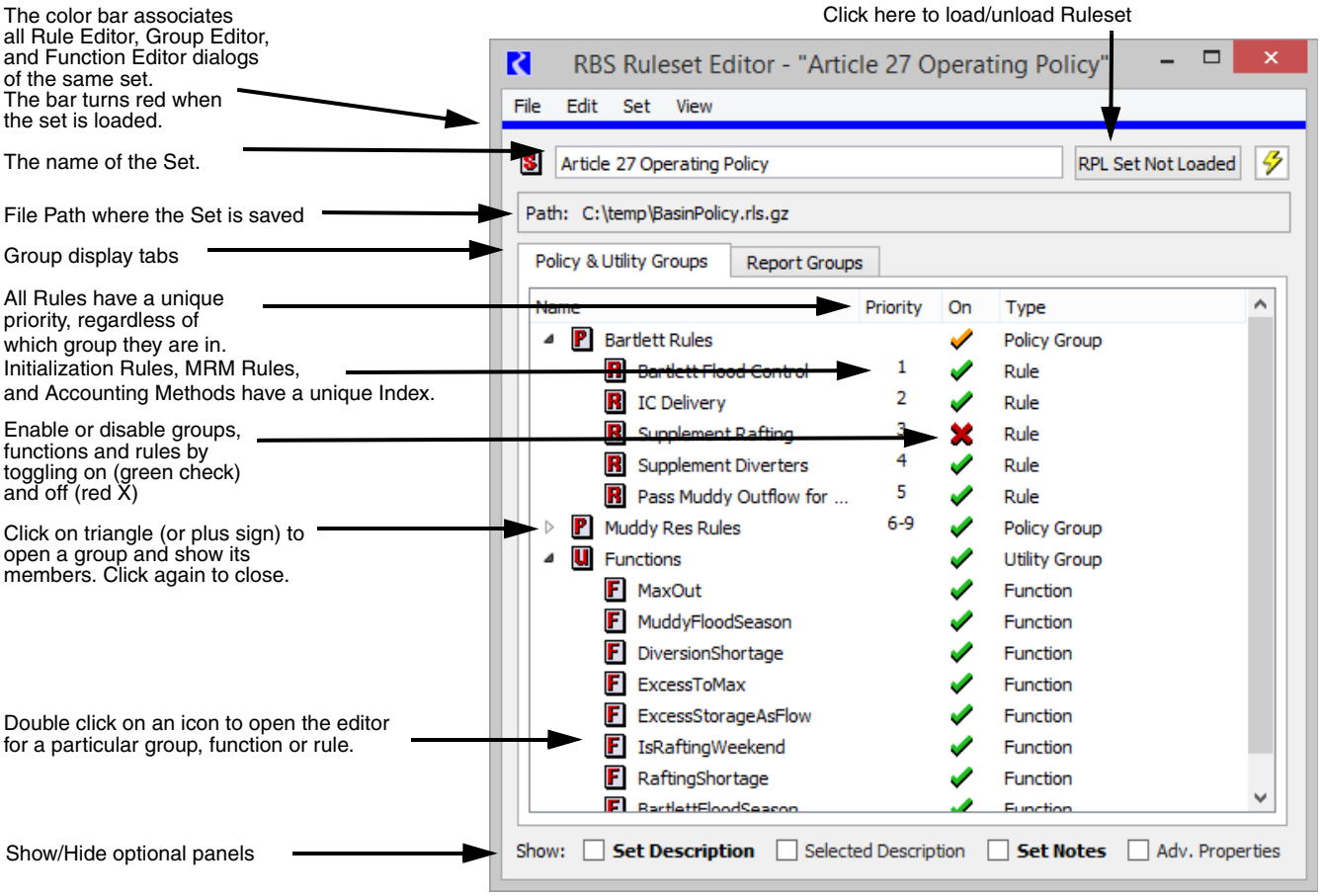
Letter	Icon	Item
G		Goal
M		Method
P		Policy Group
		Report Group
		Report Item

Letter	Icon	Item
R		Rule
S		Set
=		Statement
F		User-defined Function
U		Utility Group

Predefined functions use the F icon but always have the light blue color: 

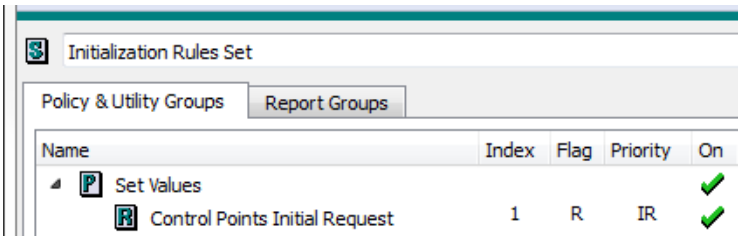
1.5.2 RPL Set Editor View

Information about a RPL set is displayed in the RPL Set Editor dialog. Each policy group, utility group, block, and function is represented in the RPL Set Editor on a different line.

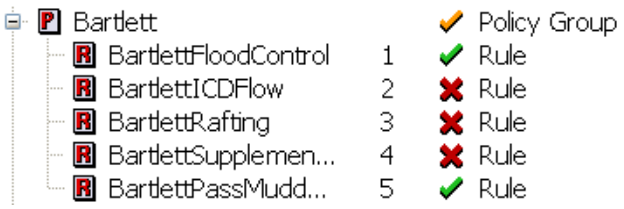


Each line contains several pieces of information:

- **Name:** The name column contains an icon representing the type of the element and its name. All elements are assigned a default name when created.
- **Priority:** A priority is shown to the right of an element's name, when appropriate. Policy groups, utility groups, and functions do not have priorities but policy groups display the range of priorities of its member rules. Priorities number indicates the relative importance of the block.
- **Index**: For Initialization Rulesets, Object Level Accounting Method Sets, and MRM Rulesets, the **Index** column is shown. The priority column may not be shown. Rules or methods have a unique **Index** which is how the rules/methods are referenced.

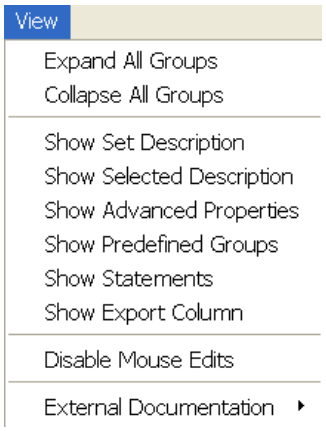


- **Flags**: Initialization Rules allow you to specify the flag that is set. For initialization Rulesets, a Flags column is shown.
- **On/Off Status:** All elements may be turned on or off independently by clicking on the green check mark or red X in their line, unless mouse editing has been disabled. If mouse editing has been disabled, a block/function can be turned on or off by right clicking. This will bring up a menu to turn the element on or off. A green check mark indicates that an element is active. A red X indicates that an element has been turned off. An element that is turned off cannot be used during the model run. Turning off a policy group or utility group disables all of the blocks and/or functions within that group. Clicking on a group's red X now turns the entire group "on" without making any changes. This prevents you from inadvertently changing the status of an item in a group that is off only to find that it is now different when the entire group in turned back "on". If all individual items in a group are turned off (red X), then the entire group is considered off and the group will also have a red X. If you click on the group's red X, a warning message will appear to indicate that at least one item in the group should be turned on. An orange check mark on a policy or utility groups specifies that one or more items in that group is "off" as shown in the following figure. As before, you can click on the orange check to turn "off" the entire group. Clicking on it again, restores the previous state thus showing an orange check.



- **Type**: The type of each item in a set is shown in the **Type** column.

There are several additional pieces of information that can optionally be shown from a RPL set as accessed from the **View** menu:



Expand/Collapse All: A RPL set's contents may be viewed with all groups expanded, some groups expanded, or all groups collapsed. To expand or collapse all groups simultaneously:

- Select **View ➥ Expand All Groups** or **Collapse All Groups** item in the RPL Set Editor menu bar.
- To expand or collapse a single group, click on the triangle next to the group name.

Note: The following four items can be shown using the View menu or clicking on the toggle in the **Show**: row. If there is an entry in these panels (or a non-default value) the text is **bold**. For example, in the above screenshot, the **Selected Description** and **Set Notes** have a non-default entry.

Show RPL Set Description: A description of the RPL set may be typed into the RPL Set Description: text frame and saved in the RPL set file. The description may contain any combination of letters,

numbers, spaces and punctuation, except double-quotes (“). The area is opened and closed by toggling the **View ➥ Show RPLSet Description** in the menu bar or using the **Set Description** toggle at the bottom of the window.

Show Selected Description: A special expanded area of the RPL set is used to enter a description of any specific block. The area is opened and closed by toggling the **View ➥ Show Selected Description** in the menu bar of the **RPL Set Editor** dialog or using the **Selected Description** toggle at the bottom of the window. Single click on a block’s name to view the description of that specific block.

Show Set Notes: Notes about the RPL set may be typed into the **Set Notes** text frame and saved in the RPL set file. The Notes may contain any combination of letters, numbers, spaces and punctuation, except double-quotes (“). The area is opened and closed by toggling the **View ➥ Show Set Notes** in the menu bar or using the **Set Notes** toggle at the bottom of the window. Notes are often used for commenting on development practices; they provide a place to annotate when/why/who changed the set.

Show Advanced Properties: The **Advanced Properties** are accessed from the **View ➥ Show Advanced Properties** menu or using the **Advanced Properties** toggle at the bottom of the window The **Advanced Properties** areas has two components to:

- **Agenda Order:** Rulesets can be configured to execute in either ascending or descending Priority/(it called Index in Initialization Rulesets) order. Descending order executes the highest priority block first, then each of the lower priority blocks (1,2,3,...). Ascending order executes the lowest priority block first, then each of the higher priority blocks (...,3,2,1). Ascending Priority (...3,2,1) is the default selection and the recommended priority order. The agenda order is only applicable and shown for RBS rulesets, initialization rules, iterative MRM rulesets and optimization goal sets. Also, optimization goal sets always must execute in the 1,2,3... order and cannot be changed.
- **Precision:** The precision spinner determines the number of digits after the decimal which are displayed for numbers in blocks and functions. 8 is the default value. The displayed precision has no effect on the 17 significant figures of internal precision used for calculations.

Show Predefined Groups: In the **View ➥ Show Predefined Groups** menu, you can choose to show the groups for the predefined functions. When you also show Selected Description above, the documentation for that predefined function is shown in the description panel.

Show Statements: In the **View ➥ Show Statements** menu, you can choose to show statements (assign, print, etc...) as another level in the RPL dialog’s tree-view. You can also change the name of statements once shown. To do this, right-clicks on that item to show the context sensitive menu. Then, selecting **Rename** will open an inline name editor and you can enter a new name. In addition to the RPL Set view, this statement name will be shown in the RPL debugger and RPL Analysis Tool.

Show Export Columns: The **View ➥ Show Export Column** menu allows you to specify if the selected row should be exported. Import/Export of RPL sets is described **HERE (Section 1.8)**.

Disable Mouse Edits: It is easy to rename a group, block or function, change the priority of a block, and turn a block on or off. To prevent accidental changing of a block's priority or turning the block on or off in the Set Editor as a result of mouse clicks, select **View ➥ Disable Mouse Edits.** This will prevent editing with mouse clicks in the RPL Set Editor. To edit a block or group after mouse editing is disabled, double click or right mouse click on the block, function, or group's name to bring up an editor dialog. We highly recommend disabling mouse editing to prevent accidental changes to a RPL set.

External Documentation: View, edit, or configure the external documentation as described **HERE (Section 7)**.

1.6 Editing RPL Sets

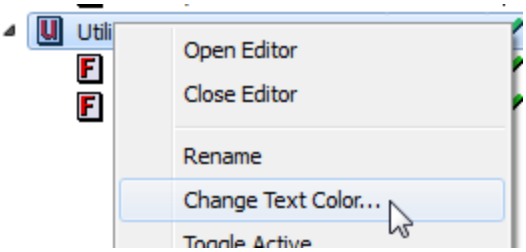
Following is a description of how to manage and edit RPL sets once they are open. This includes creating, naming, and editing blocks/groups, adding statements, and validating blocks or the entire set. Following this is a description of how to edit individual expressions, **HERE (Section 2.4)**.

When a RPL set dialog opens, the "**Policy & Utility Groups**" tab is initially selected, presenting a comprehensive high-level view of the policy. Switching to the "**Report Groups**" tab provides a more selective, user-defined view. Note that many actions described below are appropriate only for one or the other main tabs (e.g., adding a new Report Group), and these actions are disabled when they do not apply.

1.6.1 Blocks and Groups

Naming: To name a block or group, right click or double click on the name field of the block or group. Right clicking will bring up a menu from which the editor dialog can be opened. Double clicking will bring up an editor dialog directly. Type the name in the name field.

Name Color: The color of each name in the set can be specified by right clicking on an item and choosing **Change Text Color…** The selected color is then used by the RPL set editor, the RPL set analysis tool, and the RPL object selector dialogs for the name of the selected object. For groups, the color is changed for the group name and all members of the group that have the default black color.

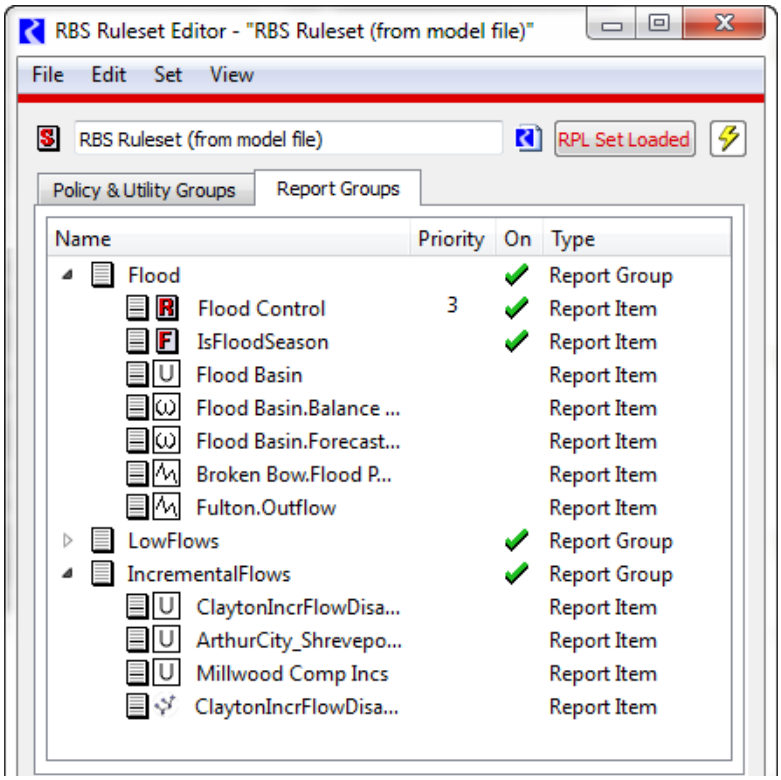


Adding Groups: Groups (of type Policy, Utility, Global Utility, or Report) are added to a RPL set by selecting the appropriate item from the RPL set's **Set** menu (e.g., **Set ➥ Add Policy Group**, **Set ➥ Add Utility Group**, **Set ➥ Add Report Group,** or **Set ➥ Add Global Utility Group,** for global function sets). New policy and utility groups are added to a set directly below the currently highlighted group of that type.

Add Blocks and Functions: Blocks and functions are added to a RPL set by highlighting the group they are to be placed in and selecting **Set ➥ Add Block**, **Set ➥ Add User-Defined Function,** or **Set ➥**

Add Global User-Defined Function (for global function sets) from the RPL set's menu bar. New blocks and functions are added to a RPL set directly below the currently highlighted element of that type or at the bottom of a highlighted policy or utility group.

Add Report Items: The **Report Groups** tab provides a more selective, user-defined view of the policy/slots and object. Items are added to a RPL report group (visible in the **Report Groups** tab) by highlighting the group they are to be placed in and selecting the desired action from the **Set ➥ Add Report Items** submenu, which contains an option to open a selector for each of the following types of objects:

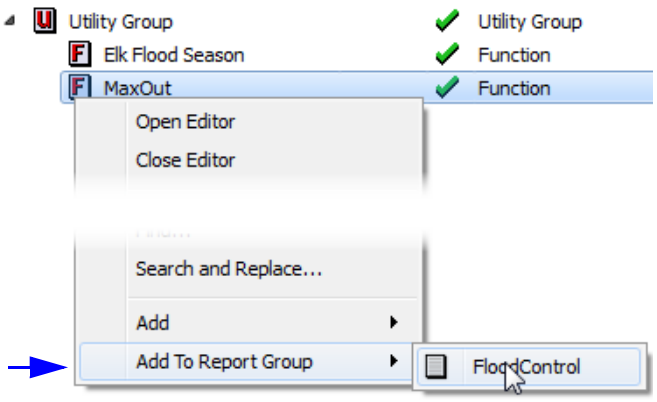


- **RPL Object Items...** Blocks and functions in the current set. Click HERE (Section 6) for a description of how to use the RPL Object selector.
- **Object Items...** objects on the workspace.
- **Slot Items...** slots on objects or accounts.
- **Subbasin Item...** a single subbasin (automatic or user-defined).

Clicking the **Ok** button in the selection dialog will append the items for the selected objects to the end of the selected report group, but can be moved to any position by using the drag/drop technique.

You can also right click on any RPL item (rule, function, etc), and choose Add to Report Group and select an existing Report Group. This will append the selected RPL item to the Report Group as shown in the screenshot to the right.

Click HERE (Output.pdf, Section 4.2.2) for more information on including a Report Group in a model report.



Change Priorities. Blocks and functions may be rearranged in a set. By changing the relative priority of blocks, the order of firing, success of slot assignments, and order of dispatching is altered. Changing of priorities is often used to evaluate the effectiveness of different operational policies. Rearranging

blocks and functions in a set is simple. To move a block in the list, click on its icon and drag the block to the new desired location within the set. If you place it on another block, it will append after that block. Then, simply release the mouse button to drop the block in its new location. A dialog will ask you to confirm that you wish to complete the move. Click OK to confirm the move. All other blocks will be shifted to comply with the new priority structure. There is no need to re-load a loaded set when rearranging blocks. Moving a block or a function is done in an identical manner. Predefined functions cannot be moved.

Delete Groups, Blocks and Functions: Groups, blocks and functions are deleted from a RPL set by highlighting them and selecting **Edit ➥ Delete**. Deleting a policy or utility group deletes the contents of the group as well. A confirmation dialog is shown.

Open and close Policy, Utility, and Report Groups. Policy and utility groups which contain at least one block or function can be opened in a separate window by double clicking their  ,  , or  icon or highlighting them and selecting **Set ➥ Open Editor...** They may also be expanded within the **RPL Set Editor** window by clicking on the white tree-view triangle to the left of their icon or selecting **View ➥ Expand**. An expanded policy or utility group can be collapsed by clicking on the white tree-view triangle again or selecting **View ➥ Collapse**.

Open and close Rules, Functions, Methods, Goals, and Report Items. Blocks and functions are opened in a RPL Viewer by double-clicking the icon or highlighting the item and selecting **Set ➥ Open Editor**. An opened block or function in the RPL Viewer may be closed by clicking on red X on the tab or by selecting **File ➥ Close Window** from an individual RPL Editor.

Copy and Paste Groups, Blocks, and Functions: Groups, blocks and functions may be copied and pasted within a set or between open sets. An item is copied into memory by highlighting it and selecting **Edit ➥ Copy**. An item in memory is pasted just below any highlighted item in a RPL set by selecting **Edit ➥ Append**. An item may also be copied and pasted just below the item by selecting **Edit ➥ Duplicate**. Copy/Paste/Duplicate is only available within a single model; you cannot copy/paste items from RPL sets that are opened in separate instances of RiverWare. Click **HERE (Section 1.8)** for more information on Import/Export to perform this task.

> **Note:** A copied group/block/function can be pasted (or dragged and dropped) as a text representation into any text editor such as a word processor (Word), text editor (notepad), or email program (thunderbird). This includes all information about that group, block, function including the logic, description, active state, etc...

Exporting and Importing Groups and Block: In addition to copy and paste operations, it is also possible to export and import all or part of a RPL set. This is useful to share user-defined functions between two RPL sets. Using the export and import utility, you can share groups, blocks, and functions amongst these sets. The export and import dialogs are described **HERE (Section 1.8)**.

1.6.2 Validity

Before loading a newly built set into a model (or running with one of the other types of sets), the validity of the set must be checked. This can be done manually or is automatically done at run start. Manually validating a RPL set, however, allows you to perform these checks without having to bring up the Run Control dialog, start the run, and stop the run or wait for it to finish. Validating a RPL set checks for unspecified expressions, illegal object and slot names, conflicting expression types, and syntax errors. It *does not* check the consistency of unit types in mathematical expressions. Unit consistency is done while the block is evaluated during the run.

- To check the entire set, in the RPL Set Editor, select **Set ➥ Check Validity.**
- To check a single block or user defined function, select **Block ➥ Check Validity.**

If the block or set is valid, a confirmation window appears. Click **OK** and continue. Otherwise, a validation error dialog appears and the diagnostic window posts the location of the invalid expression. You then must fix the block.

1.7 Comparing RPL Sets

When developing RPL sets, it is often desirable and/or necessary to compare two RPL sets and see what is different between the sets. Following are some examples of when you might want to compare two RPL sets:

- You have multiple copies of similar RPL sets that represent different policy alternatives. You need to determine what is different to make sure they represent the correct policy alternative.
- Your co-worker has made changes to the RPL set and you want to know what changes have been made. Perhaps then you will incorporate the changes into a master version.
- You need to identify all of the changes that have been made to the set since some previous version.

The **RPL Set Comparison Tool** compares two RPL sets and shows you the differences between the sets. This allows you to see where items are different, what the specific differences are, and allows you to easily access the RPL set dialogs so that you can change one or both sets. Although this tool doesn't support automatic merging, it provides quick access to the RPL dialogs and the ability to copy and paste from the utility to the RPL editors.

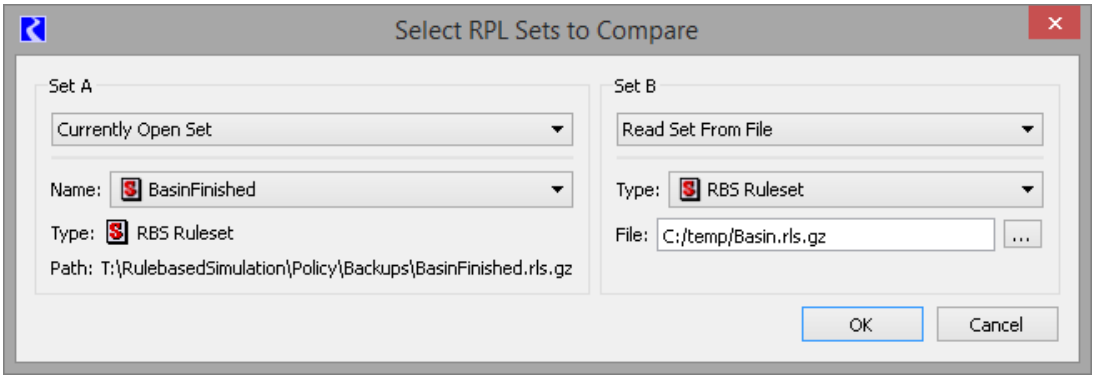
1.7.1 Accessing the Comparison Tool

From the RiverWare workspace, use the menu: **Policy ➥ RPL Set Comparison Tool...**

Or from any RPL set editor dialog select **Set ➥ Compare Set...**

1.7.2 Selecting RPL Sets to Compare

The tool opens and shows a dialog where you specify the two sets to compare. The two sets are referred to as **Set A** and **Set B**. For both sets, choose from either:



- **Currently Open Set**: Choose any of the sets that are open in the current model.
- **Read Set from File**: Choose a RPL Set on the file system.
- **Read Set from Model**: Choose a RPL Set that is embedded in another model file on the file system.

Next, choose the set.

For the **Currently Open Set** option, select the name of the RPL set. For the **Read Set from File** option, choose the type and file path. Use the ellipsis button to open a file chooser. For the **Read Set from Model** option, first select the model file. Then select the name of the set.

> **Note:** If reading the set from another model file that was last saved with a RiverWare version prior to 7.0, you will get a notification that you must specify the type of the RPL set.

