RIVERWARE DEMONSTRATION TUTORIAL

Date Modified: January 9, 2017

This RiverWare informational tutorial introduces you to the capabilities of RiverWare as an advanced water resource modeling tool and consists of three sections, **Simulation in RiverWare**, **Rulebased Simulation**, and **Multiple Run Management**. These tutorials are independent documents but later topics do build on skills you gain from earlier exercises, so if you have not gone through the earlier topics or have not used RiverWare recently it is recommended that you start with the tutorial on **Simulation in RiverWare**.

Before you start... You must have RiverWare 7.0.1 or later installed on your computer and have a valid license file. For more information about downloading RiverWare, visit RiverWare.org. To acquire a demonstration license, please email Installation Support (installsupport@colorado.edu).

You must have downloaded the files associated with this tutorial and moved them to a reasonable folder like `C:/temp`. The following files are used in this tutorial:

- MuddyBartlettBasin_RBS_Start.mdl.gz - Starting MODEL
- MuddyBartlett_Info_RBSstart.rls.gz - Starting RULESET
- MuddyBartlett_Info_RBSfinal.rls.gz (this is just the final resulting ruleset of this module)

2. Rulebased Simulation

This informational tutorial introduces you to Rulebased Simulation in RiverWare. You will be introduced to:

- Running a Rulebased Simulation
- Examining functions and rules
- Writing a rule using the RPL Editor and Palette
- Obtaining more information on a run using the diagnostics and debugger.

To get started let's open the model in RiverWare:

☞ Double-click the RiverWare icon on the desktop.

☞ Select **File ➥ Open Model...** from the main workspace menu bar or click on the Open Model File toolbar button.

☞ Open the file `MuddyBartlettBasin_RBS_Start.mdl.gz`

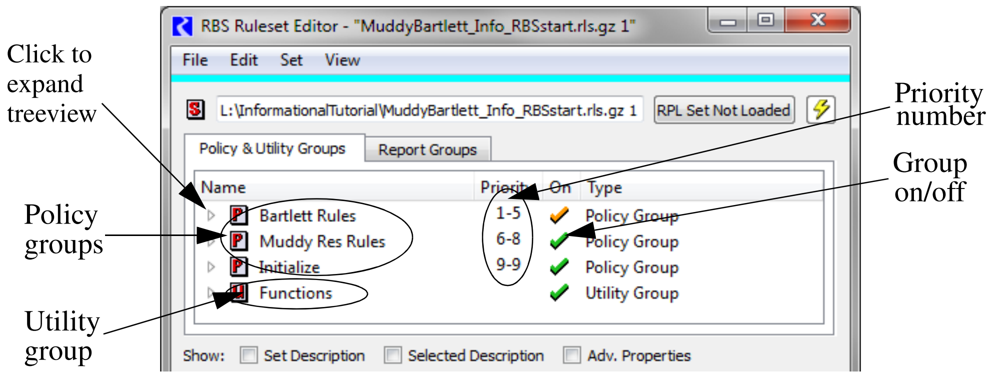
2.1 Running a Rulebased Simulation

Rulebased simulation is an extension of basic Simulation in which some of the data to solve an under-determined model is provided by a set of user-defined rules. Rule execution alternates with dispatching to simulate the effects of the rules in the model.

The policy for the basin is in the Ruleset.

☞ In the **Policy** menu of the main workspace, select **Ruleset ➥ Open...**

☞ In the file chooser, select the following ruleset: `MuddyBartlett_Info_RBSstart.rls.gz`


Click to expand treeview
Policy groups
Utility group
Priority number
Group on/off
RBS Ruleset Editor - "MuddyBartlett_Info_RBSstart.rls.gz 1"
File Edit Set View
L:\InformationalTutorial\MuddyBartlett_Info_RBSstart.rls.gz 1
RPL Set Not Loaded
Policy & Utility Groups
Report Groups
Show: Set Description Selected Description Adv. Properties

Name	Priority	On	Type
Bartlett Rules	1-5	✓	Policy Group
Muddy Res Rules	6-8	✓	Policy Group
Initialize	9-9	✓	Policy Group
Functions		✓	Utility Group

The Ruleset Editor initially displays three **Policy Groups** and one **Utility Group**. A Policy Group is a folder for holding rules; Utility Groups contain functions.

☞ Click on the triangle to the left of each of the groups to expand the treeview.

Now you can see all individual rules within each Policy Group. Note that the check mark next to Bartlett Rules is orange because the priority 1 rule, **Bartlett Flood Control***, is turned off.*

For this simple model, there are nine rules (indicated by a red **R** symbol) and eight functions (indicated by the **F** symbol).

Before running, you must tell RiverWare that this is the ruleset you wish to use, i.e. Loading the ruleset. Loading a ruleset validates it for unspecified expressions, illegal object and slot names, conflicting expression types, and syntax errors.