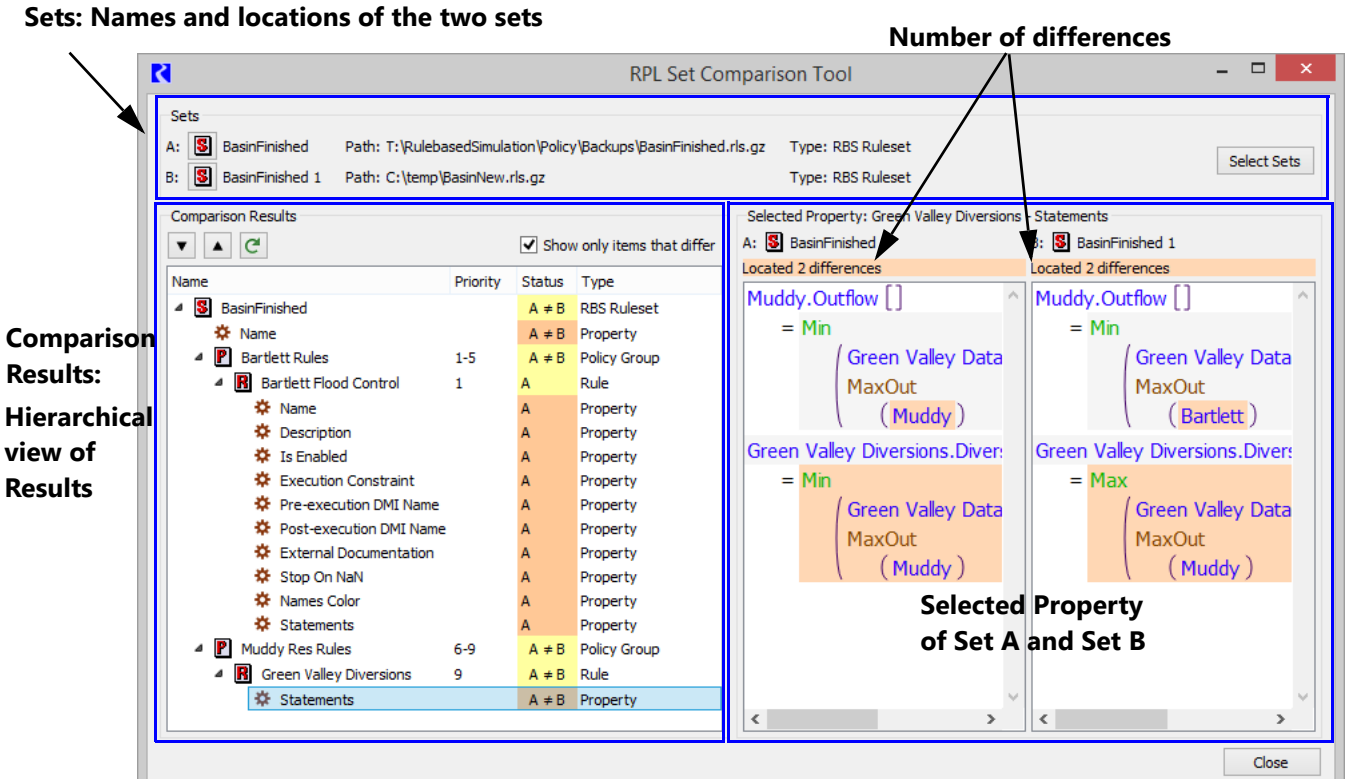
When finished, click **OK**. The main **RPL Set Comparison Tool** opens.

1.7.3 Tour of the RPL Set Comparison Tool

The RPL Set Comparison Tool contains three main areas as shown in the screenshot:

- The **Sets** panel shows the names, locations and type of the selected Sets.
- The **Comparison Results** area shows a hierarchical view of the sets and the comparison results. This shows where there are differences between the two sets
- The **Selected Property** panel shows the values for the selected row for both Set A and Set B. This shows the differences for the selected item.

The following sections describe these areas and how to use the tool.

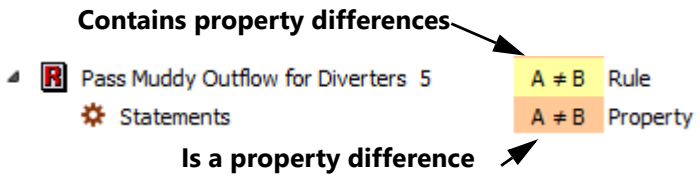


Sets panel: The Sets panel shows the sets that are being compared. It shows the set Name, Location, and type. Use the **Select Sets** button to choose alternative sets. This opens the dialog shown **HERE (Section 1.7.2)**.

Comparison Results: The **Comparison Results** panel shows a hierarchical view of the results of the comparison. The hierarchy is based on the structures of the two set. Columns include the **Name**, **Priority/Index**, **Status** and **Type**. The Set, Groups, and Rules/Goals/Methods show their particular icon. The **Status** column shows the results of the comparison. The following table gives the possible status and the meanings:

Status	Meaning
A = B	The item is identical in Sets A and B.
A ≠ B	The item is different in Sets A and B.
A	The item exists in set A but not in Set B.
B	The item exists in Set B but not in Set A.
A ->	The item exists here in Set A and elsewhere in Set B.
<- B	The item exists here is Set B and elsewhere in Set A.

The **Type** column lists the type of that row, either a Set, Group, Rule, or Property. The comparison tool ultimately compares Properties of the items, not the items themselves. In the **Status** column, a lighter yellow shading indicates that the item **contains** properties and that within the item there are differences in properties. The darker orange color indicates that the value of the property is different.



The Comparison Results panel has buttons on the top for navigation and control of what is shown:

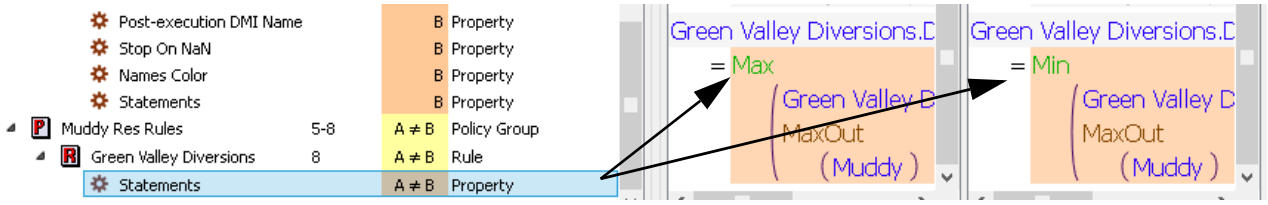
Button	Text	Meaning
▼	Next Difference	Select the next property that has a difference
▲	Previous Difference	Select the previous property that has a difference
↻	Recompute Differences	Recompute the differences between the two sets. This is useful if you make changes to Set A or B
☑ Show only items that differ	Show only items that differ	Show only items that are different in the results panel. This is the default behavior.

Double click on a row in the **Comparison Results** to open the RPL dialogs for that item. Note that if the set was loaded from a RPL set file or another model file, the dialogs cannot be opened. If both sets are opened in the model file, then double clicking a row will open two dialogs, one for each set.

Note: The **RPL Set Comparison Tool** compares internal copies of the specified sets. If you make changes to one of the chosen sets, it is not immediately reflected in the comparison tool. Use the Recompute Differences button ↻ to update the status in the tool.

Selected Property: The **Selected Property** panel shows the values for the selected row. If a yellow row (Rule, Set, Group, etc...) is selected, a note is shown that "No Property Selected". Only properties are shown in these views. When an orange **Property** row is selected, the panel shows the values for that property. The information shown is based on the property type and can include text, colors, true/false, and RPL expressions. When a statement, function body, or execution constraint is selected, the RPL expression is shown for both sets A and B. For example in the following screenshot, a statement is selected and the RPL statements are shown. You can see that Set A has a Max function while Set B has

a Min function. The total number of differences found is listed above each Selected Property panel. Multiple differences are highlighted whenever possible.



Following are known limitations related to the highlighting of differences:

- When two statements or expression differ only by comment, the tool highlights the entire statement/ expression, not just the comment. Ideally the tool would highlight the individual differences in the comment text.
- The tool does not highlight differences in symbol declarations (E.g. NUMERIC num), either in the type or name, but the entire expression is highlighted
- Adding, deleting, or moving a statement within a block (or an item in a list expression) will usually lead the tool to overestimate the number of changes (i.e., every statement after the first difference will be considered different). This is not the case for insertions/deletions within textual properties.

1.7.4 Example of using the RPL Set Comparison Tool

Following is one use case example where the **RPL Set Comparison Tool** is very useful.

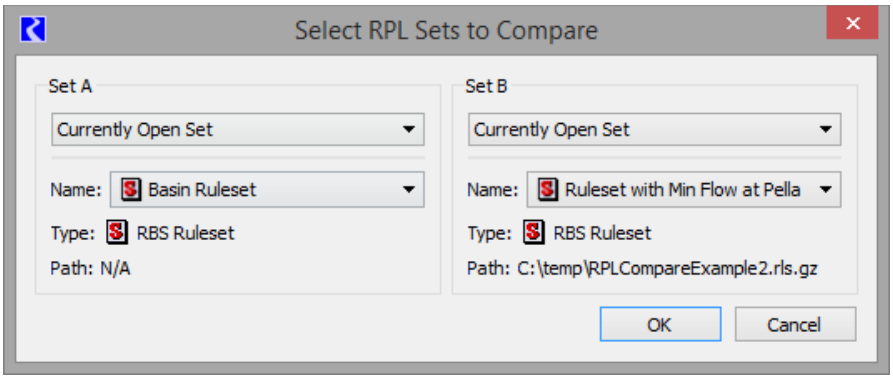
We have been developing a model and ruleset of our basin and need to add some proposed policy changes to the set. I am busy with other things, so I asked my co-worker to help implement the policy in the ruleset. She did this and sent me the new ruleset on her way out the door for vacation. She left the note that “I added the new rule and had to update a couple other things” but didn’t describe what she changed. I also have been working on the model and now it is time to see what she did and merge it into my work. Following are the steps I’ll take:

1. I open my model and load my ruleset. In this case, my ruleset is embedded in the model file; it is called **Basin Ruleset**.

2. I open (but don’t load) my co-worker’s modified ruleset. This is called **Ruleset with Min Flow at Pella**.

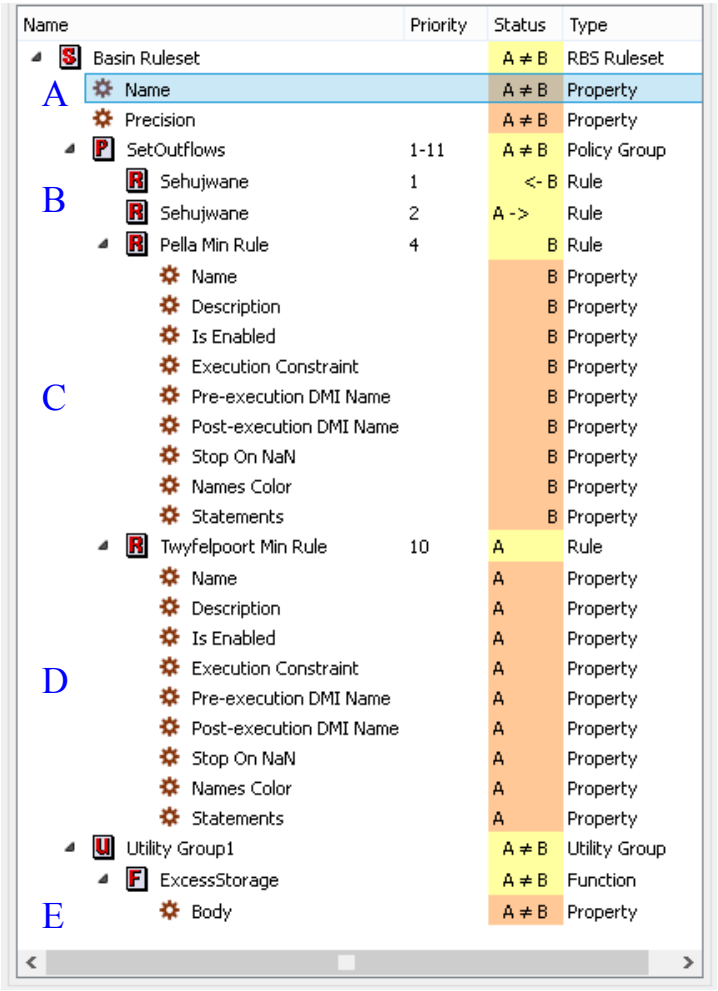
3. I use the menu: **Policy ➥ RPL Set Comparison Tool...**

4. I then choose my embedded ruleset, **Basin Ruleset**, as Set A and my coworker's modified ruleset as Set B. This is shown in the following screenshot. I then click **OK**.

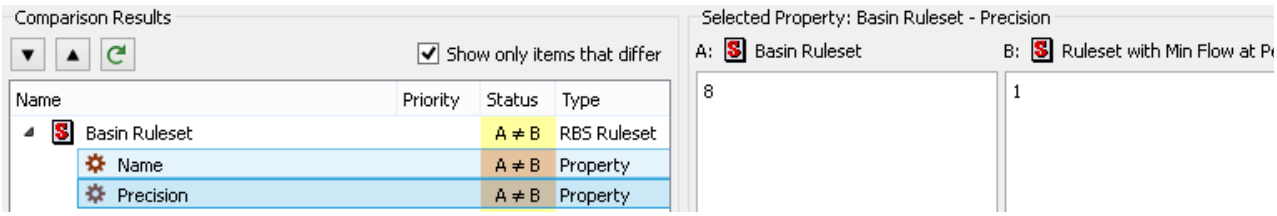


5. The RPL Set Comparison Tool opens and shows that there are differences between the two sets. I can use the Next/Previous to step through the differences. I can also uncheck the **Show only items that differ** to get a sense of where in the set the differences occur. I usually only like to see the differences, so I re-check the **Show only items that differ.** Following is a screenshot of the results.

6. It looks like there are five areas of differences. We'll go through all of the differences in the following table. I click on each row to see the property difference in the right hand panel.

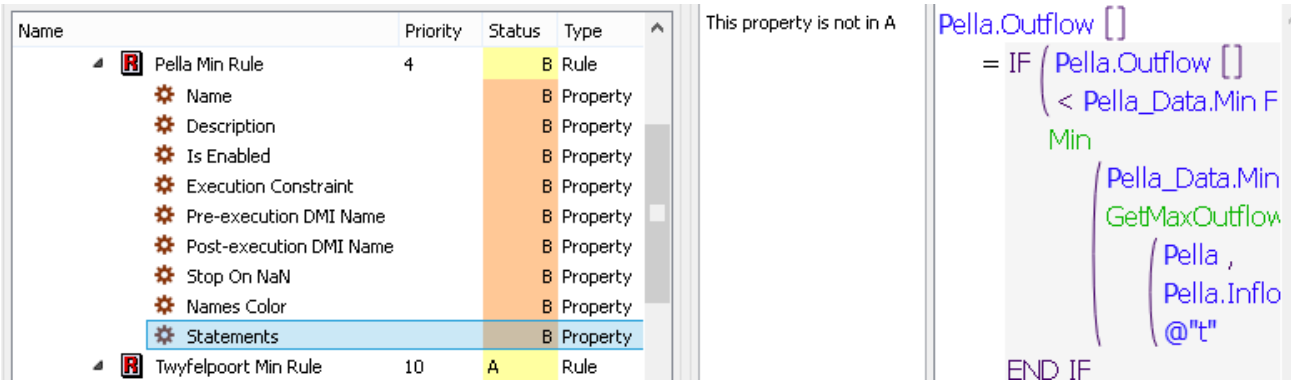


Difference	Description
A	The ruleset name and precision are different.

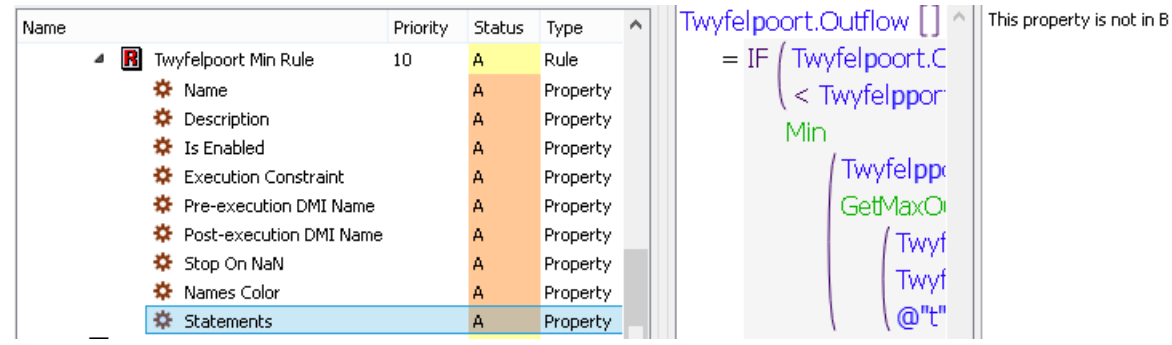


B	Rule 1 and 2 have changed priority.In Set B, Sehujwane is priority 2. In Set A, Sehujwane is priority 2. Since it is strictly the priority order that has changed, there are no property differences, so the right panel doesn't show any differences.

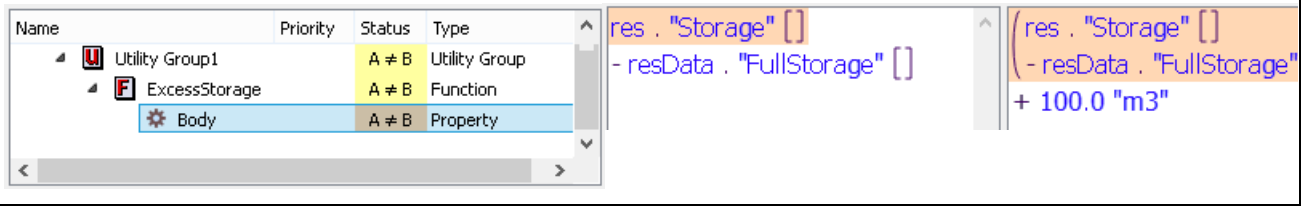
C.	Rule 4 is in Set B but not in Set A. The right panel shows the logic in Set B and lists that the property is not in Set A.



D.	Rule 10 is in Set A but not in Set B.



Difference	Description
E.	The **ExcessStorage** function has differences. It looks like my co-worker added a 100 m3 tolerance for the comparison. It would be nice if she had documented why she did this.

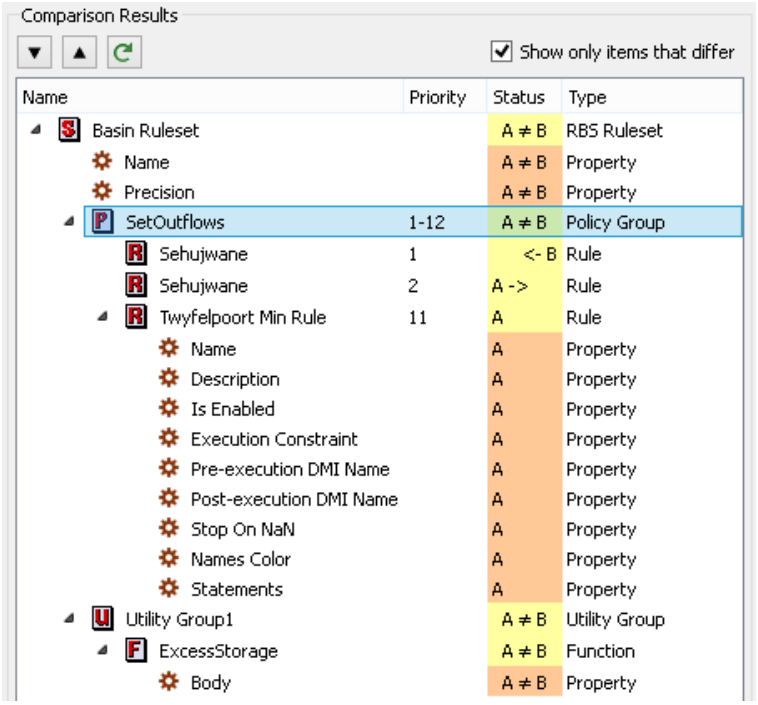


7. Now I need to combine or merge her differences into my ruleset. First I'll click on the SetOutflows row to open the policy group for both sets.

8. The priority order of the first two rules is correct in my set, so I'll leave it as is, with no changes.

9. I need to get the new rule that my coworker developed. Since the desired rule is in Set B, I open that set and right click the **Pella Min Rule** and Copy. Then append it to that same group in Set A. I'll re-arrange to make it rule 4.

10. The RPL Set Comparison tool hasn't changed even though I am editing the set. The comparison tool uses copies of both sets so the originals aren't affected. I must click the Recompute button  to get the changes. Now I see that both sets have 12 rules in the SetOutflows group.



11. Now I'll update the function that was changed. I double click on the ExcessStorage function Body row to open both function editors.

12. I copy the entire expression from Set B and paste it over the expression from Set A. Now they are the same.

In this way, I've looked at the differences and decided which ones I want to apply to my set. Although I had to manually merge the two sets, the **RPL Set Comparison Tool** really helped me to identify where there are differences and then present those differences in an organized fashion.

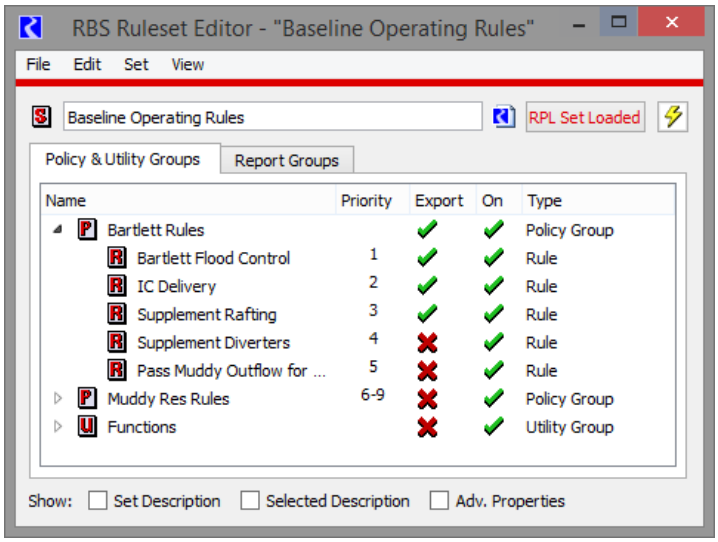
1.8 Exporting and Importing RPL sets

In this section we describe how to export all or a portion of a RPL set to a separate RPL set file, as well as how to import an existing RPL set file. The export and import mechanisms provide a way to incorporate all or some of one policy into another policy, and it is available for all application of RPL within RiverWare, including:

- Rule sets - collections of rules and functions organized into policy and utility groups.
- Optimization goal sets - collections of optimization goals and functions organized into policy and utility groups.
- The object-level accounting method set - a collection of methods and functions organized into predefined category groups and utility groups.
- The Expression Slot set - a collection of functions organized into utility groups. These functions may be called from an expression defining an expression slot.
- Initialization Rules - collections of rules and functions organized into policy and utility groups.
- Iterative MRM RPL sets - collections of rules and functions organized into policy and utility groups.
- Global Function Sets - collections of functions organized into Global Utility Groups.

There are many situations in which export and import functionality is useful. For example, assume that you are writing a policy which could make use of some functions written as part of an existing policy. To avoid having to rewrite all these functions, you could open both sets and use the Copy/Paste mechanism to copy the functions in question from the existing policy to the new one. However, if you then modify the original functions, you might want to do the copy again to keep the functions identical, but you would have to first remove the copies of the functions before doing this. Furthermore, in the context of the object-level accounting method set it is not possible to have two such sets open simultaneously, so this mechanism would not work. The RPL set export and import functionality provides a simpler and more flexible method to implement sharing of policy elements. Alternatively, you could use Global Utility Groups described **HERE (Section 5)**.

1.8.1 Export



The RPL set export mechanism allows you to save a portion of a RPL set as a separate RPL set file. The first step is to show the Export column of the RPL set editor. If this column is not currently shown (between the columns labeled “Priority” and “On”), then it can be shown by selecting **View ➥ Show Export Column**. Initially no item is selected for export, this is indicated by a red “X” in the Export column for each item.

Click in the Export column of the items that you would like to export. This will convert the red “X” in that cell to a green check mark. Note that selecting a block or function for export automatically selects the containing group, and unselecting a group will unselect all of its members.

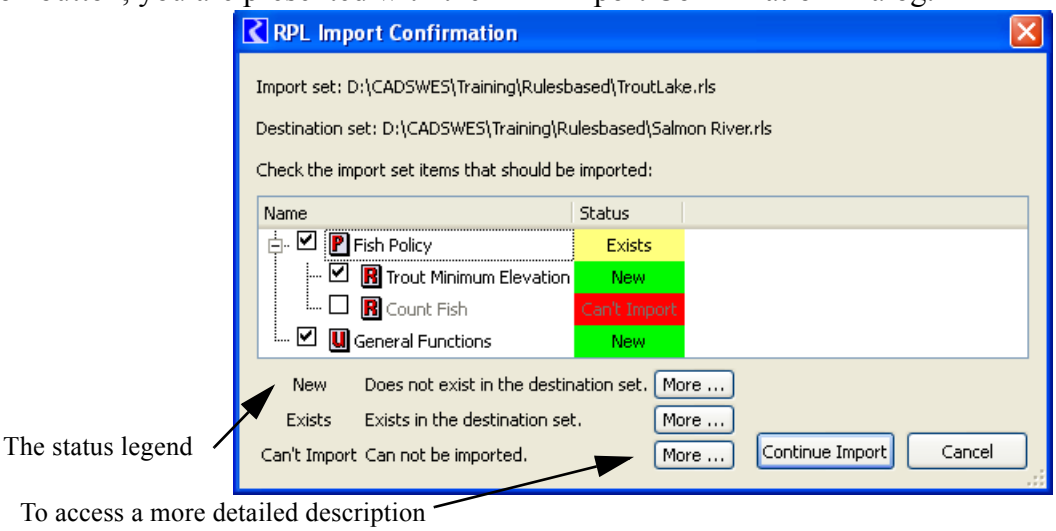
To select multiple items, right-click and choose **Select Export For All in Group** or **Select Export For All in Set.**

Once the items to be exported have been identified in this way, the export operation is initiated by selecting **File ➥ Export Selected Items.** You will then be presented with a confirmation dialog summarizing the items to be exported If you choose to continue the export operation, a file chooser dialog appears, allowing you to specify the file to which the items will be exported. Once a file name has been selected, the file is created and a RPL set containing only the selected export items is written to that file.