☞ Load your ruleset by clicking on the **RPL Set Not Loaded** button. This validates the ruleset and loads it into the model.

☞ Run the simulation by selecting in the main workspace.

Notice that the controller is set to Rulebased Simulation

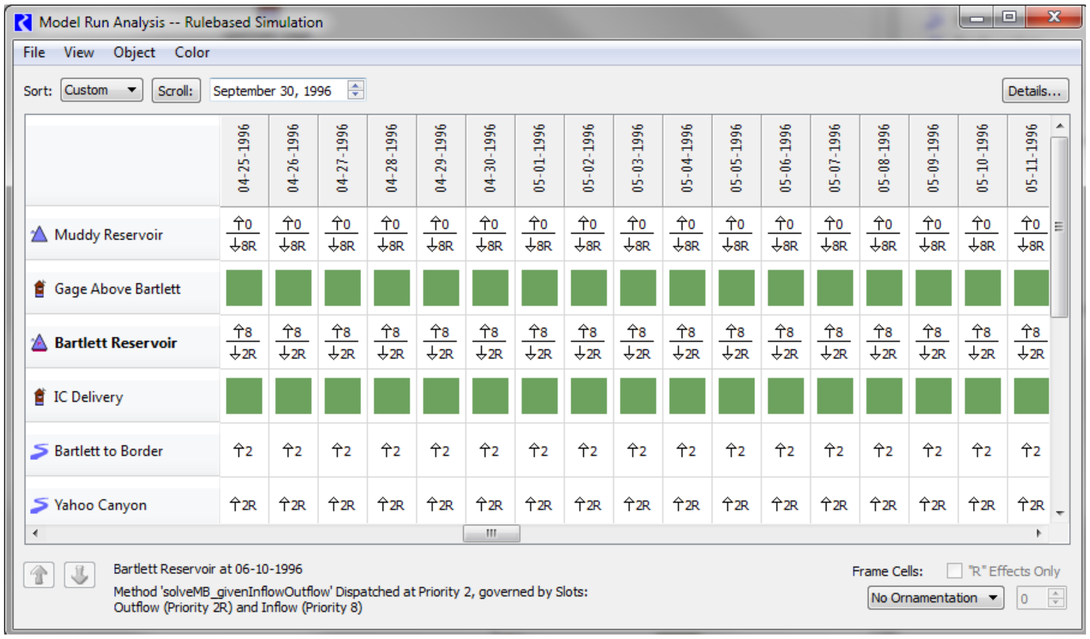
☞ Click **Start**.

☞ When the run is finished, close the **Run Status** and **Run Control** dialogs.

On each timestep, each rule is executed, then the objects solve as necessary to simulate the effects of rules setting values. Rule #9, the lowest priority, is executed first, while rule #1, the highest priority, fires last. Now let's briefly examine the results of the run and see how the rules set values in the model:

☞ Open the **Model Run Analysis** dialog by clicking in the main workspace.

For a more in depth explanation of the Model Run Analysis dialog, see pages 14-16 of the Introductory Information Tutorial


Model Run Analysis -- Rulebased Simulation

File View Object Color

Sort: Custom Scroll: September 30, 1996 Details...

	04-25-1996	04-26-1996	04-27-1996	04-28-1996	04-29-1996	04-30-1996	05-01-1996	05-02-1996	05-03-1996	05-04-1996	05-05-1996	05-06-1996	05-07-1996	05-08-1996	05-09-1996	05-10-1996	05-11-1996
Muddy Reservoir	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R
Gage Above Bartlett	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Bartlett Reservoir	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R
IC Delivery	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Bartlett to Border	↑2	↑2	↑2	↑2	↑2	↑2	↑2	↑2	↑2	↑2	↑2	↑2	↑2	↑2	↑2	↑2	↑2
Yahoo Canyon	↑2R	↑2R	↑2R	↑2R	↑2R	↑2R	↑2R	↑2R	↑2R	↑2R	↑2R	↑2R	↑2R	↑2R	↑2R	↑2R	↑2R

Bartlett Reservoir at 06-10-1996

Method 'solveMB_givenInflowOutflow' Dispatched at Priority 2, governed by Slots: Outflow (Priority 2R) and Inflow (Priority 8)

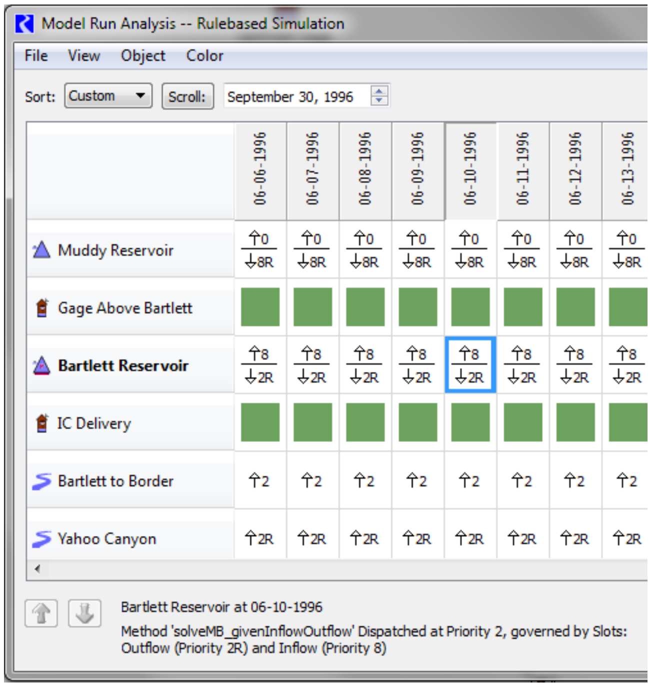
Frame Cells: "R" Effects Only

No Ornamentation 0

☞ Scroll through the timesteps horizontally and notice the reservoirs and reaches were dispatched in accordance with various rules in the model ruleset.

☞ Find the dispatch cell for **Bartlett Reservoir** on June 10 (**06-10-1996**) and click on it.

Text at the bottom of the window gives details about dispatch method and the rules that set the values on the dispatch slots.



Sort: Custom Scroll: September 30, 1996

	06-06-1996	06-07-1996	06-08-1996	06-09-1996	06-10-1996	06-11-1996	06-12-1996	06-13-1996
Muddy Reservoir	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R	↑0 ↓8R
Gage Above Bartlett	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Bartlett Reservoir	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R	↑8 ↓2R
IC Delivery	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Bartlett to Border	↑2	↑2	↑2	↑2	↑2	↑2	↑2	↑2
Yahoo Canyon	↑2R	↑2R	↑2R	↑2R	↑2R	↑2R	↑2R	↑2R

Bartlett Reservoir at 06-10-1996

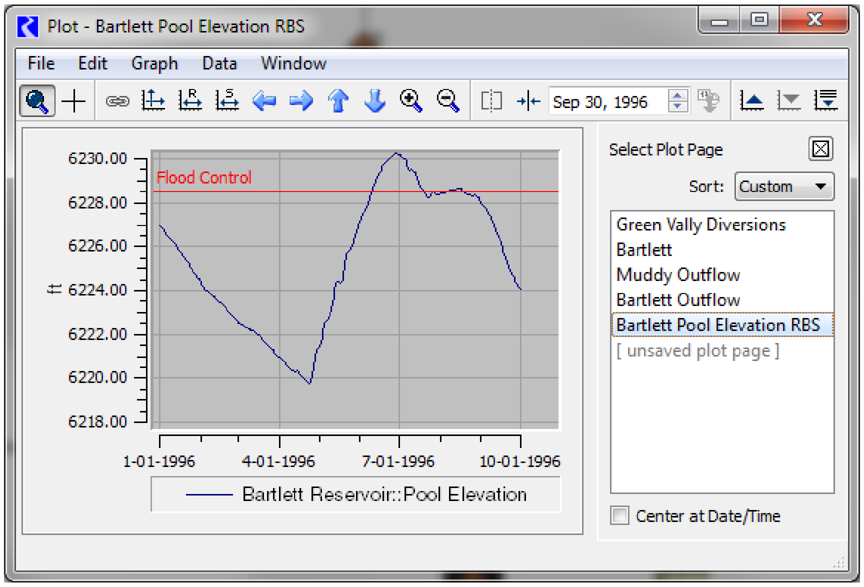
Method 'solveMB_givenInflowOutflow' Dispatched at Priority 2, governed by Slots: Outflow (Priority 2R) and Inflow (Priority 8)

↑8 ↓2R The Inflow was set by resulting output of rule 8- **Green Valley Diversion** requirements, the Outflow by rule 2- **IC Delivery** requirements. You can click on **View ➥ Grid Cell Legend** for more information.

In this way, you can determine the policy in effect for each timestep and each object in the run.

☞ Close the Model Run Analysis dialog.

☞ Open the plot dialog by clicking P in the main workspace and click the **Bartlett Pool Elevation RBS** plot to open it.



Green Vally Diversions
Bartlett
Muddy Outflow
Bartlett Outflow
Bartlett Pool Elevation RBS
[unsaved plot page]

If you remember, we saw that the Flood Control rule is turned off. Therefore there is no logic or policy in effect to keep the Bartlett Pool Elevation from exceeding maximum prescribed height. We see this illustrated in the plot in July 1996. In the next section, we'll turn on the rule and see its effect.

☞ Close the plot dialog.