1.8.2 Import

RPL set import allows you to incorporate all or some aspects of an existing RPL set policy into a set that you are editing. To initiate an import, select **File ➥ Import Set**. You will be presented with a file

chooser dialog for specifying the name of a RPL set file. Once you have selected a file and clicked the **Open** button, you are presented with the RPL Import Confirmation Dialog.



This dialog presents a listing of the import set. This layout is similar to the RPL set editor's presentation of the set, but each item being imported is classified as having one of the following statuses:

- **New** - The group, block, or function being imported does not appear to exist in the set into which it is being imported (the destination set).
- **Existing** - The group, block, or function being imported appears to exist in the set into which it is being imported (the destination set). In the case of a group, there is a group of the same type with the same name in the destination set. In the case of a block (or object-level accounting method), there is a block with the same name in a group which is of the same type and has the same name as the block's parent group. Similarly, in the case of a function, there is a function with the same name in a group which is of the same type and has the same name as the function's parent group.
- **Can't Import** - The group, block, or function begin imported can not be imported. There are several reason why this might be the case and a warning is posted to the Diagnostic Output Dialog for each item that can not be imported, explaining the problem. Possibilities include an attempt to import a TCL function (not supported) or an attempt to import a function into a policy group when there is already an item of the same name in the global name space, e.g., a function of the same name exists in a utility group.

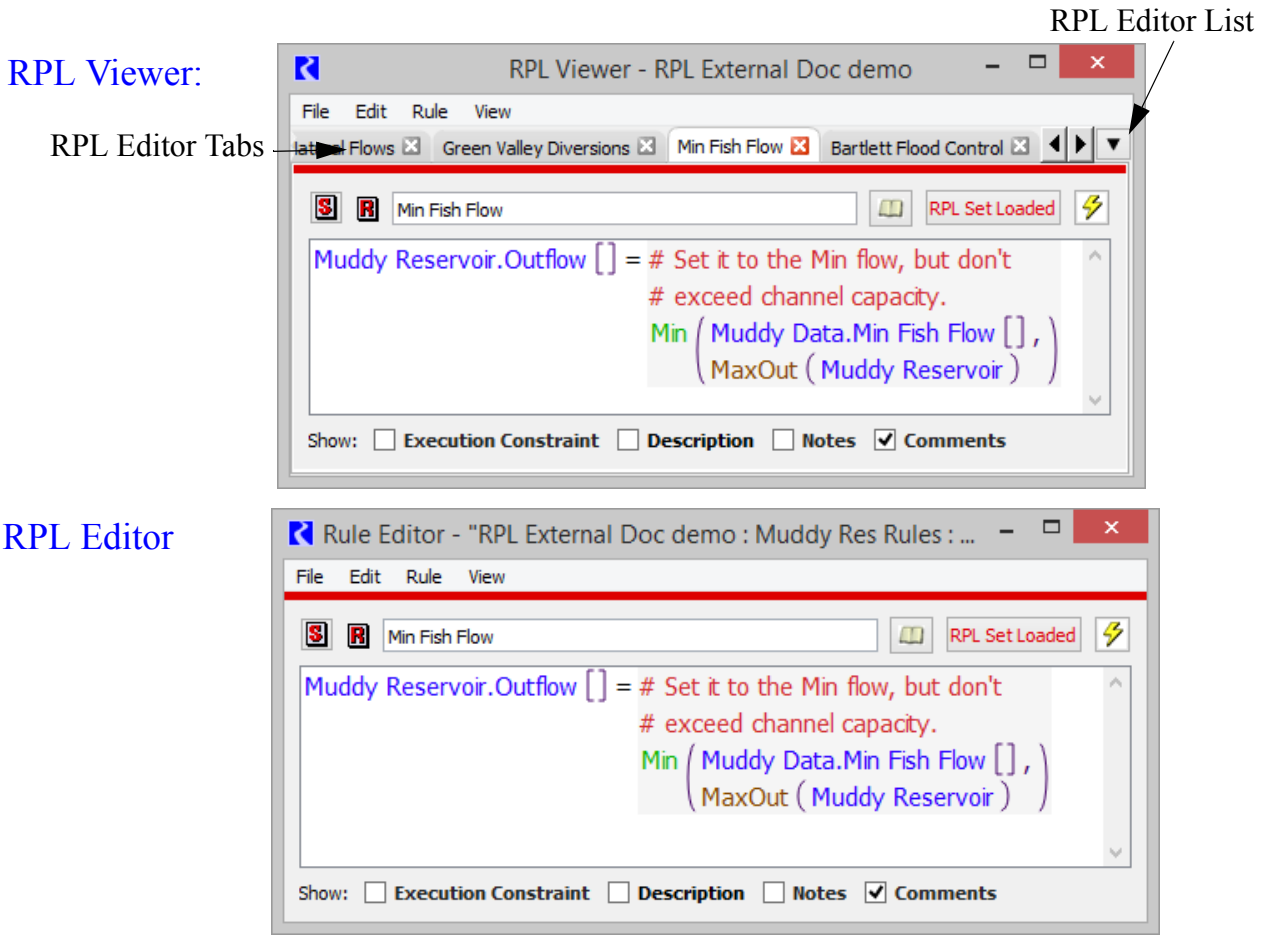
Before continuing with the import operation you should review the items being imported and indicate which items should in fact be imported. To select an item for import click in the box to the left of its name in the list. Importing a new group will append it to the list of groups; importing a new function or block into an existing group will append it to the groups of functions which already exist. Importing an already existing group will copy its description to the existing group and copying an existing block or function will replace the existing block or function with the imported block or function. Initially new items are selected for import and existing items are not, but these selections can be modified. To complete the import click on the Continue Import button. Clicking on the Cancel button at any point will cancel the import operation and leave the RPL set unchanged.

2. RPL Editor Dialogs

This section describes the editor dialogs used to build RPL Rules, Functions, Goals, and other expressions. There are two ways to view the RPL editor dialogs, either in the **RPL Viewer** or in the single **RPL Editor**.

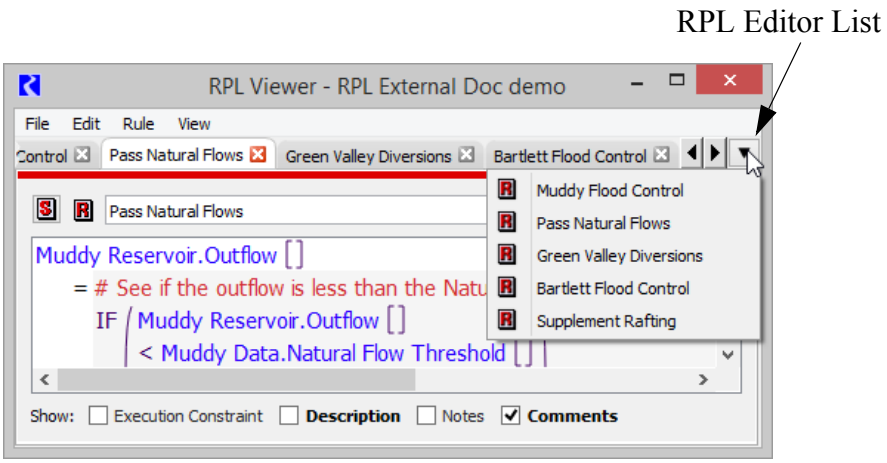
2.1 RPL Viewer vs RPL Editor

When you open a function, rule, method, goal or any other block, the item will open as a new tab in the **RPL Viewer**. Opening subsequent RPL Items will add tabs to this **RPL Viewer**. This allows you to have one dialog where all RPL editors are shown. But, you can undock any item into its own **RPL Editor**. the following screenshot shows the **RPL Viewer** and the **RPL Editor** with the same rule shown. Notice the tabs in the viewer.



Note that the menu bar options will be the same. They are dynamically updated in the viewer based on the tab selected.

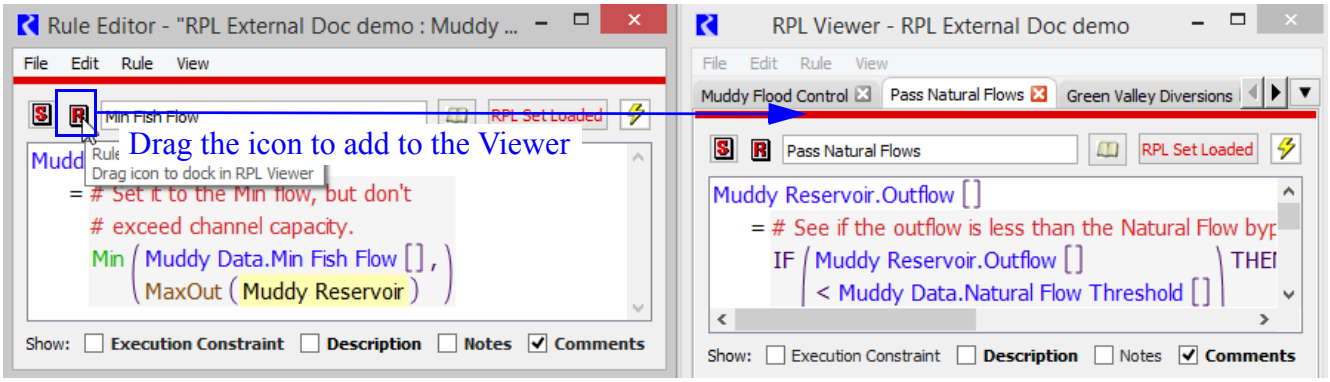
The **RPL Viewer** is the default dialog used to look at an RPL editor. It is convenient as all RPL editors open in the one dialog. Click on the tabs to switch between items. Or use the **RPL Editor List** menu on the right of the tabs. This is particularly useful if you have many items in the viewer. Tool tips on the tabs give the full name and priority/index where relevant.



Arranging tabs: Drag the tabs left and right to re-arrange. Use the left and right arrows to scroll to tabs that are not shown.

Undocking RPL Editors: To look at two editors at once, drag the tab off of the viewer to create a RPL Editor dialog for that item. Or use the **File ➥ Undock...** menu.

Redocking RPL Editors: To move a RPL Editor dialog back onto the viewer, grab the R, F, G, Icon and drag it anywhere onto the viewer. A new tab will be created. Or use the **File ➥ Dock In RPL Viewer** menu. Also, from the viewer, you can use the **File ➥ Redock All** to move all RPL Editors to the viewer.

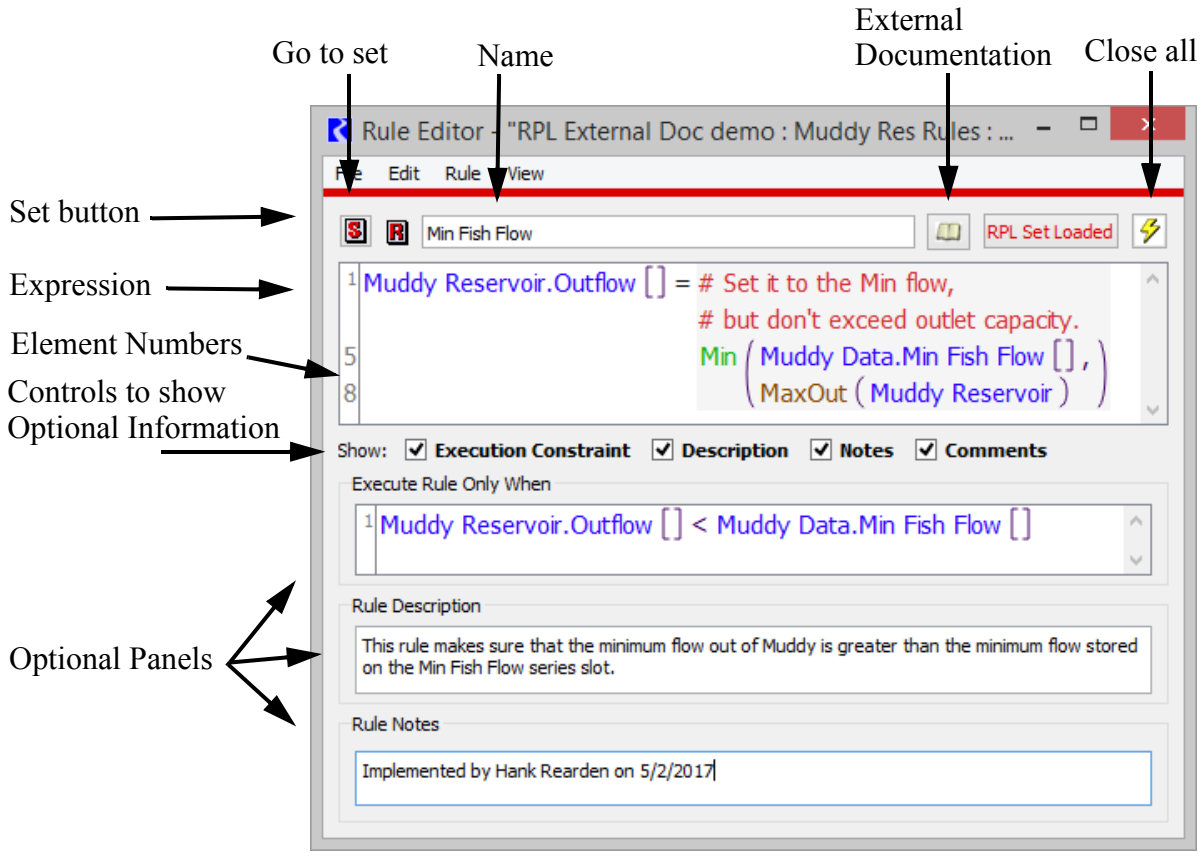


Closing a RPL Editor tab: Use the red X on the tab to close a tab that is on the viewer.

Note: The RPL Viewer is transient; once it is closed, the configuration of items shown and their order is lost.

2.2 Working with RPL Dialogs

Whether you are looking at the editor or the viewer, the RPL dialogs consist of menus, name and buttons, the expression, and optional additional panels as shown in the following screenshot. These items are described in this section.



- **Type of Set:** The icon indicates the type of set to which this editor belongs.
- **Name:** The name field is where you specify the name.
- **External Documentation:** The button, when configured, opens the external documentation as described **HERE (Section 7)**.
- **Close All**: The button invokes a dialog in which you can close all open RPL dialogs except the top level editor.
- **Set Button**: Icon buttons allow you to show the containing set for each RPL dialog.
- **Expression:** This area contains the expression.
- **Element Numbers**: This column shows the first element shown on that line. For more information, click **HERE (Section 6.2.4)**.

Button to take you to the Set

- **Optional Information:** Optional information can be shown for various types of RPL dialogs. These are described in the following sections. If there is an entry in these panels (or a non-default value) a box is drawn around the toggle.

2.2.1 Execution Constraint / Execute Block Only When

A block can be set up to execute only when a given condition is true. This functionality can be used to increase performance or it may be necessary to implement your RPL set correctly.

- From a RPL editor, select **View ➥ Show Execution Constraints** or click the **Execution Constraint** toggle.

An area is added to the bottom of the window. By default, the block executes only when **TRUE**. This means that the block will always execute when it comes up on the agenda. You may want this block to execute only when some other condition is true. The **TRUE** expression can be replaced with any logical structure that can be constructed from the palette

> **Note:** If you enter a boolean expression to replace the default TRUE, then clear this expression, an unspecified expression will remain. This unspecified expression will invalidate the rule. To prevent this, if you clear out this boolean, make sure you re-type TRUE to get the default behavior.

2.2.2 Descriptions

A special expanded area of the RPL dialog is used to enter a description of the block. The area is opened and closed by toggling the **View ➥ Show Description** in the menu bar or clicking the **Description** toggle at the bottom of the window. Note, if there is any description entered, the **Description** toggle will have bold text. Mouse over the word Description to see up to 140 characters of the description as a tooltip. Descriptions can be optionally included in Model Reports.

Description can be included in Model Report RPL items (RPL Group, RPL Rule/Goal, RPL Set) with the **Show Descriptions** settings described **HERE (Output.pdf, Section 4.2.3.14)**.

2.2.3 Notes

Notes, like Descriptions, can be entered in a panel at the bottom of the dialog. Notes can be used when you have information that you want to enter that doesn't belong in the **Description** field. For example, development notes or change logs could be entered in the Notes panel.

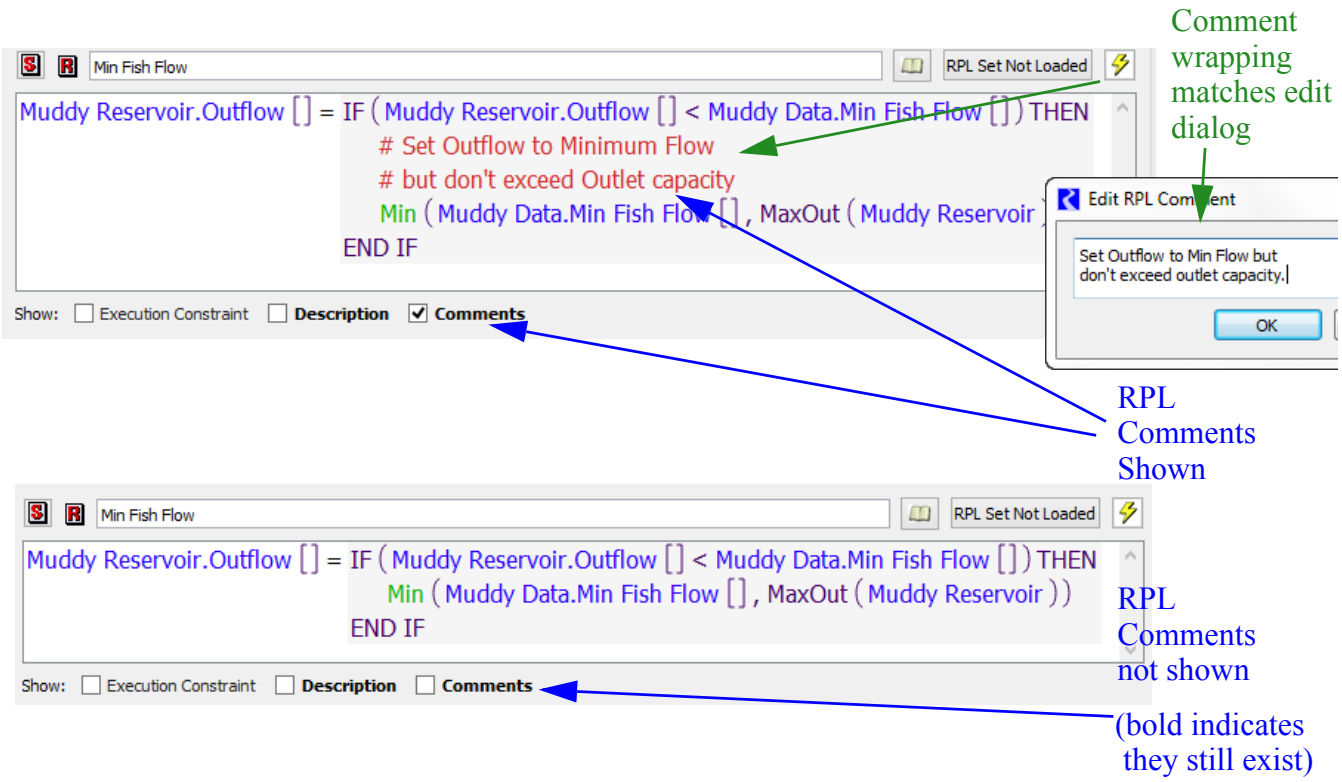
The area is opened and closed by toggling **View ➥ Show Notes** in the menu bar or clicking the **Notes** toggle at the bottom of the window. Note, if there is any description entered, the **Notes** toggle will have bold text. Mouse over the word Notes to see up to 140 characters of the note as a tool tip.

Notes can be included in Model Report RPL items (RPL Group, RPL Rule/Goal, RPL Set) with the **Show Notes** settings described **HERE (Output.pdf, Section 4.2.3.14)**.

2.2.4 Comments

Inline Comment are added from the palette using the **Add Comment** button. A separate dialog is opened that allows you to type in a comment. In the RPL editor, the comment is displayed with # characters on the left, lines are wrapped as they were in the comment editor dialog. Double clicking the comment reopens the edit dialog.

Inline RPL comments can be optionally shown/hidden using the **View ➥ Show Comments** menu or the **Comments** toggle. Note, if there are any comments defined, the **Comments** toggle will have bold text. Also, **Comments** are shown by default, if you do not want to see them, you must explicitly hide them.



2.2.5 Executing DMIs from Blocks

RPL blocks can be used to execute DMIs or DMI groups. DMIs are described **HERE (DMI.pdf, Section 2)**.

To add a DMI, Select **Rule/Method ➥ Add Pre-execution DMI** or **Rule/Method ➥ Add Post-execution DMI**. Then, select the name of a DMI or DMI group that is defined in the model.

To remove a DMI, use the **Rule/Method ➥ Remove Pre/Post-execution DMI** menu.

A pre-execution DMI is executed as the first step of the execution, a post-execution DMI is executed as the last step of execution. Values imported by the pre-execution DMI are available for use by the statements. The order of execution for a rule is:

- If the Execution Constraint evaluates to TRUE, continue. Otherwise terminate the rule early.
- Run the Pre-execution DMI.
- Evaluate the body of the rule.
- If the rule evaluation is successful, run the Post-execution DMI. If the rule is empty, terminates early, or is ineffective, the Post-execution DMI is not run.

To prevent unexpected solving of objects and/or conflicts between values, input DMIs executed from RPL blocks can only import values to **unlinked series slots on data objects**. Furthermore, values imported from a DMI executed from a RPL block are given the following flags:

- Control File-Executable DMIs: imported values are given the INPUT (I) flag (priority zero).
- Database DMIs: imported values are given the OUTPUT (O) flag and the priority of the rule executing the DMI.

An example application of this feature is executing an external water quality model at each timestep. In this example, the water quality model might take as input RiverWare values which reflect the current state of the system and return a recommended reservoir release value. At each timestep the block would first execute an output DMI to export current values from relevant RiverWare slot(s), e.g., inflow on a reach. An input DMI then runs the external water quality model (which reads the data from the output DMI) and then imports the resulting recommended reservoir release values.

Note: You can turn on Diagnostics to see when rules and Rule DMIs execute. Use the **Rule Execution** and **DMI ➥ Rule DMI** diagnostic categories as described **HERE (Diagnostics.pdf, Section 3.1)**.

2.2.6 Stop On NaN

Typically, when a RPL block accesses a slot but the slot is NaN, the block terminates early but the run continues. Under certain conditions, you may wish to specify that a block does not terminate early but instead stops the run when a NaN is encountered. This could be useful for data checking rules; if the referenced slot value has not be specified, then you wish to stop the run and fix the data error.

To specify that a block should “Stop On NaN”, use the **Block ➥ Stop On NaN** menu. When enabled, a check mark is added next to the menu. Any time a slot expression accesses a NaN, the run will stop and a diagnostic will be posted that explains which block referenced a NaN.

Note that in the context of initialization rules the run is not stopped immediately when an invalid value is encountered, but rather it is stopped only after all of the rules have executed. This allows RiverWare to report on all missing data with a single run.

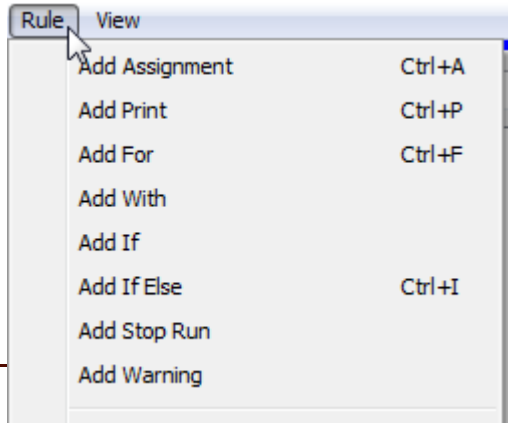
2.3 Statements

As described **HERE (Section 1.1)**, blocks are the upper level construct of a RPL set; for example, rules and accounting methods are blocks. Blocks are made up of statements which are different structures depending on the desired result.

- In the RPL set editor, open a block by double clicking or right clicking on its icon. This will bring up the **<block> Editor** dialog**.** Rename the block by typing in a new name in the Name text field.