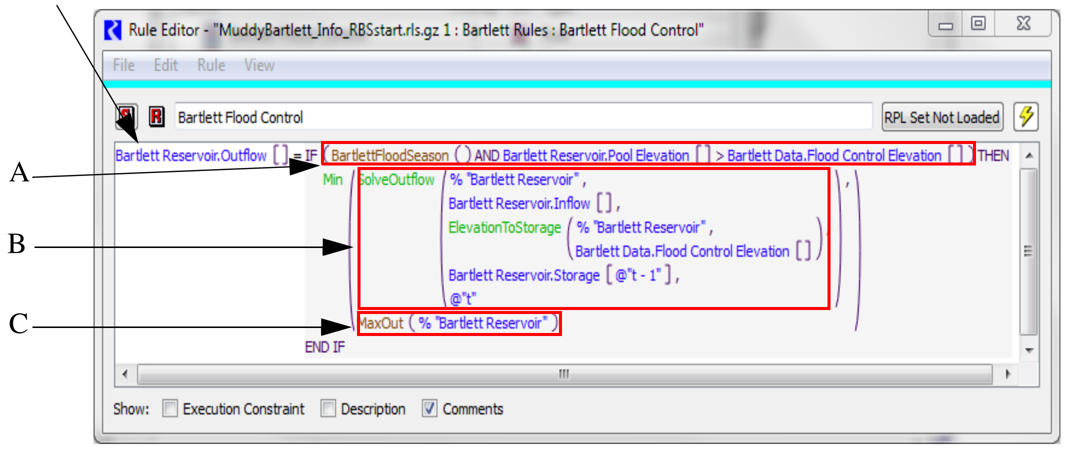
2.2 Examining functions and a rule

The Ruleset contains the policy of your basin. You write the rules and then give each one a priority starting with 1 up through the number of rules you have.

Rules are set up using logic in the RiverWare Policy Language (RPL). You can make use of predefined as well as user-defined functions to build up your logic and make it understandable by others. Let's start by walking through the logic of a rule:

☞ Open the **Bartlett Flood Control** rule by double clicking on its name in the **Ruleset Editor**.

Set Bartlett Reservoir.Outflow



```
Bartlett Reservoir.Outflow [] = IF ( BartlettFloodSeason ( ) AND Bartlett Reservoir.Pool Elevation [] > Bartlett Data.Flood Control Elevation [] ) THEN
    Min ( SolveOutflow ( % "Bartlett Reservoir" ,
                         Bartlett Reservoir.Inflow [] ,
                         ElevationToStorage ( % "Bartlett Reservoir" ,
                                              Bartlett Data.Flood Control Elevation [] ) ,
                         Bartlett Reservoir.Storage [ @"t - 1" ] ,
                         @"t" ) ,
          MaxOut ( % "Bartlett Reservoir" ) )
END IF
```

The rule contains logic (IF, THEN), slot names (e.g. Bartlett.Pool Elevation), and functions, both user defined and predened functions. In general, it says:

Object.Slot [ ]= IF (A) THEN

Minimum (B, C)

END IF

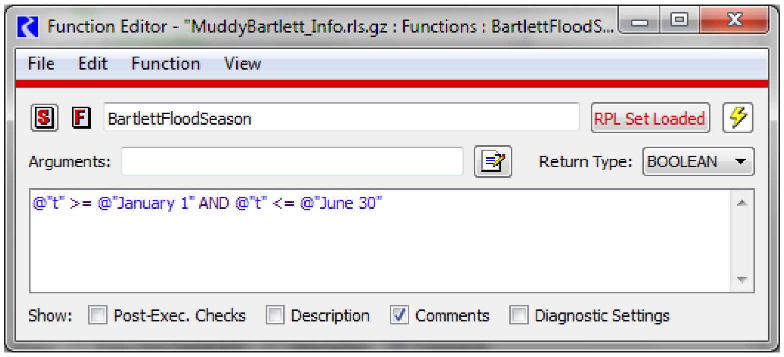
This rule is setting the value of the **Barlett Reservoir Outflow** slot at the current timestep. This rule is executed at each timestep, so the logic is written with that assumption.

We'll go through each item A, B, and C to describe the logic:

A. If it is **BartlettFloodSeason()** ***AND*** the current **Bartlett.Pool Elevation** is above the **Bartlett Data.Flood Control Elevation**,

This expression is used to check the state of the system. We only want this policy in effect during specific conditions. Thus, we are defining the logic for which we want this policy. The first item BartlettFloodSeason is a User-defined function, which allows you to encapsulate an expression and give it a meaningful name. You can then call this function from any other rule or function. This allows you to simplify and modularize your policy.

☞ Open the **BartlettFloodSeason** function by double clicking on the function name in the rule.