Block are constructed from basic statements and expressions. The following basic statements are available in the **Set** menu in RBS rulesets, initialization rules, MRM rulesets, and accounting method sets**.** They are:

1. Assignment: An assignment statement assigns to the slot on the left-hand side (LHS) of the equality the numeric value evaluated on the right-hand side (RHS) of the equality. The basic assignment is:



<numeric expr> = <numeric expr>

where the LHS <numeric expr> is a slot and the RHS <numeric expr> is any expression which evaluates to a value in the unit type of the LHS slot.

Note: RBS rules, MRM rules, and accounting methods can assign values on **series slots only**. Initialization Rules described **HERE (Simulation.pdf, Section 5.1.2)** can assign values on series slots, table slots, and scalar slots.

2. Print: A print statement evaluates its expression and formats the result into a message. The blue message is displayed in the **Diagnostics Output** window when the **Print Statements** diagnostics group is enabled. The basic print statement is:

Print <expr>

where the **<expr>** is any expression or concatenated expressions which can be fully evaluated and represented as a string. For more information on the Print statement click **HERE (Diagnostics.pdf, Section 3.3)**

3. For: Iterative loops can be very useful for computations and multiple assignments. For loops are available to make multiple slot assignments from similar logic and calculations. An index variable is assigned a new value for each iteration of the loop. The inside of the loop is one or more regular assignment statements which should use the index variable to perform a different assignments for each iteration of the loop. This variable may be used in both the LHS slot assignment and the RHS evaluation to affect slightly different behavior within each pass. The default for loop is:

```
FOR (NUMERIC index IN <list expr>) DO
    <numeric expr> = <numeric expr>
END FOR
```

where the number of loops/assignments is determined by the number of elements in the **<list expr>**, the **NUMERIC** label indicates the expression data type of the elements in the **<list expr>**, and the **index** is the variable name which will take on the value of each element for use inside the loop. All of these parts of the for statement may be modified.

Note, there is also a For expression on the palette as described **HERE (RPLTypesPalette.pdf, Section 2.6)**.

4. **With:** Evaluate the expression and set the result to a local variable with the given name and type. then evaluates the contained statements, which may reference the variable. The default WITH syntax is

```
WITH (NUMERIC val = <numeric expr>) DO
    <numeric expr> = <numeric expr>
END WITH
```

Note, there is also a WITH expression on the palette as described **HERE (RPLTypesPalette.pdf, Section 2.6)**.

5. **If**: Make a statement conditional on a boolean expression without an ELSE.

```
IF(<boolean expr>) THEN
    <statement>
END IF
```

Note, there is also an IF expression on the palette as described **HERE (RPLTypesPalette.pdf, Section 2.6)**.

6. **If Else**: Make a statement conditional on a boolean expression with an ELSE.

```
IF(<boolean expr>) THEN
    <statement
ELSE
    <statement>
END IF
```

Note, there is also a IF ELSE expression on the palette as described **HERE (RPLTypesPalette.pdf, Section 2.6)**.

7. **Else If branch**: Although not a statement by itself, you can add one or more ELSE IF branches to an IF or IF ELSE statement. The ELSE IF is available when the boolean condition or a consequence statement is selected (except when an ELSE consequence statement is selected). An ELSE IF branch is added after the selected branch.

```
IF(<boolean expr>) THEN
    <statement>
ELSE IF(<boolean expr>) THEN
    <statement>
END IF
```

Note, there is also a ELSE IF expression on the palette as described **HERE (RPLTypesPalette.pdf, Section 2.6)**.

8. **Else branch**: Although not a statement by itself, you can add a single ELSE branch to an IF Statement or ELSE IF statement. This part of the statement will be evaluated when none of the other IF or ELSE IF boolean conditions are true.

```
IF(<boolean expr>) THEN
    <statement
```

ELSE

<statement>

END IF

Note, there is also a ELSE expression on the palette as described **HERE (RPLTypesPalette.pdf, Section 2.6)**.

9. Stop Run: Abort the run and post the provided expression as a diagnostic error message. When executed from within an iterative MRM rule, this statement aborts the MRM run.

STOP RUN <expr>

10. Warning: Post a brown warning to the diagnostic output window with a brief message followed by the value of the expression. This does not stop the run and is shown regardless of diagnostics settings.

WARNING <expr>

11. Execute Script: In Iterative MRM Rules only, execute the named script. Note, this statement should be used with **extreme caution** as scripts can modify many parts of your model.

EXECUTE SCRIPT <string expr>

2.4 Editing a RPL Expression

Expressions in the RPL editor are displayed as <expr>, <numeric expr>, <boolean expr>, <list expr>, <string expr>, <object expr>, <slot expr> or <datetime expr>. The type of the expression indicates the valid values that can go in that expression. Click **HERE (RPLTypesPalette.pdf, Section 1)** to see a description of each data type.

There are two ways to enter data into an unspecified expression, typing and using the palette. Any expression can by typed by first double-clicking then typing in the expression. There are many problems with this approach. First, you must know the exact syntax to be typed. Second, you must type all strings perfectly as typos in strings may not be caught on validation. A better approach is to use the palette to build expressions. You only need to type when they are specifying the inner most expression like exact numbers (100 cfs), boolean (true, false), datetimes (@“t”), or strings (“Entering Runoff season”). Following is a description of using the palette.

2.4.1 Using the palette

First, the palette buttons are described **HERE (RPLTypesPalette.pdf, Section 2)**. Each of the buttons in the RPL Palette represents an expression. These expressions are convenient operations which evaluate to one of the expression data types just mentioned. Buttons on the RPL Palette are enabled and disabled dynamically. When an expression is highlighted in the Editor, the Palette buttons that satisfy the expected data type are enabled. Thus, you create expressions in the following manner: click on an unspecified expression, click on a palette button or function. Click on another unspecified expression and click on a palette button or function. Repeat this process until the expression is created. Then, if necessary, double click on any remaining unspecified expressions that cannot be created from the palette (like entering numbers) and type in the value.

2.4.2 Entering Values

To replace an unspecified expression with a literal value (like 10 “cfs” or TRUE) double click on the expression and type in the value. For NUMERIC data types, you may also enter a unit. If the unit has a “-” or “/” in it, you must include the quotes when typing the unit, like “acre-feet”. Without these special characters, you can type the units and no quotes. Here are some examples of correctly specified units: `10 cms`, `100 "acre-ft"`, `50`. Illegal entries include `100 acre-ft/day` (no quotes) and `100 "cfx"` (no unit of that type). Additional information on entering units can be found **HERE (RPLTypesPalette.pdf, Section 1.1)**.

Another illegal entry is `1,000 "acre-ft"`. This is because the comma (a thousands separator) is not allowed as an input. Instead, RiverWare gives the option of whether or not to display a comma as a thousands separator for all values throughout the model and RPL set. This option is selected or deselected by clicking on the main **Workspace ➥ Show Commas in Numbers** option, and is described in more detail **HERE (Workspace.pdf, Section 5.8)**.

Note, after typing a literal value, you do not need to hit **Enter** to commit the change. Typing **Enter** or any mouse click outside of the editor will signal completion of the editing.

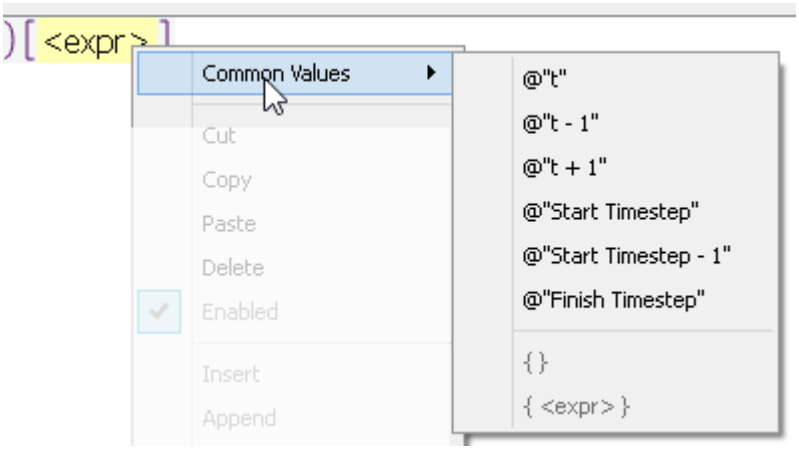
Auto-Correct: When typing in values, if the value you provide is not a valid replacement for the expression, RiverWare attempts to coerce the specified string into one that is valid. This auto-correction process is guided by the types which can legally replace the existing expression. RiverWare tries a series of variations on input, where each variation is an attempt to coerce the input into a different value type. Types are considered in the following order: DATETIME, OBJECT, SLOT, SRING, and LIST. If a valid auto-correction is found, it is used to replace the existing expression; if no valid auto-correction is found, an error notification is presented describing the problem with the input.

For example, consider the statement:

```
WITH (STRING val = <string expr>) DO
   PRINT "value: " CONCAT <expr>
END WITH
```

If the text `t + 1` is entered as the variable value in the WITH expression, it is interpreted as a STRING "t + 1" because that is the only legal type for that expression; whereas the same text entered as the right-hand side of the CONCAT expression is interpreted as the DATETIME @"t + 1", because all types are valid in that location and a DATETIME conversion is considered first.

Common Values : Common values are also available by right clicking on the selected expression and choosing the **Common Values** menu. These include common DATETIME values and list expressions.



- @“t” - Current Timestep
- @“t-1” - Previous Timestep
- @“t+1” - Next Timestep
- @“Start Timestep” - First timestep in the run period
- @“Start Timestep - 1” - Initial time-step
- @“Finish Timestep” - Final Timestep in the run period
- { } - Empty list expression
- { <expr> } - List containing one empty expression.

2.4.3 Undo and Redo

When editing RPL expressions, undo and redo are available from the:

- **Edit** menu on RPL dialogs
- Right-click context menu
- Keyboard shortcuts
  - Undo is ctrl-z
  - Redo is ctrl-shift-z
- **Expression** menu on Expression slots.

Any action that edits a RPL expressions can be undone/redone including:

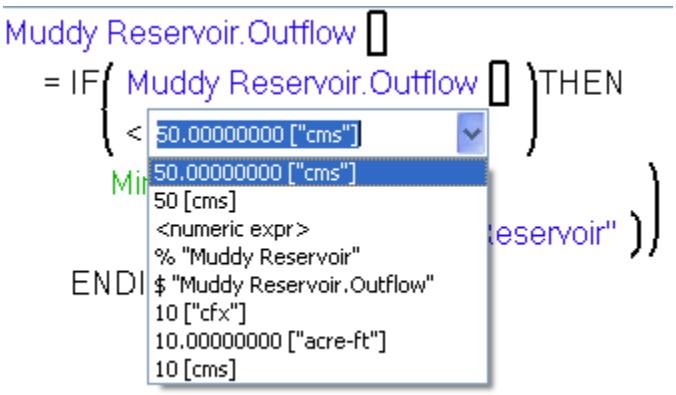
- Replacing the selected expression with a new expression by clicking on a palette button
- Cutting or deleting the selected expression.
- Pasting an expression in the place of the selected expression.
- Adding or removing a statement from a block.
- Adding or removing a constraint from a function.
- Double-clicking on an expression and changing it via an inline editor.

Undo and redo are on a per-dialog basis; the dialog must be selected before the undo/redo operation is performed. Also, the history of changes is preserved if a RPL dialog is closed and re-opened, but is not preserved if the set is closed. On expression slots, the history of changes is lost when the slot is closed.

The number of undos/redos is only limited to take you back to the original expression; there is no artificial limit imposed.

Finally, sections on the RPL set level cannot be undone including adding or deleting rules and re-arranging rule priorities.

2.4.4 Using the History



When you double click an unspecified expression or literal value, a pull-down menu appears to the right of the inline editor. This pull-down menu tracks the history of past edits that have been used in this dialog you can then use the pull-down menu to choose a value from the list. The arrow keys can also be used to scroll through the list.

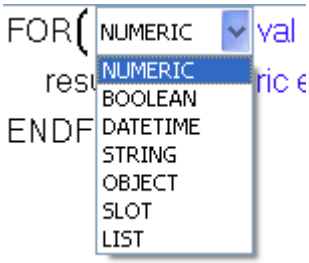
The history can also be used as a starting point if you type in an invalid expression. It provides an opportunity to correct the entry and try again without having to type it from scratch. For example, you wish to enter "Flood Season" into a string, but when you type, you only enter the string `"Flood S"` and hit Enter before finishing typing Season. RiverWare will not issue an error as "Flood S" is valid. You can then select `"Flood S"` from the pull-down history, and complete the desired text.

Note: The saved history it is a per-dialog history only but does persist when the dialog is closed and re-opened. It does not persist if the RPL set is closed.

2.4.5 Data types for Looping Variables

In the looping functions, FOR, WHILE, WITH, SUM, AVE and in the FOR statement, you specify the data type (More on data types can be found **HERE (RPLTypesPalette.pdf, Section 1)** of the looping variable. For example, the first line of a default FOR statement looks like
FOR (NUMERIC val IN <list>) WITH NUMERIC result = <numeric> do

You can change the type of the looping variable NUMERIC and the name of the variable. Pull-down menus provide an easy way to do this. You double click on the NUMERIC and is provided with a pull-down menu listing the 7 data types in RPL as shown to the right. A second pull-down menu is provided for the looping variable; this provides a history of all string values that have been previously entered in the dialog, similar to the history described in the previous section.

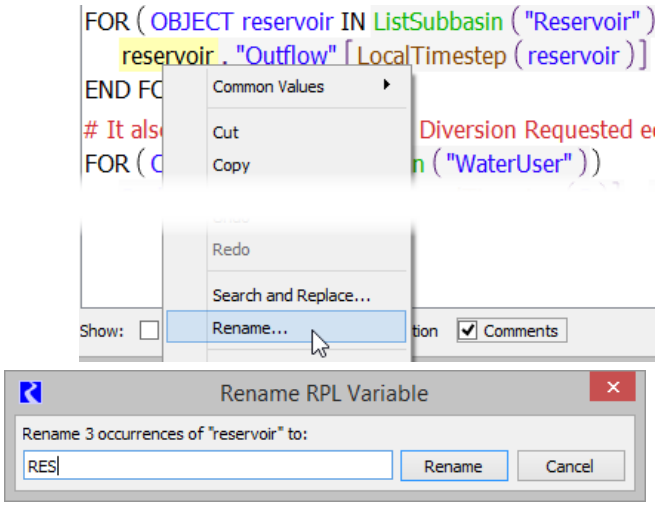


2.4.6 Renaming Looping Variables and Function Arguments

When a looping variable or function argument is selected, you can right-click and choose to **Rename...** all of the occurrences of that variable or argument within the expression and the definition. Note that variable names can be associated with statements (FOR and WITH), expressions (FOR and WITH), or functions (argument names).

To initiate a rename, use the right-click context menu and choose **Rename...** A dialog displays the number of occurrences of the name to be replaced, and allows you to type in the new name. Clicking **Rename** then makes the replacements.

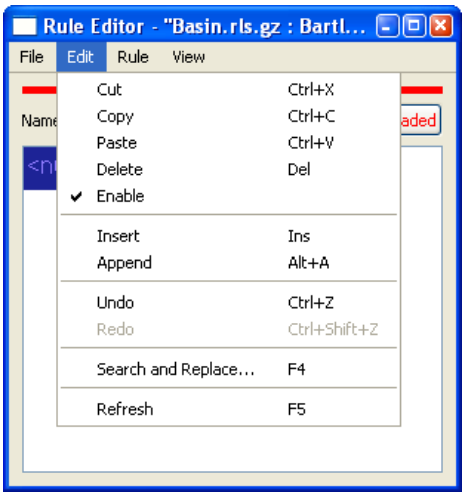
In the above example, three occurrences of the looping variable “reservoir” will be renamed “RES”.



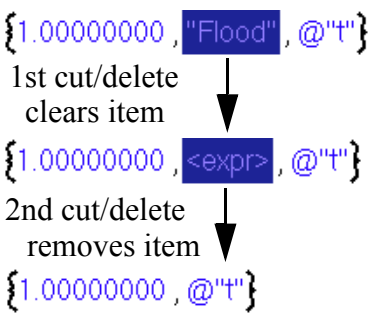
2.4.7 RPL Short Cuts

If part of an expression can be used in another assignment, rule, or function, rather than repeating the above process for each new expression, highlight the portion of the expression that can be used elsewhere. From the menu bar, select **Edit ➥ Copy** or type **<Ctrl C>** to copy the highlighted portion. Right clicking on the highlighted portion will bring up a menu with the option to copy. In the location in which the expression is to be pasted, highlight the **<expr>** and select **Edit ➥ Paste <Ctrl V>**, or right click and select **Paste** from the menu. This will paste the expression into the new location.

Also, expression can be deleted using the **Cut <Ctrl X>** or **Delete <delete>** operations. The **Cut** operation, puts the expression in the copy buffer, the **Delete** operation does not.



Note: There is one exception to the cut/delete behavior. When deleting an item in a list, the first cut or delete operation removes the expression but leaves a blank expression. A second cut/delete removes the item from the list. For example:

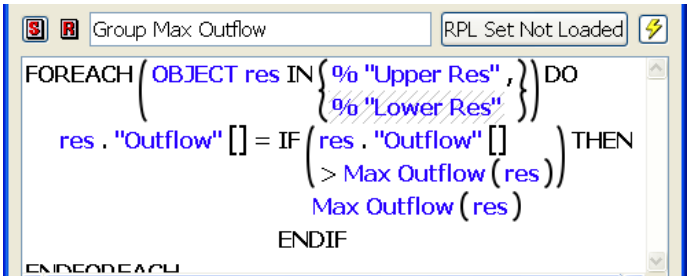


Statements may be cut, copied, pasted from the menu bar of the RPL Editor, using **<Ctrl C>,** or right clicking and selecting **Copy** from the menu. Select **Paste** to paste over an existing statement, **Insert** to add it above the currently selected statement, or **Append** to add below the last statement.

2.4.8 Disabling an Item in a List or a Statement

On the RPL set editor, functions and RPL blocks can be disabled by clicking on the green check mark turning it into a red X. Within a block, it is also possible to disable an individual item in a list or an entire statement.

- In the RPL editor, highlight the statement or item in the list that you wish to disable.
- Right click on the selected expression. Notice that there is a check mark next to the word “Enabled” indicating that the item is enabled.
- Click “Enabled” to toggle off the check mark thereby disabling the item. The disabled variable now shows up with cross hatching. The cross hatch color is defined in the set layout dialog **HERE (Section 6.2.2)**. When the expression executes, it will skip the disabled item(s).



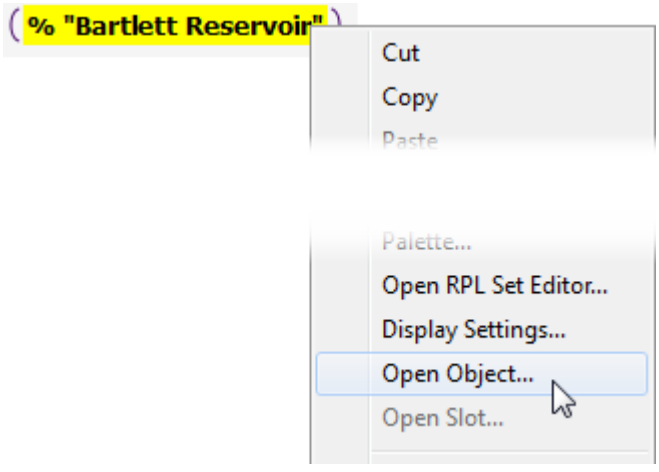
In general, disabling an assignment or an item in a list should only be used in the debugging and model building process. For example, you might want to have two assignment statements in one block. While debugging the model, it may be necessary to disable one assignment to ensure the other assignment is working correctly.

2.4.9 Open Slots and Objects from RPL dialogs

When the selected RPL expressions is a valid OBJECT or SLOT, the right click context menu

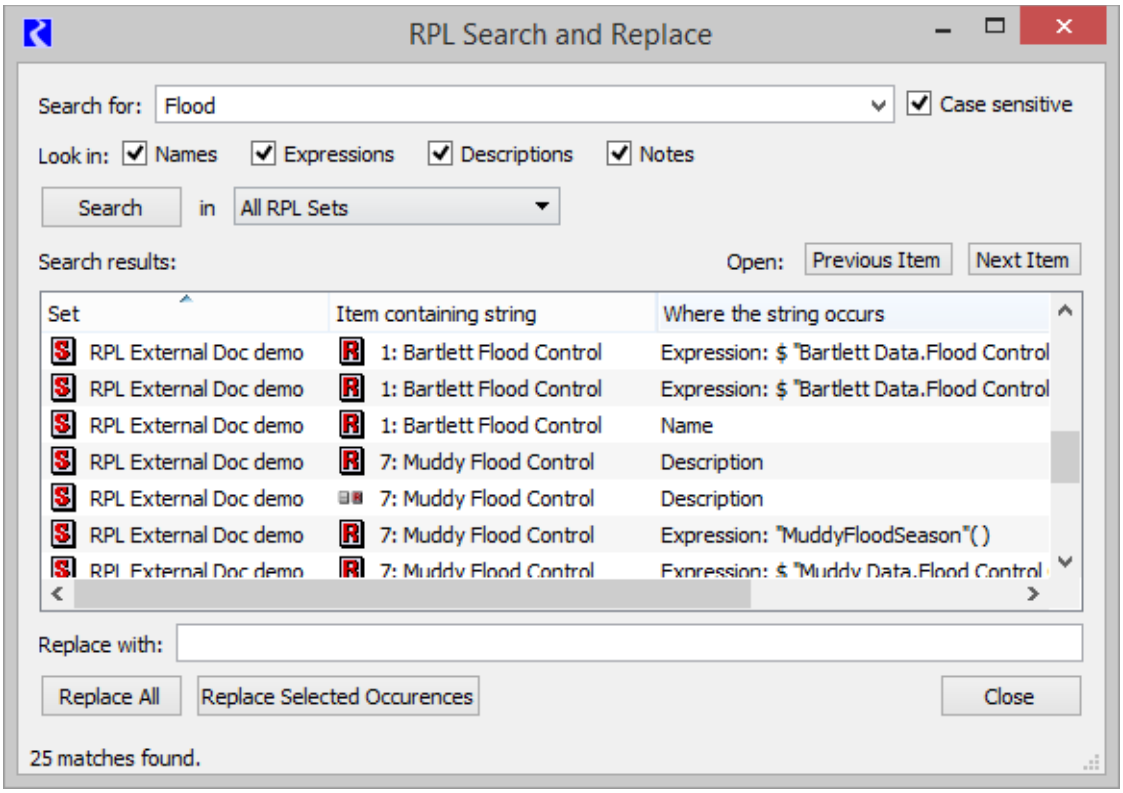
- **Open Object...**
- **Open Slot...**

will open that object or slot dialog. Note that this only works for expressions for which the workspace object can be determined without evaluation.



2.5 RPL Search and Replace Dialog

The RPL search and replace dialog supports flexible replacement of strings within a single RPL set or all RPL sets. Within this dialog, you can search for all occurrences of a string and replace all or some of them with another string.



2.5.1 Accessing the dialog

The RPL search and replace dialog is accessible in many ways:

- From the main workspace, select **Policy ➥ RPL Search and Replace...**
- Selecting **Search and Replace...** from the **Edit** menu of the RPL editor and its children dialogs (the RPL Set, Function, and Rule/Method editors) or from the RPL Set analysis dialog.
- Pressing the “F4” key from any of the RPL editors or the analysis dialog.
- Clicking on the right mouse button when editing any RPL expression and selecting “Search and Replace...” from the resulting popup menu.

2.5.2 Searching for occurrences of a string

To search for occurrences of a string, begin by typing the search string into **Search for:** field and then either hit **Enter** or click on the **Search** button.

When you initiate a search, the search string is added to a "history" of searches. To re-select a previous string, click on the triangle just to the right of the search string. This will present a menu containing previous search strings, allowing you to select one as the current search string.

Exactly where and how a search is conducted can be controlled by selecting various options:

- **Case sensitive** -- when checked, RiverWare searches for strings which match the search string exactly, otherwise it will ignore case.
- Look in **Names**: when checked, RiverWare tries to match occurrences of the string within the names of RPL items in the set. This includes the names of the RPL set and its contained groups, rules/methods, and functions.
- Look in **Expressions**: when checked, RiverWare tries to match occurrences of the string within all expressions in the RPL set. Eligible sub-expressions within an expression include values of type DATETIME, NUMERIC, BOOLEAN, STRING, OBJECT, and SLOT, as well as function calls (the name of the function being called is eligible for a match)
- Look in **Descriptions**: when checked, the contents of all descriptions (set, group, and rule/method) are searched for matching strings.
- Look in **Notes**: when checked, the contents of all notes (set, group, and rule/method) are searched for matching strings.
- Search **in:** If the dialog is accessed from the workspace Policy menu, it starts with **All RPL Sets** selected. Otherwise it will be configured to search in the RPL set corresponding to the dialog through which it was accessed. The name of the set selected will be displayed in a box to the right of the **Search** button. You may switch from this set to any other open set or to all sets by clicking on the set name and choosing a new set (or all sets) from the menu.