Function Editor - "MuddyBartlett_Info.rls.gz : Functions : BartlettFloodS...
File Edit Function View
RPL Set Loaded

BartlettFloodSeason

Arguments:

Return Type: BOOLEAN

```
@"t" >= @"January 1" AND @"t" <= @"June 30"
```

Show: [ ] Post-Exec. Checks [ ] Description [✓] Comments [ ] Diagnostic Settings

This is a function created to define the time of year when there could potentially be flooding. It evaluates to of either True or False. The statement asks whether the current time (@ "t") is

- at least January 1st (>= @ "January 1") AND
- no later than June 30th (<= @ "June 30").

☞ Close the **Function Editor** window.

Now let's look at the two values to which the rule could evaluate: B or C. The rule computes the minimum of these two.

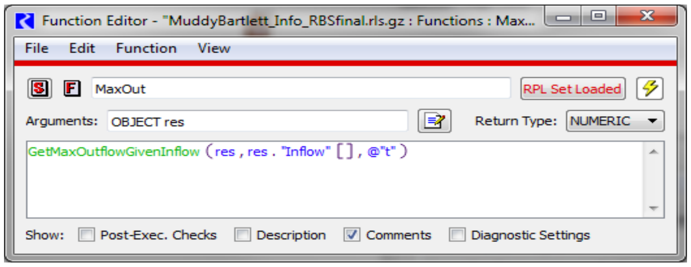
B. the **Outflow** to lower the **Pool Elevation** to the **Flood Control Elevation**

The **SolveOutflow** function (in green) is a predefined RiverWare function that performs a mass balance to compute a reservoir's outflow given its inflow, its previous storage, and the storage specified at the current timestep (in this case determined by the **ElevationToStorage** function).

For further information about this or any predefined function, see the RiverWare Help documentation, which can be accessed by clicking ? and choosing RPL Predefined Functions from the menu.

C. the maximum outflow allowed by the physical limitations of the reservoir's structures.

☞ Open the **MaxOut** function.


Function Editor - "MuddyBartlett_Info_RBSfinal.rls.gz : Functions : Max...
File Edit Function View
RPL Set Loaded

MaxOut

Arguments: OBJECT res

Return Type: NUMERIC

```
GetMaxOutflowGivenInflow ( res , res . "Inflow" [ ] , @"t" )
```

Show: [ ] Post-Exec. Checks [ ] Description [✓] Comments [ ] Diagnostic Settings

This function computes a number (it's return type is NUMERIC) by evaluating a predefined RiverWare function, **GetMaxOutflowGivenInflow**, at the current timestep when a reservoir, **res** is passed in as an argument.

The **GetMaxOutflowGivenInflow** is another predefined function that computes the maximum outflow from a reservoir considering the previous timestep storage, inflow, all sinks, side flows, and sources, and the physical characteristics of the outlet structures. In the case of a storage reservoir (like Muddy), the maximum release is interpolated from the Max Release table based on its elevation.

☞ Close the **Function Editor** window and close the **Bartlett Flood Control** rule

So now that we've looked at the details, let's go back to the overall rule. Basically, it assigns the Barlett Reservoir Outflow slot to the value that is computed on the right side of the assignment (i.e. the equal sign). During flood season and when the reservoir is high, this rule computes and sets the Outflow to ensure that the level of water in the reservoir is reduced to be able to accommodate possible flooding while not exceeding the flow that the turbines and spillways can handle.

Now we will see how the rule affects reservoir operations:

☞ Turn on the **Bartlett Flood Control** rule by clicking the red **X** in the **Ruleset Editor.**

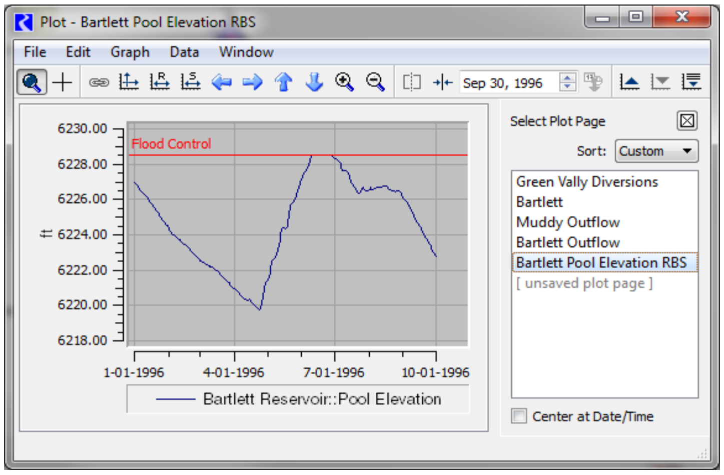
The X turns into a green check mark and the orange check mark next to Bartlett Rules turns green.

☞ Run the simulation again (click and then Start). Close the dialogs after the run finishes.