> **Note:** When you search in the **Expression Slot Functions Set** (or **All RPL Sets**), the expression slot expressions (i.e. the RPL expressions shown on the expression slot dialog) are searched along with the functions in the Expression Slot Functions Set. In this case, the parent is the object containing the expression slot.

The results of the search are displayed in the Search results table which contains one row for each string that matches the search string. Each row displays the name of the item containing the string (a set, function, group, rule/method, or expression slot) as well as where within that item the matching string was found (the object's name, description, or one of its expressions). In the case of a match within an expression, the actually matching sub expression is also displayed.

Note that a search string can match multiple times within a given item. For example, when searching for the string "res", a rule named "Forrest Reservoir Flood" will match twice.

Double-clicking within a row of the Search results table will open an editor for the item containing the matching string corresponding to that row. If the match is within an expression, the matching sub-expression will be the selected expression in the editor and the dialog will be scrolled to that

expression. Use the **Previous Item** and **Next Item** buttons to step through matches and opening/scrolling the dialog containing the match.

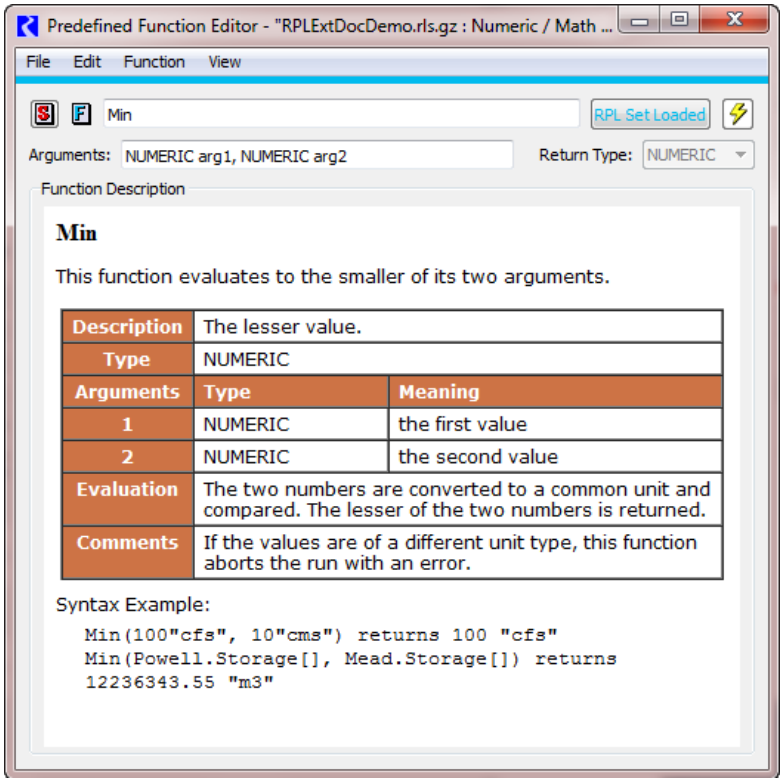
Note: If you wish to change the name of a function argument or looping variable (within a FOR or WITH), select the variable and use the right-click **Rename...** menu. This utility will rename all instances of that variable within the expression or function. This utility is described HERE (Section 2.4.6).

2.5.3 Replacing matching strings with another string

Once a search has been conducted, one can replace all or some of the matching strings with another string. First, type the replacement string into the field labeled **Replace with:**. To replace all of the occurrences then click on the **Replace All** button. To replace only some of the matching strings, select those that you wish to replace in the **Search results** list (click selects a single row, Ctrl-click allows multiple independent selections, Shift-click selects a range of contiguous rows), then click on the **Replace Selected Occurrences** button.

2.6 Functions

Following is a description of functions and their use in RPL sets.



2.6.1 Predefined Functions

Predefined functions are also useful in constructing rules, methods, and functions. Predefined functions are a set of mathematical, look-up, and mass balance routines that may be accessed from within any other expression. Predefined functions are coded into the RiverWare source code. Because of this, predefined functions cannot be modified. All predefined functions return their results in one of the standard data expression types; they can be substituted for any unspecified expression of that type. Predefined functions are accessed in the RPL Palette, HERE (RPLTypesPalette.pdf, Section 2), under the **Predefined Functions** tab at the top of the Palette dialog. Once

added to a RPL expression, double click the function to gets its editor. The documentation for the function is shown to help you specify the arguments and understand the evaluation.

Predefined functions are described in greater detail in the **RPL Predefined Functions** documentation **HERE (RPLPredefinedFunctions.pdf, Section 1)**.

2.6.2 Writing a User-Defined Function

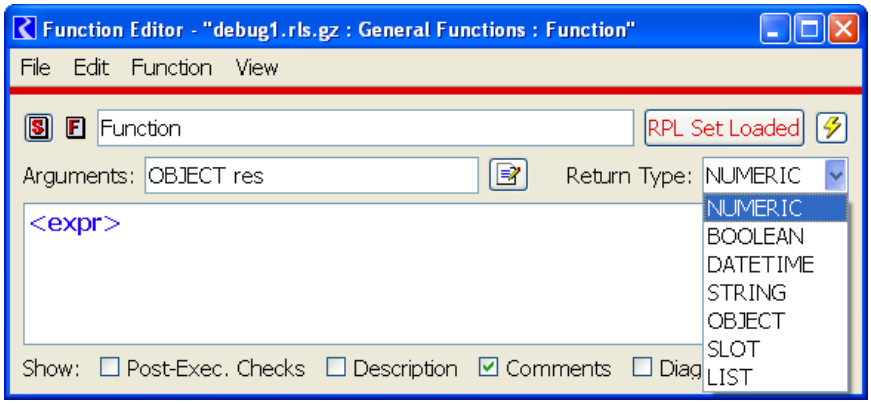
Functions are constructed in much the same way that blocks are constructed, using the RPL Palette. Expressions are simplified by using functions to perform logical and computational operations. To make a function flexible so that it may be used in a variety of situations, arguments are added to the function. Arguments permit the same function to behave differently, depending on from where it is called or by which object. For example, an internal function could be created which forecasts the evaporation from a reservoir based on the reservoir's surface area. This function could be of use at several reservoirs, but would have to know at which reservoir's data to look. The reservoir for which the function should compute the evaporation could be an argument to the function.

Arguments to functions can be of any data expression type. There is no limit to the number of arguments a function may take, but the number of arguments is not dynamic with respect to block execution. The exact number and type of argument(s) is fixed in the function definition. Dynamic argument lists and default argument values are prohibited.

- Add a function to the desired Policy or Utility Group by selecting **Set ➥ Add Function.**
- Name the function and press <Enter> after naming.

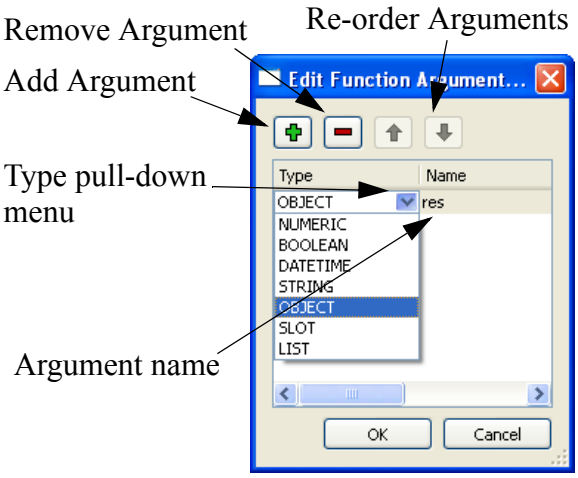
Note: If you are renaming a function that is called by other RPL expressions, a dialog asks if you wants to rename calls to this function in the applicable RPL sets. Answering **yes** will update existing calls to the function with the new name. For the Expression Slot set, the Initialization Rules set, the Iterative MRM sets, the Object Level Accounting Method set, Optimization Goal sets, and Rulebased Simulation sets only function calls within the set are affected. For Global Function sets, calls in any of the above sets will be affected. Calls to a global function from DMIs, scripts, or other places outside of RPL sets are not renamed. Answering **No** continues the name edit but does not update any calls to the function. Answering **Cancel** stops the name change all together.

- In the field **Return Type**, select the data expression type you wish the function to return. For example, if NUMERIC is selected the function will return a numeric value to the block in which the function is called.



If the function is going to be general to other functions or blocks, adding an argument might be useful depending on the return type of the function. In the above graphic, the **Arguments: OBJECT res** allows the variable "res" to be used inside of the function anywhere an **<object expr>** is allowed. This will allow the function to look at different reservoirs' data without creating a separate function for each reservoir. If the function is going to be used only in a limited manner, arguments are not necessary.

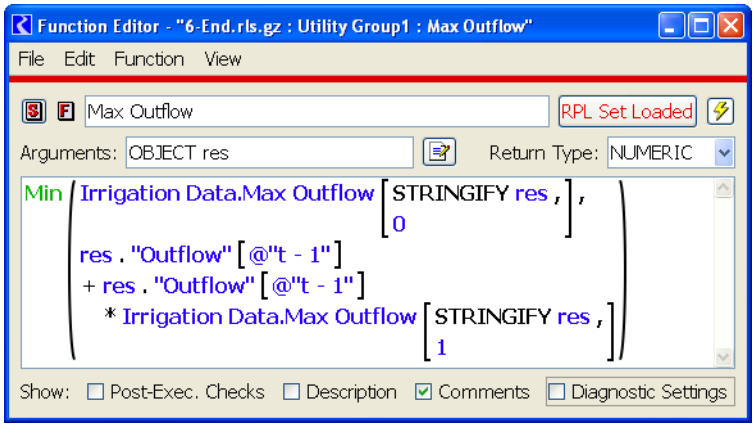
Arguments to functions can be entered in two ways, using the editor and by typing. The editor is accessed from the arguments button to the right of the Arguments line as shown to the right: Initially, the dialog is blank. You add arguments by clicking on the green check button. The red minus is used to remove arguments and the arrows are used to re-order arguments. The default type of an argument is NUMERIC. To change this, use the pull-down menu and select a different type. To change the name of the argument, double click on the name and type a new name. Click **OK** when finished. The screenshot shown to the right results in the argument shown below.



Typing was the original way to enter arguments. You type into the Arguments line using the syntax `TYPE argument`. Multiple arguments are separated by commas. For example, `NUMERIC flow, LIST elevations, OBJECT res`. The syntax, spelling, and capitalization must be exact or an error will be issued.

Note: To change the name of a function argument, select the argument in the expression and use the right-click **Rename...** option. This utility will rename all instances of that argument within the function's expression and definition. This utility is described **HERE (Section 2.4.6)**.

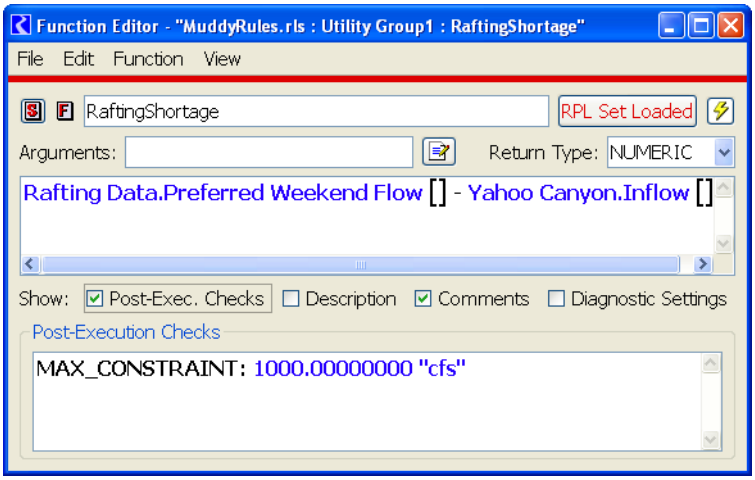
Functions are constructed like blocks and use the RPL Palette to build arguments. Below is a sample function constructed using predefined functions and RPL Palette buttons.



The same short cut copy and paste abilities seen in blocks apply to functions as well. Any time an expression is used multiple times, whether within a function or between functions, time can be saved by copying and pasting that expression into the other locations in which it is used. As with blocks, it is wise to check the validity of an individual function after it is built and fix it rather than wait until the beginning of a run to check the validity of all blocks and functions.

2.6.3 Constraints on Functions

Functions can be constrained to evaluate to either a minimum or maximum value. Or they can be configured to flag an error if a minimum or maximum constraint has been violated.



- From the Function Editor, Select **View ➥ Show Post-Execution Checks** or click the **Post-Exec Checks** toggle.

A constraint can be added using the **Function** menu item.

- Select **Function ➥ Add Min Constraint** or **Add Max Constraint** (as shown in the screenshot). The palette is then used to set up the conditions of the selected constraint.

2.6.4 Time Invariant Functions and Function Value Caching

The function editor's Edit menu provides a toggle control labelled "Set Time Varying" which is set to **on** by default. Toggling this property **off** communicates to RiverWare that the function is guaranteed to evaluate to the same value each time it is evaluated, i.e., it is *time invariant*. Note that functions with arguments will almost certainly not be time invariant; if a function has an argument, then presumably there are some argument values for which the function will evaluate to different values. If a function with no arguments is time invariant, then the first time the function is called within a run, the body will be evaluated and the result saved internally. For all subsequent calls of that function within the run, the cached value will be returned without further computation, reducing computation time. Note that incorrect application of caching to a time varying function will lead to incorrect results, so we

recommend that the time varying property be toggled **off** for a function only when it is definitely time invariant, run time is critical, and RPL set analysis has indicated that a significant portion of the run is spent evaluating the function.

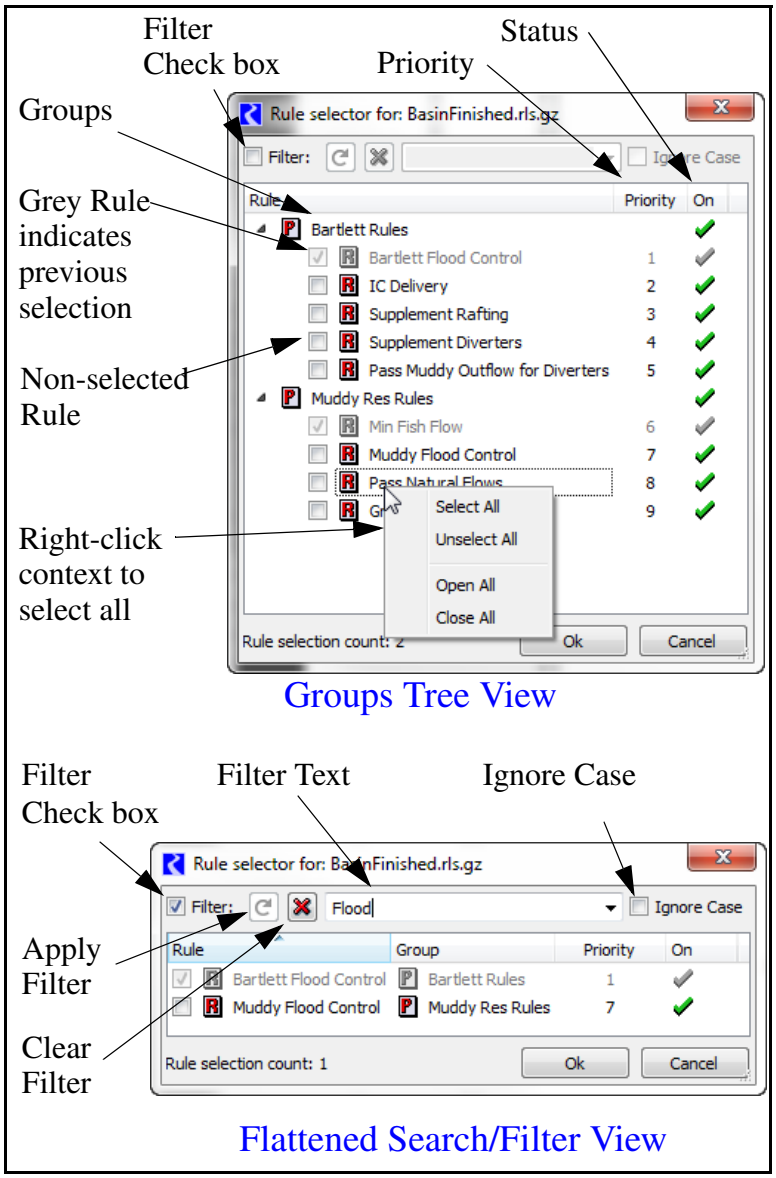
Note that during block evaluation (e.g. within a rule or accounting method), the workspace remains unchanged (because RPL expressions evaluate without side effects), so it is safe to cache values for functions without arguments within a single block. RPL does this automatically for all functions without arguments; multiple evaluations of such a function within the same block execution will cause the function to evaluate only once.

2.7 Selecting RPL Items

The RPL Item selector dialog is used whenever groups, rules, or functions (i.e. RPL items) are to be selected. For example, in diagnostics, you may select the rules for which you want to print messages.

This dialog initially shows the groups and rules in a tree view. Also shown are the rule priority and rule status. In this dialog, you select the RPL items by clicking on the boxes to add check marks next to one or more items. Depending on the context from which this dialog is called, you may be able to select one or many items. Items that have been previously selected for display are shown in grey and cannot be deselected. They can only be removed from the dialog from which this dialog was called. Note, all items can be selected by right-clicking and choosing **Select All**.

You can also click the **Filter** check box to show a flattened list of the rules. Then you can sort by column heading or filter by entering a text string at the top. Use the green arrow to refresh, red X to clear the filter, and the Ignore Case check box to do a case insensitive filter. Select RPL items by checking the boxes as described above.



2.8 Developing Efficient RPL Expressions

RPL expressions are built using the palette (HERE (RPLTypesPalette.pdf, Section 2)). Together with user and pre-defined functions, HERE (RPLPredefinedFunctions.pdf), these provide all of the pieces necessary to create a simple or complex RPL expression. Typically, an analysis of your RPL set's performance is necessary to locate slow or inefficient items. Tools for this can be found HERE (RPLDebugging.pdf, Section) and in particular the **RPL Analysis Tool** located HERE (RPLDebugging.pdf, Section 5).

Following are some suggestions to writing efficient RPL sets in terms of performance.

- Use predefined functions or operators when available. In general, always prefer a built-in operator or predefined function to a user-defined function (or complex expression) which performs the same computation.
- Move complex logic to user defined functions. This not only makes the logic more readable and easier to debug, but it makes performance analysis easier. This is because the RPL set analysis tool reports the times for function calls but not for other parts of expressions (i.e., the granularity of the performance information is "per function").
- Use WITH expressions to avoid re-evaluating expensive expressions.
- Make sure there is no unnecessary LIST processing or STRING manipulation.
- OBJECT and SLOT representations of workspace objects and slots are generally more efficient than string representations of them. For example, the expression

  GetSlot((STRINGIFY reservoir) CONCAT "^" CONCAT account CONCAT ".Storage")

  probably takes significantly longer to evaluate (and is more complex) than

  reservoir ^ (account CONCAT ".Storage")
- For a large model, it is computationally expensive to obtain a SLOT given the full slot name (i.e., a STRING representation of the slot) because RiverWare must first break the string into its components, then look for an object with the appropriate name. Once the object is found, then for accounting slots RiverWare must search through the accounts. Finally a search is conducted for a slot on the object/account with the given slot name. Thus when referencing slots, one should take special care to apply the suggestions mentioned above.
- Make sure that time consuming functions like FloodControl() or the hypothetical simulation functions are not being called more than necessary.
- Use functions with no arguments to cache values. If a function has **no** arguments, then the first time it is executed in a block (a rule, accounting method, optimization goal, initialization rule, MRM rule or expression slot), the return value is cached. For the remainder of the block execution, the function need not be evaluated again, the cached value is used. Thus, if you have multiple assignments within one rule that call the same argument-less function, the function will be evaluated once and the value will be used for all function calls.
- If you are sure that a function with no arguments is always going to return the same value regardless of the timestep, set the **Set Time Varying** toggle to be off. This will lead to the function's first return value being reused throughout the run. Click HERE (Section 2.6.4) for more information on this feature.

- MAPLIST is an efficient expression which operates on a list *and* returns a list, but if one wants to operate on a list and compute a single value, FOR expressions (or a variation of FOR, like SUM or AVE) are more efficient. E.g.

```
Sum( MAPLIST (...))
```

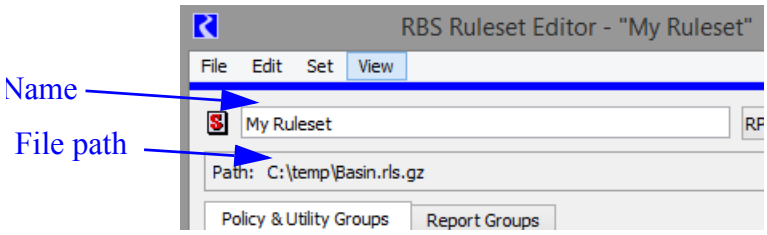
is slower than

```
FOR ( ... ) SUM
  ...
END FOR
```

3. Example: Creating a new RBS Ruleset

Following is an example of how to build a new Rulebased Simulation (RBS) ruleset. Similar actions can be used to build any of the other types of RPL sets.

- **Policy ➥ Ruleset ➥ New...** from the main menu bar. This will bring up a blank RPL Set Editor dialog.
- Enter the desired Ruleset Name.
- Save the ruleset to the desired directory by selecting **File ➥ Save** or **File ➥ Save As** from the menu bar of the Ruleset Editor dialog.



Adding Groups, Rules, and Functions:

To begin the ruleset, add the desired number of policy groups and utility groups by selecting:

- **Ruleset ➥ Add Policy Group** or **Ruleset ➥ Add Utility Group** from the RPL Set Editor's main menu bar.
- Name the Policy or utility Group by double clicking on its icon to open the Policy Group Editor or Utility Group Editor. Right clicking on an icon will bring up a menu from which the Policy Group or Utility Group Editor can be accessed. Name the Policy or Utility Group and type <Return> after entering the name.

Policy groups are used to organize blocks and functions. Utility groups are used to organize functions. Although an entire set can be created entirely within a single policy group, it is not recommended. Blocks should be split up into different policy groups according to object, geography of the model, operation type, and/or priority of operations. Supporting functions which are common to several rules should be placed in utility groups.

To add a block or function:

- Click on the Policy or Utility Group so that it is highlighted.
- Select **Ruleset ➥ Add Rule** or **Ruleset ➥ Add Function.**

To view the elements in the policy or utility group, click on the tree-view triangle left of the group name.

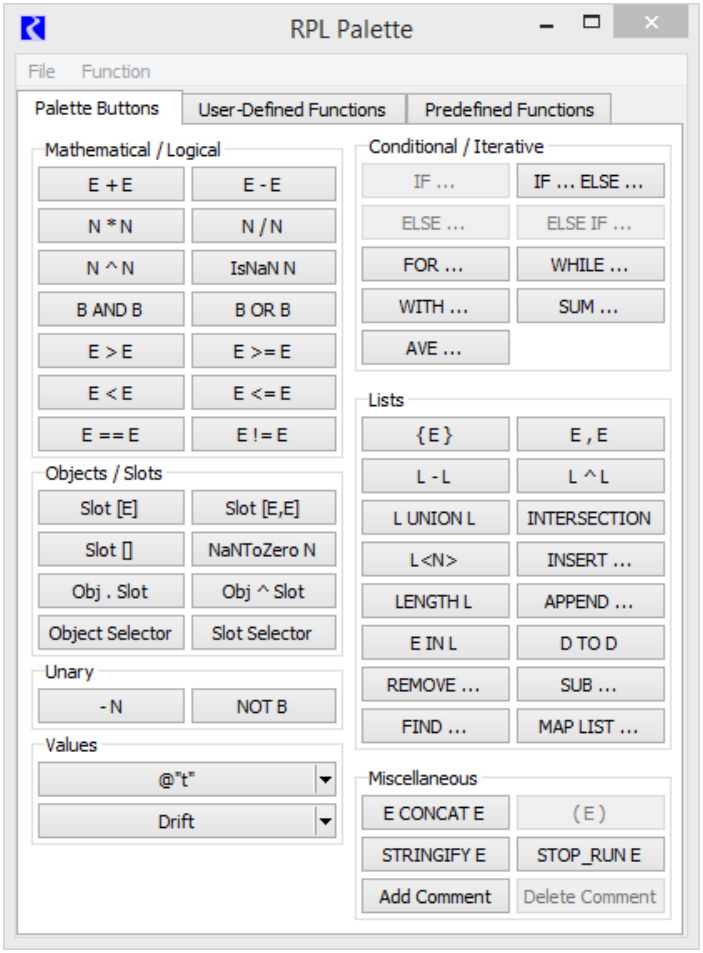
Unspecified expressions are portions of the block that have not yet been set when building a rule. These expressions are represented as the expression data type.

- Add an Assignment using the **Rule ➥ Add Assignment** menu.