☞ Open the Model Run Analysis (click) and scroll to the June 10th entry for Bartlett Reservoir.

↑8 ↓1R *As you can see, the outflow is now set by rule 1 (1R) where before it was set by rule 2. In fact, all of the Bartlett Reservoir outflows through June 27th are set by rule 1.*

☞ Now open the Pool Elevation plot you looked at before by clicking P and choosing **Bartlett Pool Elevation RBS** from the list on the right.



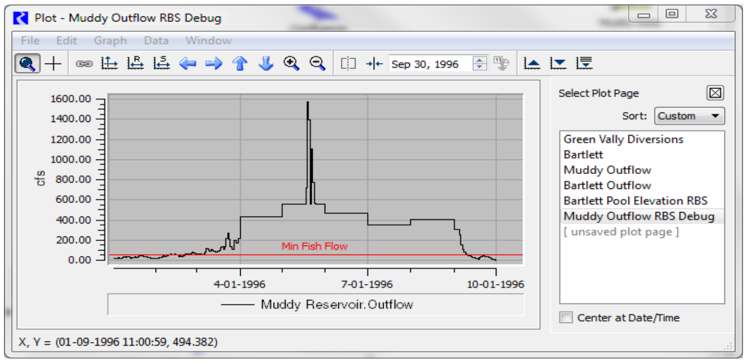
Now that the **BartlettFloodControl** rule is activated, Pool Elevation does not exceed the Flood Control height at any time. Because the rule has the highest priority, no other operation request (i.e. rule) can override it.

2.3 Building a rule

The current ruleset is overlooking a basin requirement to keep the downstream fish population healthy. This operation is especially important during the winter months when natural streamflow is generally low, as you can see in the following plot:

☞ Open the plot of Muddy Reservoir.Outflow by clicking on the plot icon P and clicking on the **Muddy Outflow RBS Debug** plot.

The red line shows the minimum flow required for downstream fish health (55 cfs).



☞ Zoom in on the beginning of the simulation (January to April) to see clearly that Outflow is below the minimum by clicking and dragging a rectangle over the far left portion of the graph (resize the window if necessary).

☞ Close the plot.

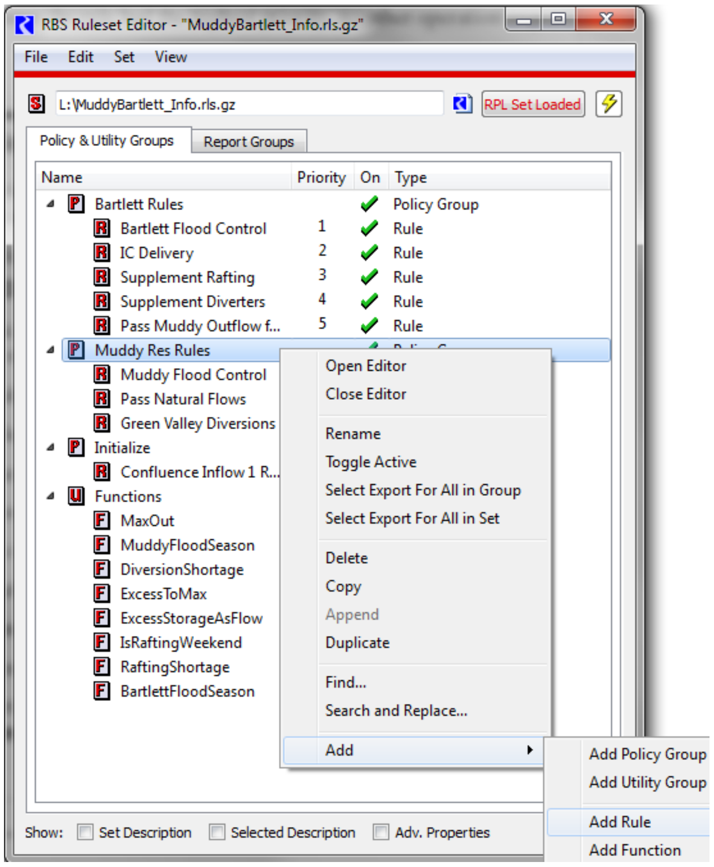
You will now build a simple rule which sets Muddy Reservoir's Outflow to meet this downstream environmental flow requirement of 55cfs.

☞ Add a new rule to the Muddy Reservoir Policy Group by highlighting **Muddy Res Rules** and right clicking. In the ensuing popup menu, choose **Add ➥ Add Rule**.

A new entry called "Rule" appears in the Policy Group as priority 9.

☞ Open **Rule** by double clicking on it and change the name from "`Rule`" to "`Min Fish Flow`" by entering it into the field at the top of the **Rule Editor** and pressing Enter.

Rules are constructed by adding statements at the top level and then filling in the expressions in the statements. Statements can be added from the **Rule** menu. The most common statement is the **Assignment** which is used to set a slot value.

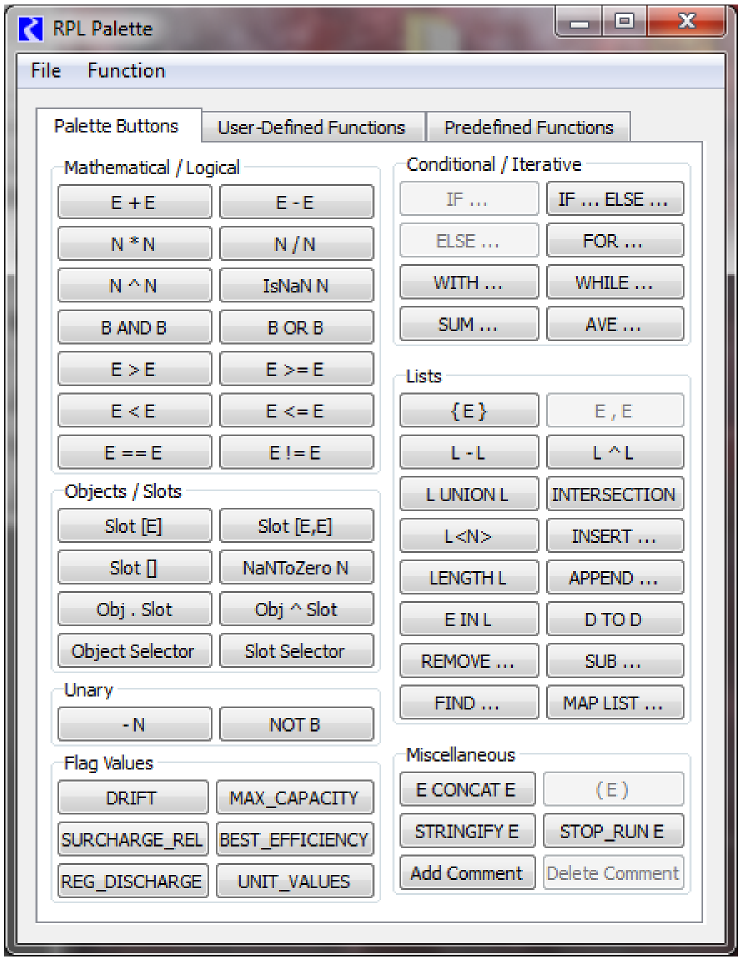


Name	Priority	On	Type
Bartlett Rules		✓	Policy Group
Bartlett Flood Control	1	✓	Rule
IC Delivery	2	✓	Rule
Supplement Rafting	3	✓	Rule
Supplement Diverters	4	✓	Rule
Pass Muddy Outflow f...	5	✓	Rule
Muddy Res Rules			
Muddy Flood Control			
Pass Natural Flows			
Green Valley Diversions			
Initialize			
Confluence Inflow 1 R...			
Functions			
MaxOut			
MuddyFloodSeason			
DiversionShortage			
ExcessToMax			
ExcessStorageAsFlow			
IsRaftingWeekend			
RaftingShortage			
BartlettFloodSeason			

☞ Add an assignment statement to the empty rule by selecting **Rule ➥ Add Assignment**.

Now we must now specify which slot to set (the left side) and logic to determine the value to set on the right side. We will fill in both of the unspecified expressions. Unspecified expressions are portions of the rule which have not yet been defined, for example: **<numeric expr>**. The two **<numeric expr>** expressions which appear in your rule indicate that both sides must eventually evaluate to a numeric value. Building rules involves coding the unspecified expressions one by one into functional expressions.

You specify the expressions using the **Palette.** Depending on the type of the expression highlighted, the Palette only enables the buttons for expressions of that type. In this way, the Palette provides syntax support and helps guide you in building rules.



☞ Open the **Palette** by selecting **Rule ➥ Palette....** Or right click on the expression and select **Palette**.

When the Palette window is active, you can hover over any button on the Palette to get a "tool tip" describing that button.

☞ 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 **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. The three possibilities are:

1. **Slot [E]** A series slot at a given row (often specified by a date).
2. **Slot [ ]** A series slot at the current controller timestep or a scalar slot.