Expressions are built using the RPL Palette. The RPL Palette contains most of the expression elements needed to build a block, including logical operators, slot and object access expressions, flow operators, list operators, predefined and user made functions. Click **HERE (RPLTypesPalette.pdf, Section 2)** for more information on the palette. Depending on the data type of the expression highlighted in a block, the RPL Palette only enables buttons for expressions of that data type. The data types are described **HERE (RPLTypesPalette.pdf, Section 1)**.

To open the RPL Palette:

- select **Rule ➥ Palette** from the menu bar of the **Rule Editor** dialog**.**



To finish building the rule:

- Highlight the left **<numeric expr>**. This will be the slot to which the rule will assign a value.
- In the **Objects/Slots** section of the RPL Palette, click on the **Slot[]** button. This replaces the unspecified numeric expression with a series slot expression.

The square braces following a slot expression refer to the timestep or the row and column of the slot. There are three slot expression possibilities:

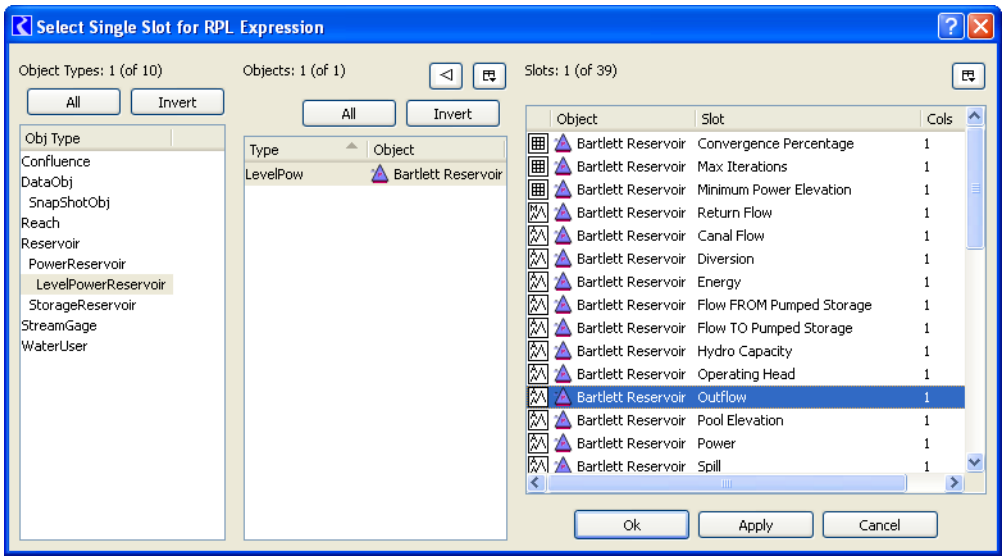
Slot [E] A series slot at the timestep indicated by **E**.

Slot [E, E] A table slot at the row and column indicated by **E** and **E**, respectively. Table slot column references are zero based, i.e. the first column is referred to as column 0.

Slot [ ] A series slot at the current controller timestep or a scalar slot's value

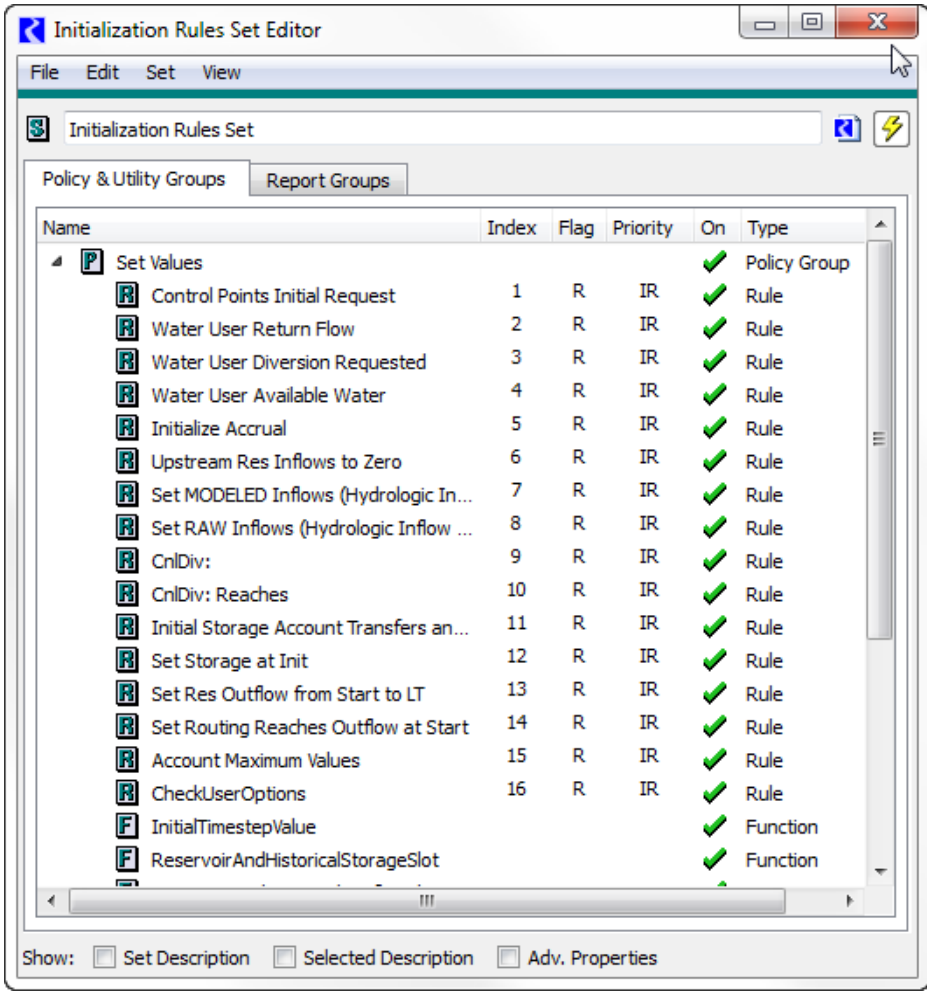
Next:

- Highlight the unspecified portion of the series slot expression, **<expr>**, and click on the **Slot Selector** button in the **Objects/Slots** section of the RPL Palette. This will bring up the slot selector dialog.



- In the Single Slot Selector, select the appropriate **Object Type, Object,** and **Slot.** Then click **OK.**
- Finish building the rule by building the right-hand side of the rule. The RHS should return a numeric value and units that are to be set on the slot specified on the left-hand side.

4. Initialization Rules Set



Initialization Rules are a set of RPL rules associated with the model which can be executed as part of run initialization to "set up" data for the run. Initialization Rules provide a complement to user inputs and execution of DMIs.

Unlike other rules which can only set series slots, initialization rules can be used to set Table Slots and Scalar Slots.

This section describes the applicability and use of Initialization Rules. The user interface for Initialization Rules is described in general with the other RPL sets **HERE (Section 1)**.

Applicability (when to use Initialization Rules): Each method for providing data to a simulation is suited to different modeling scenarios. For situations in which there are a small number of inputs that don't change frequently, interactive setting of input values is often the simplest approach. However, when this process would be time-consuming or error-prone, it is preferable to automate the setting of user inputs. Input DMIs provide a flexible way of accomplishing this by providing mechanisms for communication between RiverWare and external programs (usually databases). If there is computational logic involved in this process, e.g., if the data need to be massaged to remove outliers, then this logic must somehow be embedded in the external DMI program. On the other hand, using initialization rules to accomplish the same task allows this logic to be viewed and edited from within RiverWare, often improving the clarity and ease of maintenance of the model.

Initialization Rules also differ from DMIs in that in the context of a RBS simulation, they can be used to provide default values which the policy can then override. To illustrate a scenario in which this might be useful, consider a run in which some quantity is generally assumed to be zero, but which can take on non-zero values under some conditions. One strategy for modeling this situation is to set the relevant

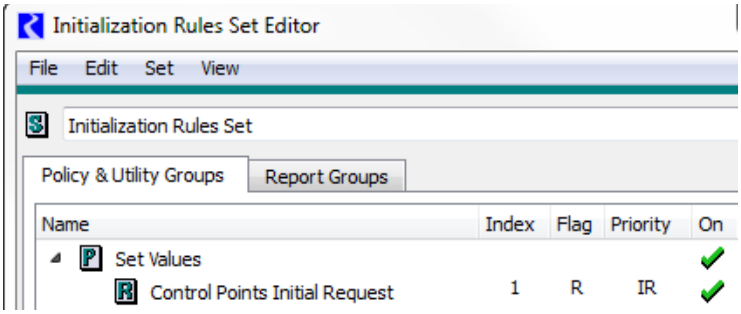
values to zero using low priority rules with execution constraints that cause them to execute only on the first timestep. Alternatively, Initialization Rules accomplish this same behavior with less user effort (no execution constraints are necessary) and with the additional advantage that these rules are organized separately from the policy (if there is one). In fact, most RBS rules which are constrained to execute only on the first timestep are likely to be more appropriately included with the Initialization Rules.

Another important distinction between Initialization Rules and policy (rules) is that the former are executed before data checking at the beginning of the run. Thus, Initialization Rules can be used to set values such as initial **Storage** or **Pool Elevation** on a reservoir, whereas RBS rules execute only after data checking and so could not be used to give these slots reasonable values. Further, unlike other rules which can only set series slots, initialization rules can be used to set table and scalar slots.

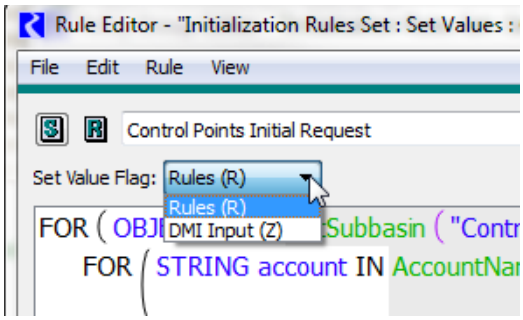
Behavior (How to use Initialization Rules): Following is a description of how to use initialization rules.

- **How do I access the Initialization Rule Set?** From the workspace, use the **Policy ➥ Initialization Rules RPL Se**t menu.
- **Where are they saved?** The set is saved in the model file. The Initialization Rules Set Editor dialog does have a **File ➥ Save Initialization Rules Set As** menu item to save the initialization rules to a file. A **File ➥ Replace Initialization Rules Set from File** menu item allows the initialization rules set to be replaced by the contents of a specified file. These menu items allow an initialization rules set to be moved between models via a file. However there is only a single instance of initialization rules in a model and this set is still saved and loaded with the model file
- **When are they executed?** The rules are executed during the initialization phase of all Simulation and Rulebased simulation runs (including accounting) after values are cleared and inputs are registered but before beginning of run checks on all objects. The exact ordering is described **HERE (Simulation.pdf, Section 5)** for simulation, **HERE (, Section 1.7.2)** for rulebased simulation, and **HERE (Accounting.pdf)** for accounting.
- **In what order are they executed?** The Initialization Rules are executed in Index order based on the specified “Agenda Order”. To see the order, click **View ➥ Show Advanced Properties.**
- **What do Initialization Rules set?** Initialization rules can set series slots, table slots, and scalar slots. For series slots, the current timestep is the start timestep. For table slots, specify the rows and columns using labels or a zero based index. For Scalar slots, use the Object.ScalarSlot[ ] syntax on the left side of the assignment statement.
- **Do Initialization rules re-execute when dependent slots change?** No, initialization rules execute once and do not re-execute even if dependencies change.

- **Do Initialization Rules have Individual Priorities.** No, the Initialization rules are organized by **Index** as shown on the RPL set editor.



- **What Flag/Priority are the values given?** Series values set by Initialization Rules are given one of the following flags/priorities as configured through the **Set Value Flag** menu on each Initialization Rule:

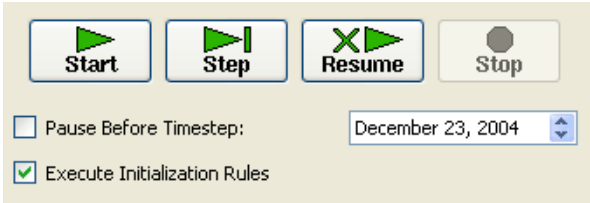


  - **Rule (R)** - Set the value with the "R" flag. In RBS, all values set this way are given a very low priority (It is a very large number) and the letters "**IR**" are shown indicating the Initialization Rule priority. R flagged values set by Initialization rules can be overwritten like other R flagged values.
  - **DMI Input (Z)** - Set the value with the "Z" flag and give it a priority of zero (0) (in RBS). When simulating, this flag behaves identically to the INPUT (I) flag. Note, an Initialization Rule setting a value with the Z flag can overwrite an "I" or "Z" flagged value.

Note: The values set by initialization rules are given the priority as described above. If these values are propagated or provide enough information for an object to dispatch, any subsequent values set will be given the controller priority, which would be zero (0) if they solve before any rules execute.

Note: Table and Scalar slot values set by initialization rules are not give a priority or a flag. Choosing the R or Z flag from the menu shown above results in identical behavior for these types of slots.

- **Can I disable the Initialization Rules?** When the Initialization Rules set contains at least one rule and the current controller has the possibility of executing the Initialization Rules, then the Run Control Dialog displays a check box labeled "**Execute Initialization Rules**" which controls whether or not to execute the Initialization runs during the initialization phase of runs. Uncheck this box to disable the Initialization Rules.



- **How do I know which Initialization rule set a slot value?** Tooltips on the series slot values display which Initial-

ization Rule (or RBS rule or DMI) set that value. See the screenshot to the right. Click **HERE (Slots.pdf, Section 4.1)** for more information on these tooltips.To open that rule, right click and choose **Open Init Rule #.**

- **How do I debug Initialization Rules?** The RPL debugger can be used with Initialization Rules. Also, the following diagnostic categories in the Rulebased Simulation and Simulation settings dialog control diagnostics for the Initialization Rules:
  - **Init. Rules Print Statements**
  - **Init. Rules Rule Execution**
  - **Init. Rules Function Execution**

Debugging and analysis of RPL sets is described **HERE (RPLDebugging.pdf, Section 1)**.

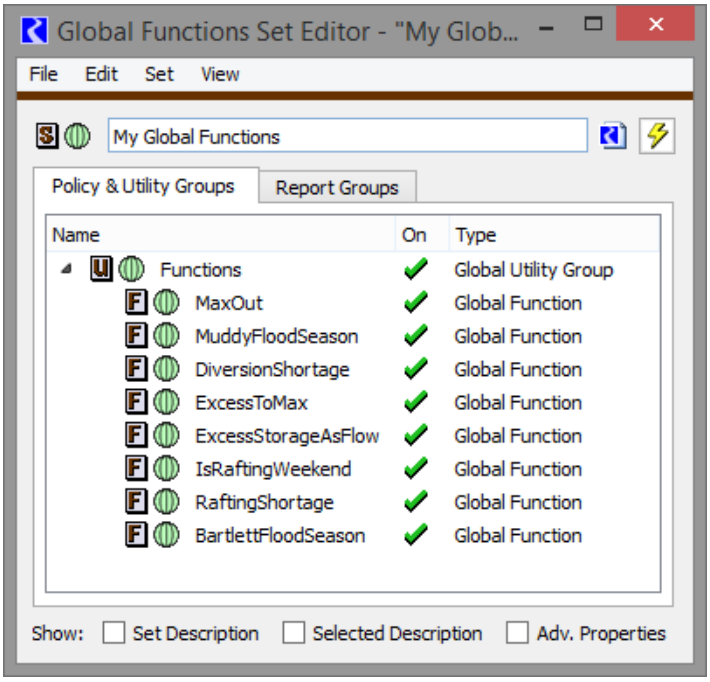
5. Global RPL Functions

Global functions allow you to define a function in one location and then use it in any of the RPL applications including Rulesets, Optimization Goal Sets, Object Level Accounting Method Set, Initialization Rules, MRM Rules, and RPL Expressions Slots.

Global RPL Functions exist within Global Functions Sets, organized in Global Utility Groups. Multiple Global Function Sets can be opened within a RiverWare session.

A RPL set is considered “global” when it is opened as a Global Set and any RPL Set file can be loaded as a Global Function RPL Set. (Only the RPL Set’s Utility Groups -- and not its Policy Groups -- are relevant in Global Function Sets).

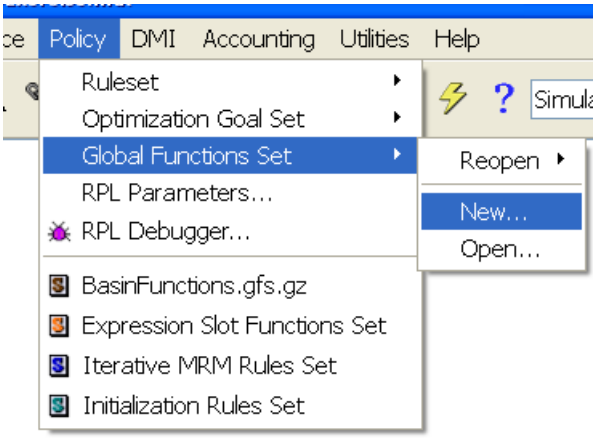
Global RPL Functions can be called from RPL Functions and Rules (and other forms of RPL Blocks) in any open RPL Set. Global Functions can call other Global Functions, either in the same Global Function Set, or a different Global Function Set.



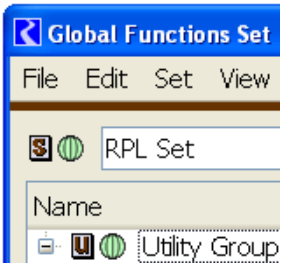
5.0.1 Creating a new Global RPL Function Set

A new Global RPL Function Set can be created from the RiverWare Workspace Menu, with this menu item:

Policy ➥ Global Function Set ➥ New...

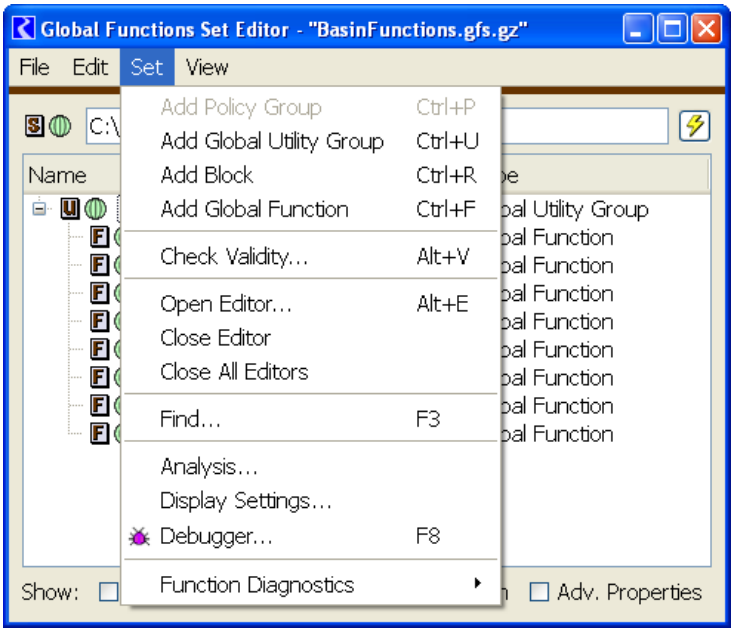


This opens up a Global Function Set editor. This is a RPL Set Editor for Global Function Sets, distinguished with a *green globe* icon in the upper-left area of the editor dialog:



As with all RPL Sets, a Global Utility Group must first be added before new user-defined RPL Functions can be defined. In a Global RPL Function Set, this is done from the RPL Set Editor's menu item:

Set ➥ Add Global Utility Group



Any function added to the new Global Utility Group is, by definition, a Global Function. A Global Function can be created in any of the following ways (not illustrated):

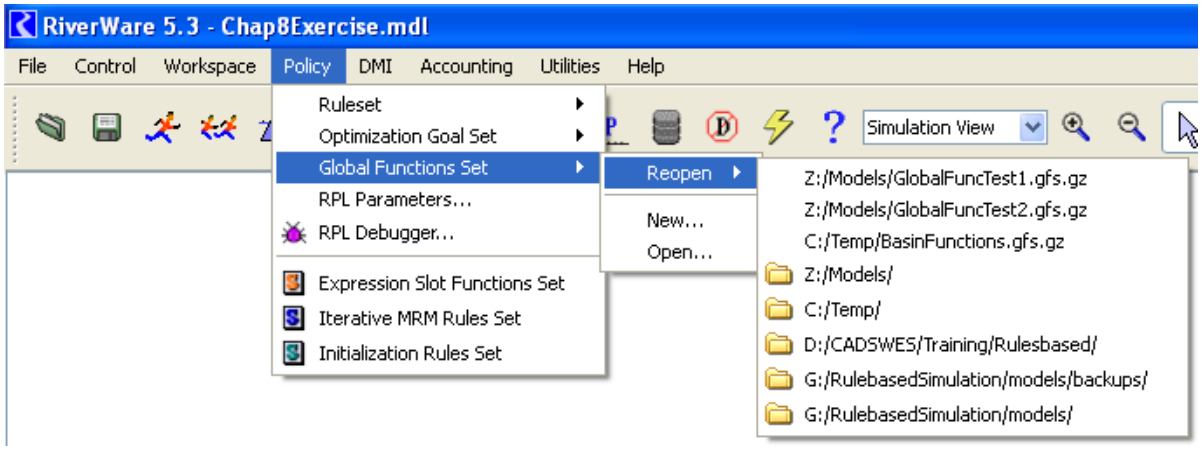
- From the RPL Set editor, with the desired Global Utility Group selected, using the Set ➥ Add Global Function (Ctrl+F) menu item.
- From the RPL Set editor, using the Context Menu (right-click) on the desired Global Utility Group item, Add ➥ Global Function.
- From the Global Utility Group Editor, using the Group ➥ Add Global Function (Ctrl+F) menu item.
- From the Global Utility Group Editor, using the Context Menu (right-click) anywhere within the function list, Add ➥ Global Function.

- By Copying a Function from another RPL Group or RPL Set, and Appending into the Global Utility Group, using Context Menu operations.
- By Dragging a Function from another RPL Group or RPL Set into the Global Utility Group. *NOTE:* The Drag-and-Drop functionality currently implemented in the RPL Set and RPL Group editors has some limitations, including the availability of that operation only if the target function list already has at least one function.
- By importing Utility Groups from a RPL Set file. This is done from the RPL Set editor, File ➥ Import Set... menu item.

Only one instance of a function can occur within all loaded RPL sets and Global Function Sets. For example, you cannot have a function named "GetMinimumFlow" in a utility group in your ruleset and another named "GetMinimumFlow" in an open Global Function Set. An error will occur in this case.

5.0.2 Opening an Existing Global Function Set

The RiverWare Workspace Policy menu supports the opening of a RPL Set as a Global Function Set in ways similar to opening other RPL Set files.

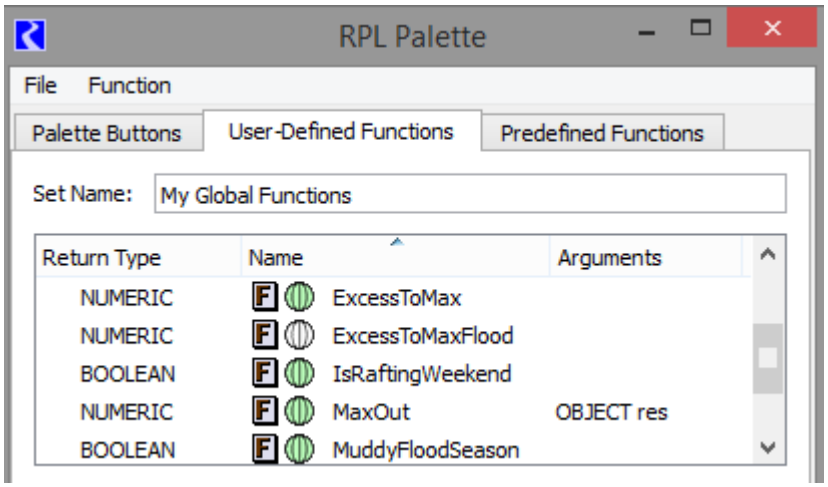


Any RPL Set file can be loaded as a Global Function Set, however only the Utility Groups within the RPL Sets loaded as Global Function RPL Sets are usable -- *Rules (or other forms of RPL Blocks) within the set's Policy Groups are not.* If a RPL Set file containing Policy Groups is opened as a Global Function Set, the illustrated warning dialog is shown, and a similar warning message is written to the RiverWare diagnostics output.

It is important to open any Global Function Sets before trying to load a ruleset that uses functions in a global set.

5.0.3 Using Global RPL Functions

When editing a RPL Expression, any Global Function in any open Global RPL Set can be called. The standard way of adding a call to any user defined function is via the RPL Palette's User-Defined Functions tab.



Within the User-Defined Function list, global functions are indicated with one of two *globe* icons:

- A **white globe** -- indicates an "external" Global Functions... i.e. a global function in a RPL Set other than the set containing the RPL expression being edited.
- A **green globe** -- indicates a "local" Global Function... i.e. a global function within the same RPL Set as the one containing the RPL expression being edited.

During evaluation of a Global RPL Function, the specific behavior is consistent with the caller's RPL Application. For example, "@t" (the current timestep) represents the Rulebased Simulation Controller's current timestep when the function is called from an Rule. But when called from a Series Slot with RPL Expression, "@t" (current timestep) represents a particular series timestep date/time.

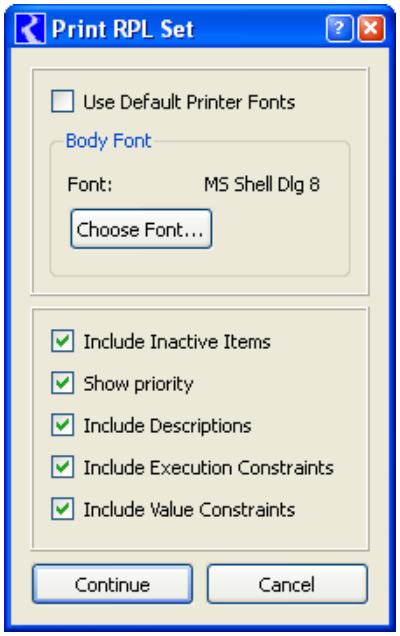
6. RPL Printing and Formatting

Blocks written in the RiverWare Policy Language can be complex, and with lengthy verbose object, slot, and function names, blocks can have very long lengths. RiverWare now provides a mechanism to print blocks and dynamically format blocks to fit the width of desktop window or a printer page.

6.1 Printing

Rules and user defined accounting methods written in the RiverWare Policy Language can now be printed. A Printing dialog is available at each editor level from the **File ➥ Print** X menu. This provides you with the means to print an individual block from the Editor, an individual function from the Function Editor, an individual group from the Group Editor, or the entire RPL Set from the RPL Set Editor.

Using the print dialog, the block, function, group, or RPL set can be sent directly to the printer or to a file. You may select landscape or portrait printing. You may choose to include or exclude inactive items, descriptions, execution constraints, and value constraints. You may also choose the print font. A note on font selection. The font selection dialog presents a list of fonts that are available on computer console, these fonts may or **may not** all be available on the printer. If the font is not available on the printer, a default font will be chosen by the printer and the printout may be misaligned -- formatting will be corrupted. Some experimentation may be necessary to determine which fonts an individual printer supports.



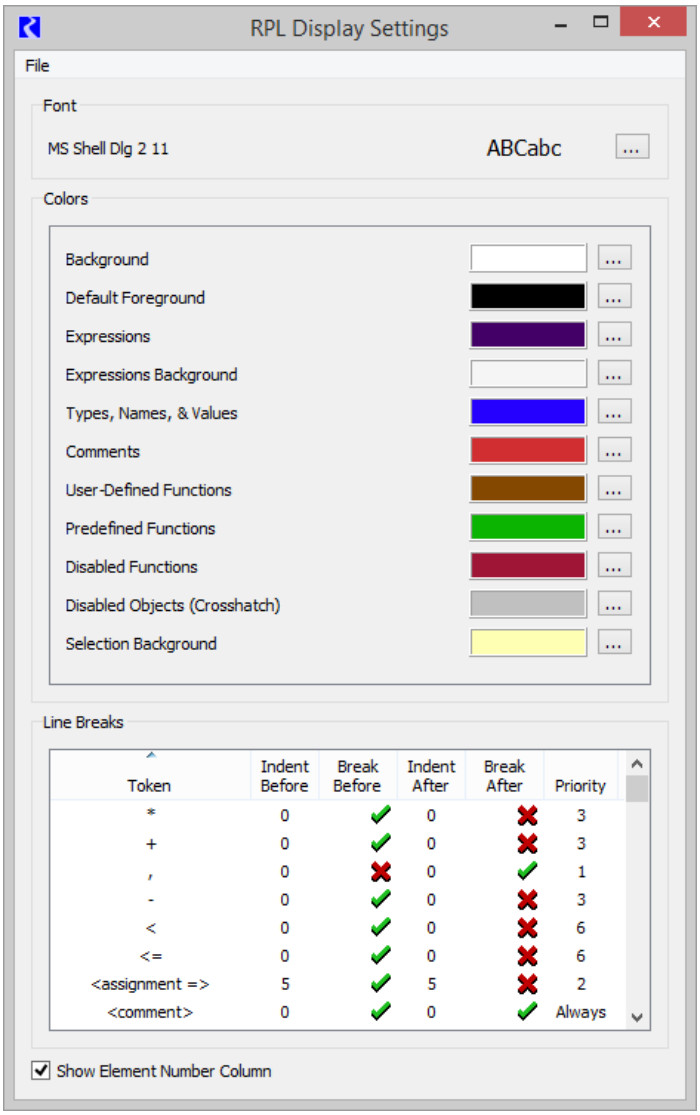
6.2 Formatting - The Display Settings

Expressions written in the RiverWare Policy Language can be complex and very long. RiverWare provides a **RPL Display Settings** dialog to control the fonts, colors, showing element numbers, and a dynamic formatting algorithm. This algorithm formats the blocks to fit the width of desktop window or a printer's page. The dialog is accessed through the **Display Settings...** button in the editor's **Set** menu. These settings will be automatically saved in the model file. This allows RPL logic to look the same from one user to the next.

Display settings may be imported and exported to/from a file through the "File" menu. Following are the options:

- **Import File:** Import the settings from a file that was exported from a RiverWare session.
- **Export File**: Export all of the settings to a file.
- **Save as User Defaults**: Export the settings as user settings that are stored in the machine's registry. These settings will be used for new models.
- **Load User Defaults**: Load the settings from the user settings defined in the machine registry. This allows you to override a model's settings with your personal settings.
- **Load RiverWare Default Colors**: Load the RiverWare default colors. This allows you to get back to the base starting colors.
- **Load RiverWare Default Line Breaks**: Load the RiverWare default line breaks. This allows you to get back to the base line breaks.

There are three area to this Display Settings dialog, **Font, Colors, Line Breaks,** and **Element Numbers** as described as follows:



6.2.1 Font

The fonts used in RPL expressions can be changed by clicking on the "..." button, ... then select the desired font, style, size, and effects.

6.2.2 Colors

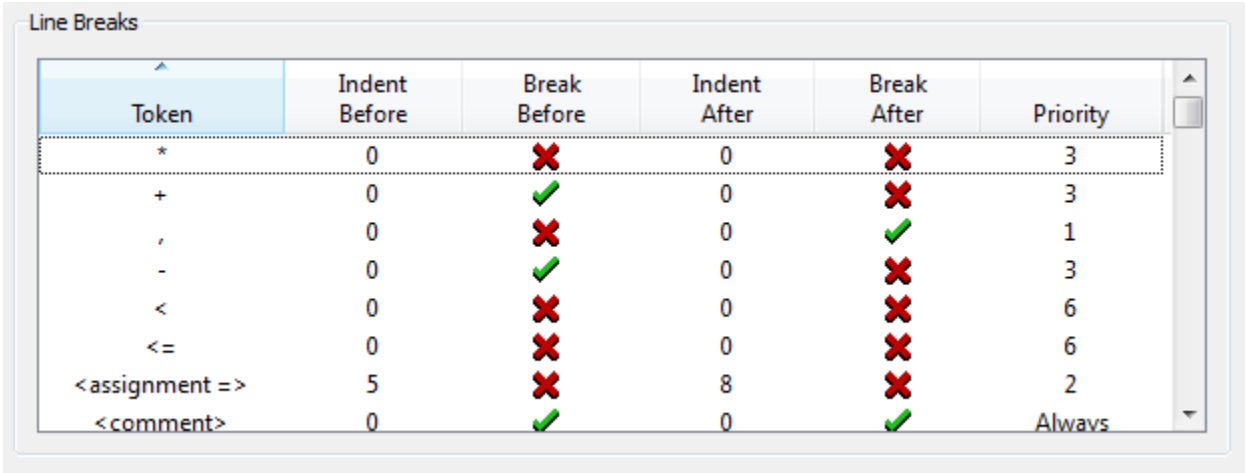
The colors used in RPL set editors can be changed to fit your needs. You are able to specify the colors for items shown in the following table. To change, click on the "..." button, ..., then select the desired color from the color chooser.

Item	Description
Background	The background color for statements and other areas (except expressions) that are not contained within the current selection.
Default Foreground	The color used to draw items in expressions or statements whose color is not governed by another color setting. For the current selection, this color is used for all items; outside of the current selection, this color is used to draw mathematical symbols, operators, and delimiters. Note, the selected expression's foreground text is either white or black (dynamically computed) to contrast with the configured background color.
Expressions	The color for key words in expressions.

Item	Description
Expressions Background	The color used for the background of the bounding rectangle for expressions.
Types, Names, & Values	The color of types (e.g., NUMERIC), names of variables, and literal values.
Comments	The color of in-line comments.
User-Defined Functions	The names of user-defined functions.
Predefined Functions	The names of predefined functions.
Disabled Functions	The names of user-defined functions which are disabled (i.e., functions for which the red X appears in the “On” column of the functions RPL Set and Group editors).
Disabled Objects (Crosshatch)	The color of the crosshatching superimposed on disabled statements or expressions.
Selection Background	The background color for statements and expressions that are contained within the current selection.

6.2.3 Line Breaks

The formatting algorithm is designed to provide you with control over the formatting process. The formatting algorithm is multi-pass algorithm. Each pass will attempt to find a position in the block where a line break can be placed to fit the block to the device width. The algorithm will cease, once the block can be rendered in the desired width, or no more positions where the block is allowed to be broken exist. You can define, through the **RPL Display Settings** dialog, the tokens (i.e., block positions) a line break can appear before or after, the indentation following the line break, and the priority (the pass) at which the line break should be inserted. Line breaks with a priority of zero will always be broken, while the maximum value will indicate that a line break at the token should never occur.



A block is formatted using the following algorithm:

1. The formatting algorithm makes an initial pass, breaking the block at priority 0 tokens. All line break defined with priority 0 (i.e., Always) in the **RPL Display Settings** dialog will be formatted in this initial pass. This pass always occurs.

2. Next the formatting algorithm checks the length of block. If the block will fit in the width of device, the formatting terminates. If the block will not fit the width of the device, the priority is incremented, and line breaks are made at all tokens defined with priority 1.

3. Repeat step 2 through all priorities until the block fits in the device.

If the block exceeds the screen width after all line break positions have been exhausted, the window will be made scrollable. If the block exceeds the printer's page width after all line break positions have been exhausted, the excess will be printed on adjacent pages.

To edit any number in the dialog, click on the number and then type in a new value. You can type in either "Always" or 0 for very high priority values. Type in "Never" or a large number (greater than about 40) for those items with low priority.

Example:

Given the line break settings:

Token	Break Before	Break After	Priority
THEN	no	yes	Always (0)
END IF	yes	yes	Always (0)
+	yes	no	1
AND	yes	no	2

The block being fitted to the box:

Before Formatting:

```
IF (SomeFunctionWithAVeryLongName() AND AnotherFunctionWithAVeryLongName) THEN Some-
ObjectSlotWithAVeryLongName + AnotherObjectSlotWithALongName END IF
```

After the initial pass (priority 0):

```
IF (SomeFunctionWithAVeryLongName() AND AnotherFunctionWithAVeryLongName) THEN
      SomeObjectSlotWithAVeryLongName + AnotherObjectSlotWithALongName
END IF
```

After the second pass (break on all priority 1 positions):

```
IF (SomeFunctionWithAVeryLongName() AND AnotherFunctionWithAVeryLongName) THEN
      SomeObjectSlotWithAVeryLongName
      + AnotherObjectSlotWithALongName
END IF
```

After the third pass (break on all priority 2 positions):

```
IF( SomeFunctionWithAVeryLongName()              ) THEN
  ( AND AnotherFunctionWithAVeryLongName()       )
      SomeObjectSlotWithAVeryLongName
      + AnotherObjectSlotWithALongName
END IF
```

Rule Fits the box. Stop.

Notice that with this naive algorithm, had the object slot names been small, they would have still been broken at the “+” symbol, since, the “+” had a higher priority (1) than the “AND” and the block did not yet fit the box.

Before Fomatting:

```
IF (SomeFunctionWithAVeryLongName() AND AnotherFunctionWithAVeryLongName) THEN Object
Slot + AnotherObjectSlot END IF
```

After the initial pass (priority 0):

```
IF (SomeFunctionWithAVeryLongName() AND AnotherFunctionWithAVeryLongName) THEN
      ObjectSlot + AnotherObjectSlot
END IF
```

After the second pass (break on all priority 1 positions):

```
IF (SomeFunctionWithAVeryLongName() AND AnotherFunctionWithAVeryLongName) THEN
      ObjectSlot
      + AnotherObjectSlot
END IF
```

After the third pass (break on all priority 2 positions):

```
IF( SomeFunctionWithAVeryLongName()          ) THEN
  ( AND AnotherFunctionWithAVeryLongName()   )
      ObjectSlot
      + AnotherObjectSlot
END IF
```

Rule Fits the box. Stop.

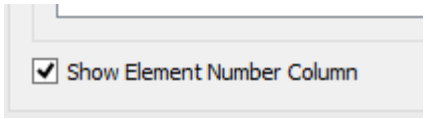
6.2.4 Element Numbers

RPL Expressions can be long and it is sometimes hard to tell where you are within the RPL expression. To help with this, you can enable to show **RPL Element Numbers** along the left hand side of the RPL frame. Notice in the screenshot below the column of numbers from 1 to 33

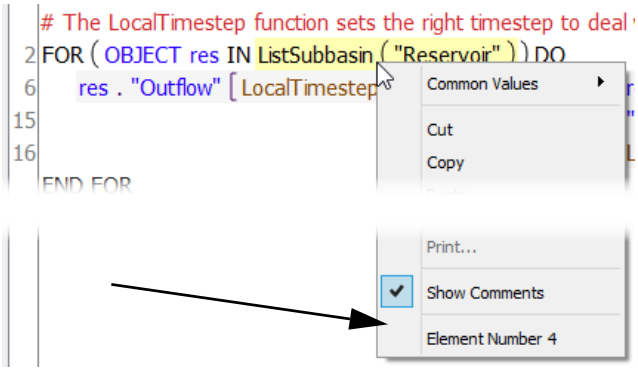
Element numbers are shown for the statements and expressions with which rules and functions are composed. As a result, they are neither continuous nor a constant increment, but are reflective of the number of elements on that line.

One can think of these like line numbers, but they are not line numbers as lines associated with a rule/goal/function vary with the formatting, which is adjusted to reflect the RPL display settings and the width of the display.

The display of Element Numbers is controlled by a check box in the **Set/Rule/Function/Goal ➥ RPL Display Settings** dialog. When display of the element numbers is enabled, a new column is shown along the left side of each RPL frame which displays, for each line, the number of the lowest numbered token beginning on that line.



In addition, when this feature is enabled and there is a selection in a RPL frame, the right-click context menu displays the Element Number of the selected element.



RPL External Documentation

7. RPL External Documentation

7.1 Overview

This document describes RiverWare's capability to link a RPL set to documentation that resides in a separate file viewable by commonly used viewing and editing programs.

RiverWare's RPL editors support an optional multiple-line text description, displayed in a plain-text editor box and is stored in the RPL set file. Often this text description is not sufficient to adequately describe a complicated rule or policy set. The purpose of a link to external documentation is to allow the user to develop more detailed documentation that will reside in a separate application like MS Word, PDF, or an HTML file. From the RPL editors, the user can click to go to the documentation file. Note, the external documentation support described in this document is distinct from this description text value.

A RPL set consists of rules (or user-defined accounting methods) and functions organized into groups. In this document, the term RPL **object** refers generically to a RPL set component of any sort, i.e., to the Set, Policy or Utility Group, Function, Rule, or Method.

The following is an overview of the capabilities:

- From a RPL dialog, the user is able to easily access external documentation for the RPL object associated with that dialog.
- The four supported types of documents are:
  - HTML,
  - MS Word,
  - PDF, and
  - plain Text
- The user is able to specify the applications for both viewing and editing each type of document
- Two modes of access are provided:
  - Edit - the user can view, create, and change the contents of the document.
  - View - the user can view but not change the contents of the document.
- The granularity of the documentation is flexible. Users can associate a separate document with each RPL object within a RPL set or they can document an entire RPL set with a single document.
- When using HTML, the application can open to the most relevant portion of the document. This applies when:
  - There is more than one RPL object described in the associated document, e.g., the documentation is a single file describing the entire ruleset, and,

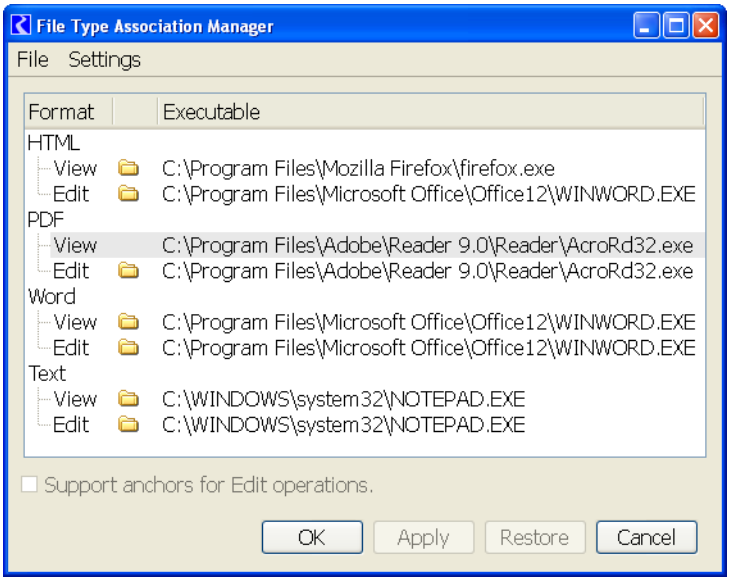
  - The user is using standard named anchors within the documentation to describe the sections that pertain to each RPL objects.
- The user can create an HTML template of the RPL set that contains the description field for each object. The user can then expand the documentation using an HTML editor.

The following sections describe the external document link feature. Configuration of this feature has two parts: first, the user must specify the application to be used to edit and/or view one or more types of files; second, the user specifies how the document is structured by associating RPL objects with external files.

7.2 File Association

The RiverWare **File Type Association Manager** is used to specify the executable programs used for viewing and editing documents of the four supported Format Types (HTML, PDF, Word, or Text). The file paths are stored with the user's settings. It is accessible from:

- The RPL editor **View ➥ External Documentation ➥ File Type Associations** menu
- The workspace **Utilities ➥ File Type Associations** menu
- The **Utilities ➥ File Type Associations Manager** from the **Configure External Document Reference** dialog.



Each file format has a distinct Edit and View mode program association. This allows the user to edit with one program and view with another. Only the document formats that the user wishes to use need to be specified. The others can be left as the default or left blank.

Executable file paths can be edited (including pasting from the system clipboard) by double clicking an item within the **Executable** column or can be picked with a File Selector by clicking the folder icon. All of the executable file paths can be set to the default executable associated with the user's account using the **Settings ➥ Set Default Executables** menu operation. Changes can either be saved (by clicking OK or Apply) or dropped (by clicking Restore or Cancel).

The PDF View item is set to the path associated with PDF files, is non-editable, and is shown with a grey background. This is the same path used to locate the pdf reader for RiverWare Help and other purposes. Click **HERE** to search for more information on setting this file type association.

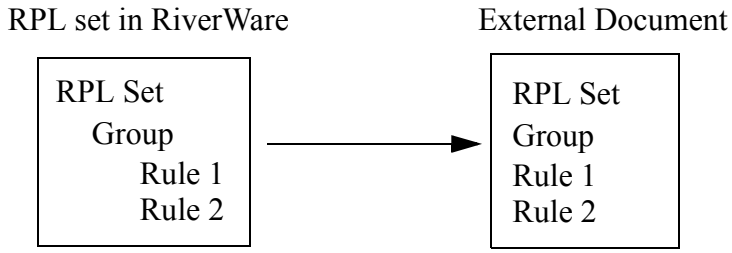
Currently, the **Support anchors for Edit operations** check box is not operable. When opening a RPL Object External Document for Editing, the anchor text (used to automatically scroll to a particular

section within the document) is only supported for HTML. For all types, the name of the RPL Object name text is copied to the system clipboard and can be used to paste into the "Search" function within the editor or viewer program.

7.3 Document Structure

This section provides an overview of the possible document configurations including some advantages and disadvantages to both. Following are setup configurations that have been envisioned:

One document that describes the entire set:

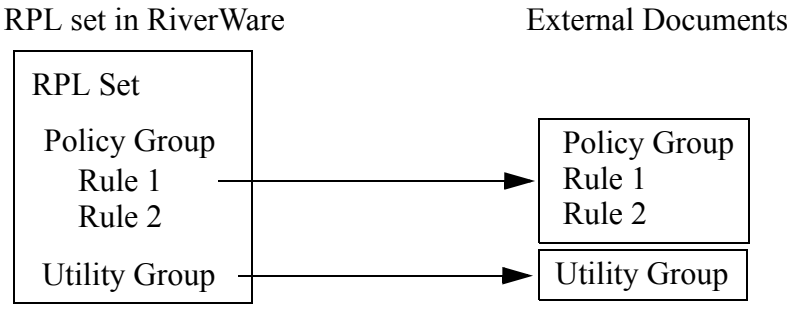


- Advantages: one document is typically easier to maintain and share. Also, the user only needs to configure the external link for the overall RPL set, not for each individual RPL object within the set.
- Disadvantages: unless the viewing document is HTML, there is no way for the user to automatically open the document to the correct location. For example, if the user clicks to view a rule's document (say in PDF) they must still search for that rule within the PDF document. Also, multiple model users have to coordinate if they wish to develop documentation simultaneously.

One document for each RPL object in the set.

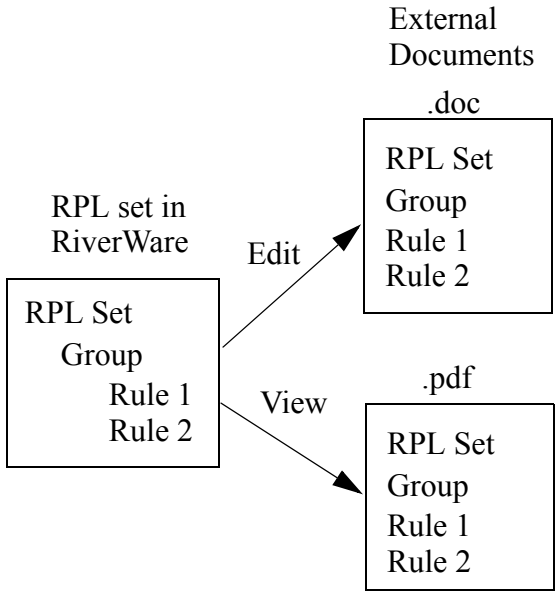
- Advantages: one document for each RPL object allows the user to easily view the external document specific to that object. For example, the user clicks on the view button for a rule and the document specific to that rule opens. Multiple users can more easily develop documentation simultaneously.
- Disadvantage: It is more time consuming to configure and maintain. When sharing, there are many more documents to pass around. If there are a large number of RPL objects, there will be a large number of documents.

Multiple documents with one document for each policy group and utility group. This structure is a hybrid of the two above and has some of the advantages and disadvantages of both. This structure is useful because exceptions to the overall scheme are natural. For example, most of the set could be described by one document, while special objects could be described in a document of their own.



Combinations of the above could be envisioned. The choice of the structure to use depends on the format and the end use.

Separate View and Edit Documents: External documentation for a single object might also be found in two different files, one for editing and one for viewing. For example, documentation might be created in MS Word, saved in Word format for the purposes of future edits, and also saved to a PDF file for the purpose of viewing. Thus the utility allows the user to specify two locations, one for the purposes of viewing and another for editing. The following diagram shows an example as two separate documents, one for viewing, one for editing.



When the documentation for an object is split across many files, then it will often be convenient to the users for all the files to be located in the same directory. This organization is supported without requiring the user to provide redundant information (i.e., repeat the full path to the documentation for each object in the ruleset). At the same time, there is enough flexibility so as not to impose a specific organization on the documentation, either that there be a one-to-one correspondence between ruleset objects and documents or that multiple documents appear in the same directory.

7.4 Configuration and User Interface

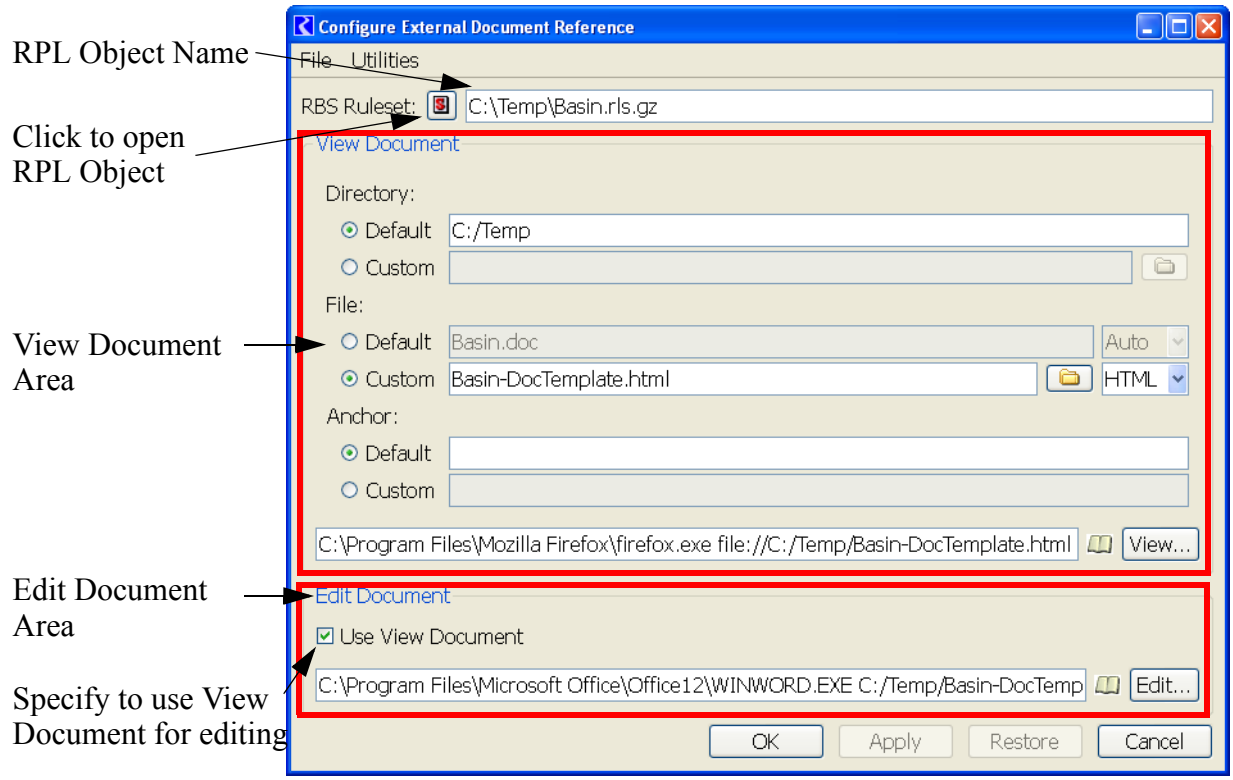
The document structure is specified in the RPL set by supplying path names to a directory and then specifying file names within that directory. Further, because RPL objects are organized in a hierarchical organization (RPL sets contain policy and utility groups, policy groups contain rules and functions, utility groups contain functions) the hierarchical organization enables RPL objects to share information

between objects. When configuration information is provided for an object, the containing objects inherit the configuration from their parent as their default configuration. The user can change the containing objects' configuration as necessary. Thus to specify a single directory in which all the documentation for a RPL set is contained, the user need only specify that value as the directory associated with the set; all groups, functions, and rules in that set will inherit this as their documentation directory.

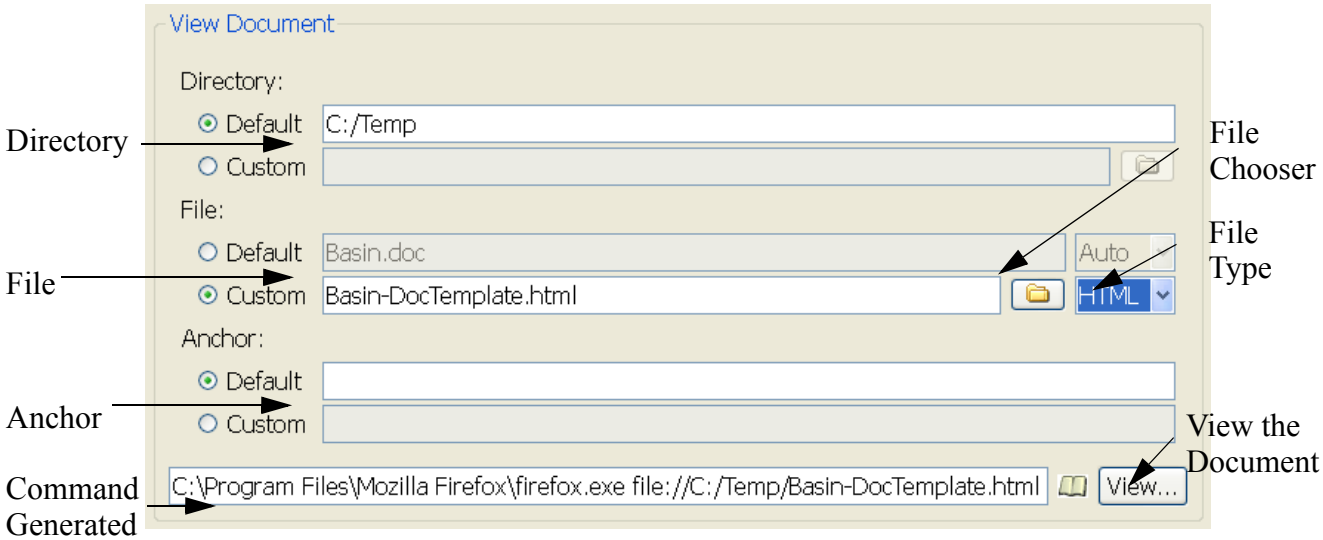
In addition to providing flexibility with respect to the organization of the documentation, this approach has the advantage that when documentation is moved (say from the modeler's file system to a stakeholder's file system), the ruleset will typically only need to be updated in one place to reflect this change. Environment variables can be used to further simplify moving RPL set documentation between machines. With the use of environment variables, the base directory of all RPL set documents within a users configuration can be set without any changes to the document link configuration within RiverWare. The environment variable substitution mechanism uses unix-style syntax (even on windows): any environment variable name (case sensitive with only alphanumeric characters and underscores) preceded with a dollar sign "$" is translated to the environment variable defined in the user session, outside of RiverWare. For example, an environment variable name "ALLUVIAL_RPL_DOC" is defined as "C:\Projects\docs", then within the RPL external documentation link, "$ALLUVIAL_RPL_DOC\forecastDraft\" will refer to C:\Projects\docs\forecastDraft. This is specifically designed for portability of external RPL documentation across machines and platforms. Slashes can be forward or back slashes and redundant slashes are condensed.

In this documentation and in the user interface, we refer to the two parts of the location specification as the **Directory** and the **File name**. There is nothing that requires that the two parts correspond to a directory and file name in the local file system. Another possibility is that the two parts together specify the location of a document as a URL (Uniform Resource Locator, a type of URI). In this case, the first part to be something like "http://www.WaterU.edu/Models", which is not strictly speak a directory. Note also that the file name specification could be an absolute path specified relative to the directory specification, e.g., "MyFunctions/FunctionA.htm".

The configuration dialog is available from the RPL editor **View ➥ External Documentation ➥ Configure** menu. The following screenshot shows the dialog for the RPL Set:



In this dialog, the RPL object is listed at the top and there is a button to take you to the that RPL object. Then, there are two areas, one for the **View Document** and one for the **Edit Document**. If the **Use View Document** check box in the **Edit Document** area is checked, the remainder of the **Edit Document** area is hidden. This toggle specifies that the **View** and **Edit** documents are the same. The process of specifying a View and Edit document is the same, so it is only described for the View Document. The View Document contains the following:



For the **Directory**, **File**, and **Anchor** (when using HTML), the user can specify to use the listed **Default** or click to use a **Custom** configuration. When **Custom** is first clicked, the **Default** value is copied down from the **Default** field. For the **Custom** configurations, the user can either type in a value or use the File Chooser button to specify. For the **File**, the user can also specify the file type or use the auto selection. Anchors are currently only supported for HTML files and are described **HERE (Section 7.5.1)**

The default **Directory** is the directory containing the RPL set itself. The default **File** is the name of the RPL set with a modified filename extension. It is typically ".doc" or ".docx", but will instead be ".html" if a file with the resulting name at the path actually exists.

For RPL objects (e.g. a group or rule) within a set, the default directory and file names are inherited from the containing RPL object, and the default anchor (when HTML is used) is based on the RPL object name. Note, spaces and most punctuation is removed. Note that the default anchors for all the objects within a RPL set are reported in a table in the generated HTML template file (described **HERE (Section 7.5.1)**).

At the bottom of the View area is the resulting command that is generated and will be passed to the operating system to start the application. This is the result of the configuration that has been defined. It is the chosen application followed by the command used by that application to open the specified file. Clicking the **View** (or **Edit)** button will execute the command.

As noted above, the Edit Document area is similar to the View Document area. At the bottom of the window are OK (apply and close), Apply, Restore to previously applied values, and Cancel buttons.

Once applied, the configuration settings are propagated to the default value of contained RPL Objects. Thus, the user only needs to specify as much information as necessary to fully configure the document. If the user is using the layout **HERE (One document that describes the entire set)**, he/she only needs to configure the RPL set and all rules, groups and methods will be configured correctly. If the user is using the layout **HERE (One document for each RPL object in the set.)**, he/she needs to make sure that each RPL object is configured correctly but using a consistent naming convention can be used to simplify the configuration. If there are exceptions to the naming convention, the user would need to configure the RPL objects that have the exception.

7.5 Viewing and Editing

This section first describes viewing and editing documents in general, then describes peculiarities and specifics for each type of supported document. Once configured, viewing can be done directly from the RPL Object Editor using one of the following:

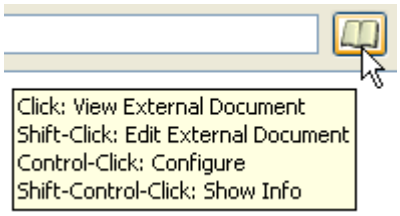
- Click on the external documentation icon . It is on the top of the dialog by the object name Note, this icon is only shown if a document actually exists at the configured file path (including the default path if that is a correct configuration). Note, if the path leads to a web address, i.e. http://, the icon is always shown. RiverWare does not attempt to verify the existence of such pages on the web.
- Use the **View ➥ External Documentation ➥ View Document...** menu.

Similarly, editing can be done directly from the RPL object using one of the following:

- Shift-Click on the external documentation icon
- Use the **View ➥ External Documentation ➥ Edit Document...** menu.

Note: Note, mousing over the External Documentation icon provides tool tips on the use of this button as follows:

Click: View External Document
Shift-Click: Edit External Document
Control-Click : Configure Document
Shift-Control-Click: Show Info



Although Anchors (described **HERE (Anchors)**) are only supported when viewing HTML documents, the actual name of the RPL Object is copied to the system clipboard. This can then be used to paste into the "Search" function within the editor or viewer program. For example, if viewing a PDF, the user can click the external documentation icon for a function, then click Ctrl+F to bring up the search field/ dialog, then type Ctrl+V to paste the name of the function in the box.

The following sections describe the specific application recommended (and not recommended) to view and edit each type of document and then the mechanism to configure the application that is associated with a document type.

- Hyper Text Markup Language (.html)
- MS Word (.doc or .docx)
- Portable Document Format (.pdf)
- Text (.txt or other)

7.5.1 HTML

Hyper Text Markup Language (HTML) is a common format used by web pages. There are numerous tools to develop and view these types of files.

Viewing: HTML can be viewed using any common browser.

Editing: HTML can be edited using web development tools such as Adobe (MacroMedia) Dreamweaver. MS Word can also be used edit HTML files.

Anchors: When a document describes more than one object and the user requests to view the documentation for a particular object, the program, if configured correctly, can open to the most relevant location within the document. In order to accomplish this, there needs to be some sort of connection between locations within the document and objects in the set. In this document we will refer to these as **anchors**. In HTML, locations within a document are defined using named anchors.

RiverWare uses anchors to provide a mechanism for navigating to a particular location within a document. For a given object that inherits its document location from a parent object, by default

RiverWare assumes the existences of an anchor for this object whose text is based on the object's name. In particular, we assume that the anchor's label is the object's name, with illegal characters replaced with legal ones. The user may override the default value and edit it or specify that no anchor exists.

Typically, the user is responsible for inserting anchors into the external documentation (see **HERE (Generating HTML Template File)** for a tool to help create a document with anchors). RiverWare doesn't attempt to inspect the documentation, so care will be required on the part of the user to maintain anchors correctly. As with other sorts of references from an external document to the details of a ruleset, the potential for inconsistencies is large. For example, if the user changes the name of a function in a ruleset, then the name of the anchor for that function in the documentation would need to be changed as well. In addition, access programs will often not normally make anchor text visible, making it more challenging for the user to verify that the anchor labels are correct. To support the management of anchors, RiverWare supports copy/paste between fields containing object names and external programs.

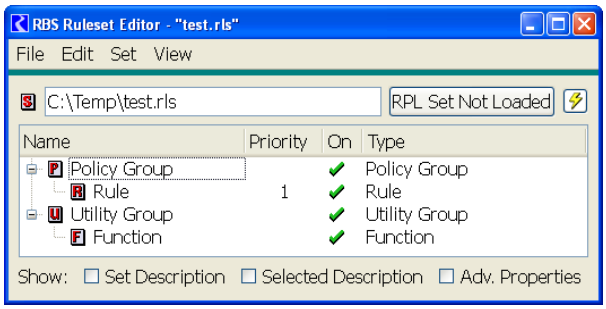
For example, if the external document points to the following HTML file:

http://www.WaterU.edu/Models/Functions.html

and a function named "Set Outputs" is given the default anchor "SetOutputs". RiverWare will then provide the following to the access program:

http://www.WaterU.edu/Models/Functions.html#SetOutputs

Generating HTML Template File: The External Documentation feature provides a utility to generate a template of the RPL objects in the set. This can be used as a starting point in which the user can fill in the missing pieces. It allows someone not too familiar with HTML to quickly generate a working document and all the associated anchors and formatting.



From the RPL set's **Configure External Document Reference**, the user can select **Utilities ➥ Generating HTML Template File** to generate an HTML file that contains all of the RPL objects in the set. The file is created in the directory specified in the View Document configuration. It is given the name specified in the File field if it is an HTML file or the default name with a "-DocTemplate.html" appended if the File field contains a non-HTML file.

The HTML document also contains a table of contents that lists the type of RPL object, the name of that object (and a link to the referenced section) and the anchor that it uses. Shown in the body of the file is the RPL object's name and any description (i.e. in the **View ➥ Show Description** in RPL editor) that existed in the RPL object at the time the template was created. Once the template is created, it can be viewed/edited using the view/edit button. The user can then fill in pieces as necessary and save the resulting file. Then there should be no need to re-generate this template again.

An example from a one group with one rule and one function is shown. The text came from the description field in each object.

Template: C:/Temp/test-DocTemplate.html
RPL Set: C:\Temp\test.rls
Generated: 12:29 August 27, 2010

Type	Name	Anchor
RPL Set	test	test
Policy Group	Policy Group1	PolicyGroup1
Rule	Rule1	Rule1
RPL Group	Utility Group1	UtilityGroup1
User-defined Function	Function1	Function1

RPL Set: test

Full name: C:\Temp\test.rls

This is the RPL Set Description

Policy Group: Policy Group1

Policy Group1 is used for ...

Rule: Rule1

Rule 1 does ...

RPL Group: Utility Group1

Utility Group1 contains functions for ...

User-defined Function: Function1

Function1 has the following arguments... It does ...

7.5.2 MS WORD

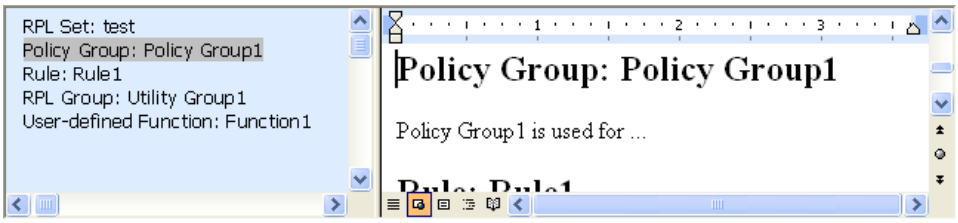
MS WORD (.doc or .docx) files are commonly developed using MS Word but more recently can be viewed (and even edited) using many browsers

Viewing: Traditionally, .doc files could only be opened using MS Word. Now, these can be viewed with certain browsers including Internet Explorer. Note, choosing to view a .doc file with MS Word and Explorer do not automatically make the document non-editable. The file system must be used if the user wishes to make the document read-only.

Editing: Again, MS Word is typically used to edit .doc files. The latest versions of Internet Explorer also has capabilities to edit these types of files.

For both editing and viewing Word and HTML files, in MS Word has feature called the Document Map that is useful to navigate through documents. Similar to bookmarks in a PDF file, a pane is added to the document that allows the user to navigate to defined headings in the document.

To access this feature, use the **View ➥ Document Map** in MS Word. A sample is shown to the right for the HTML document shown **HERE (Generating HTML Template File)**.



7.5.3 PDF

Adobe's Portable Document Format (PDF) has become the standard document format when sharing read-only versions of documents. These documents can be viewed by the free Adobe Acrobat Reader and more recently by many browsers.

Viewing: Viewing of PDF files is accomplished using a PDF Reader. This is the same format and program that RiverWare's Help file uses. This is not editable here, but you can change the default PDF reader in Windows by changing the file type association. Click **HERE** for more information.

Editing: Although limited editing of PDF can be done using the full version of Adobe Acrobat, this is not a supported editing tool. Otherwise, PDFs are read-only and cannot be edited. They are typically created from some other document editing application like MS Word or Adobe FrameMaker. We will not discuss editing PDF further.

7.5.4 Text

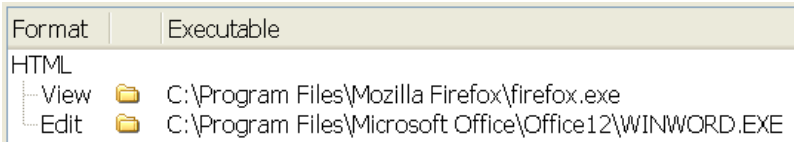
Text files can be viewed (and edited) by any text editor and many web browsers. No further information is provided.

7.6 Use Example

Following is an imaginary use example to show the process. Perhaps, we are a water agency and have a complex ruleset. We wish to share this ruleset and documentation with stakeholders in the basin but we do not wish to let them have an easily editable copy of this document. Also, we already have some descriptions entered in the ruleset but no other documentation. In this example, we wish to use our favorite web browser to view the document and MS Word to edit the document. Once the document is complete, we will post it to a web page so that stakeholders can see it. As a result, we will use HTML as the type of document.

We set this up as follows:

1. Define the file associations: We will use FireFox to view the document and MS Word to edit the document. We set up the File Type Association Manager **HERE (Section 7.2)** as shown:



Because we are using HTML, we can use anchors, **HERE (Anchors)**, to automatically open the HTML file to the right place. As a result, we only need one document for the entire RPL set. Thus we are using the structure described **HERE (One document that describes the entire set)**. We do want to have separate documents for viewing and editing. While making changes to the ruleset and hence the "Edit" document, we will still have an official "View" document that stakeholders can use. A diagram of this structure is shown **HERE (Separate View and Edit Documents)**. Hence, we will need to configure only the top level of the RPL set.

2. From the RPL set, we open the configuration menu using the **View ➥ External Documentation ➥ Configure** menu.

Because we already have some descriptions in our RPL set and we wish to use HTML, we will use the automatically generated template as a starting point. First lets specify a location to put the file, say C:\Temp\docs

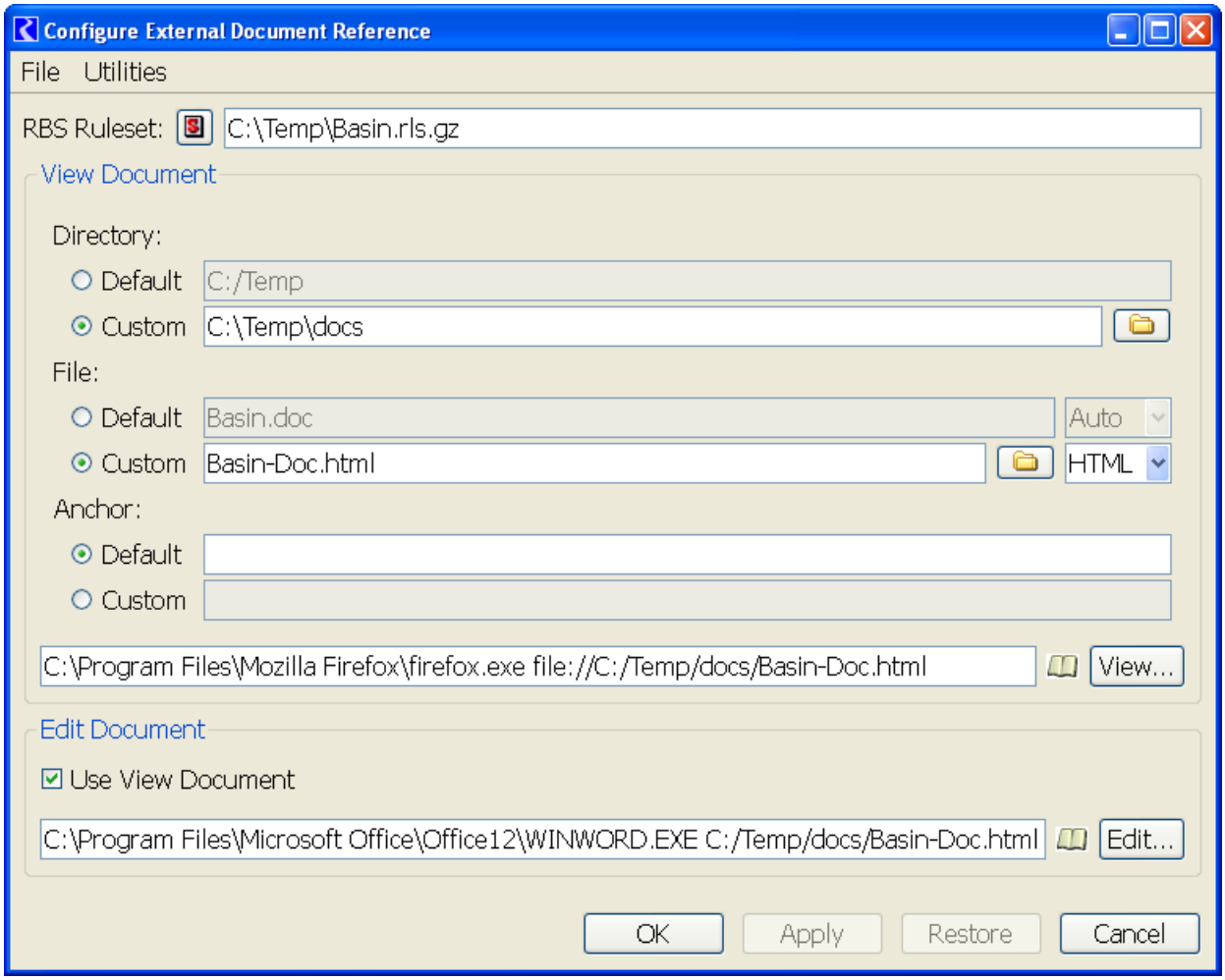
3. In the View area, click the **Custom** toggle for the Directory and enter C:\Temp\docs

Now we will generate the template:

4. In the Configure dialog we choose **Utilities ➥ Generating HTML Template File.** It gives us a warning that a file named C:/Temp/docs/Basin-DocTemplate.html will be saved. Note our RPL set is named Basin.rls.gz.

5. We do not like the name of this file (and to avoid overwriting our completed external document with an inadvertently generated template file in the future), we change both the file name and the name in the configuration. We change the file to Basin-Doc.html using windows explorer. In the File area of the configuration, we change the name to Basin-Doc.html and click **Apply** when complete.

For now we will leave the View document in the C:\Temp\docs folder. Once we are complete we will move this to a web server. Our configuration is shown in the following screenshot.



6. Now we click on the **View** button and see the template file in Firefox.

We see that we have a pseudo table of contents with links, then each object is listed with the text that came from the description.

7. We click the **Edit** button and see the file open in Word.

8. We edit the document in Word by adding text, pictures, graphs, flowcharts, etc. We make sure to save it as the same name. We can edit the formats of the headings and text. When editing the headings, we make sure to use the **Format ➥ Styles and Formatting** menu in Work to change all instances at once.

9. Anytime we wish to see how this document looks in the browser. We can save the file from Word and switch back to the RPL set and click on the external documentation icon . This takes us directly to that RPL Object's section in the document.

10. When editing is complete, we copy the final HTML to a web server.

11. Then we go back to the configuration menu for the RPL Set and change the View document. First we want to click that the Edit document is now different, so we toggle off the **Use View Document** option. We make sure the Edit document area is now correct.

12. Now we want to change the View area to point the Custom Directory to a web server: http://cadswes2.colorado.edu/docs/

The configuration for this setup is shown in the following figure.

If we click on the external documentation icon for any RPL object, it will take us to this web page and automatically scroll to the section related to that object. We now share our model and ruleset with our stakeholders.