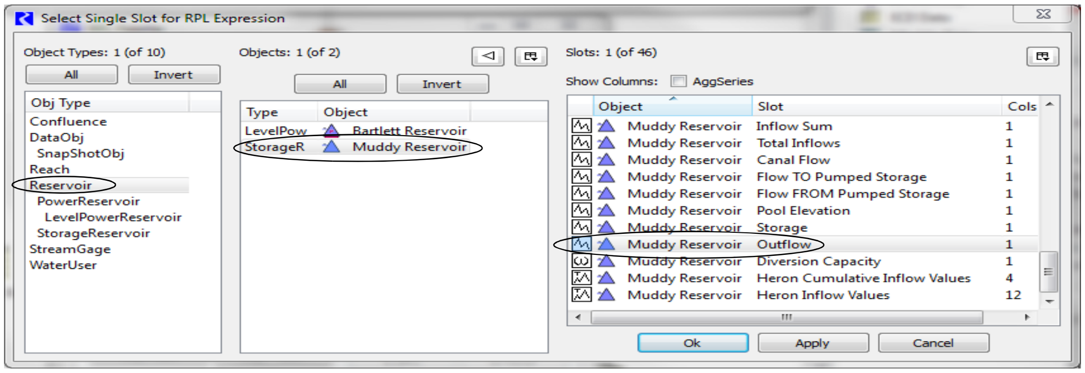
3. **Slot [E, E]** A table slot at a specified row and column.

☞ Mouse over each of these three buttons to see their description

☞ Now highlight just the unspecified portion of the series slot expression, **<expr>**, and click on the **Slot Selector** button in the **Objects/Slots** section of the Palette.

☞ In the slot selector, select the **Object Type**: **Reservoir**, **Object**: **Muddy Reservoir**, **Slot**: **Outflow**, then click **OK**.

You will need to scroll down to locate Outflow.



Select Single Slot for RPL Expression

Object Types: 1 (of 10)

Obj Type
Confluence
DataObj
SnapShotObj
Reach
Reservoir
PowerReservoir
LevelPowerReservoir
StorageReservoir
StreamGage
WaterUser

Objects: 1 (of 2)

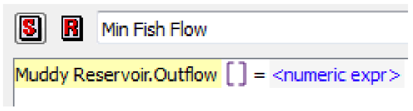
Type	Object
LevelPow	Bartlett Reservoir
StorageR	Muddy Reservoir

Slots: 1 (of 46)

Show Columns: ☐ AggSeries

Object	Slot	Cols
Muddy Reservoir	Inflow Sum	1
Muddy Reservoir	Total Inflows	1
Muddy Reservoir	Canal Flow	1
Muddy Reservoir	Flow TO Pumped Storage	1
Muddy Reservoir	Flow FROM Pumped Storage	1
Muddy Reservoir	Pool Elevation	1
Muddy Reservoir	Storage	1
Muddy Reservoir	Outflow	1
Muddy Reservoir	Diversion Capacity	1
Muddy Reservoir	Heron Cumulative Inflow Values	4
Muddy Reservoir	Heron Inflow Values	12

You have finished the left-hand side (LHS) of the rule. When the rule is executed, the rule will attempt to set a value on the Muddy Reservoir Outflow slot to the value returned from the right-hand side (RHS) of the assignment.



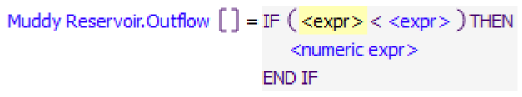
```
Min Fish Flow
Muddy Reservoir.Outflow [] = <numeric expr>
```

The right-hand side of the rule should check the existing flow and increase it if necessary:

☞ Highlight the RHS **<numeric expr>**.

☞ From the Palette choose the **IF ...** option under the Conditional/Iterative section.

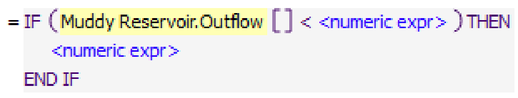
☞ Highlight **<boolean expr>** to specify the type of condition, then select **E < E** to signify that you want to determine if the value of a slot is less than the value of another.



```
Muddy Reservoir.Outflow [] = IF ( <expr> < <expr> ) THEN
                                  <numeric expr>
                             END IF
```

☞ Highlight the left **<expr>** and designate it as a slot at the current timestep, **Slot [ ]**, then highlight just **<expr>** again and click on the **Slot Selector** button in the **Objects/Slots** section of the Palette.

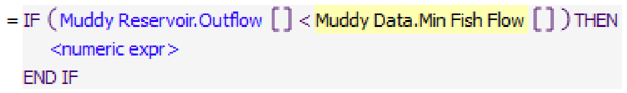
☞ In the slot selector, select the **Object Type**: **Reservoir**, **Object**: **Muddy Reservoir**, **Slot**: **Outflow**, then click **OK**.



```
= IF ( Muddy Reservoir.Outflow [] < <numeric expr> ) THEN
    <numeric expr>
  END IF
```

☞ Highlight **<numeric expr>** and designate it as a slot at the current timestep, **Slot [ ]**, then highlight just **<expr>** again and click on the **Slot Selector** button in the **Objects/Slots** section of the Palette.

☞ In the Slot Selector, select the **Object Type**: **DataObj**, **Object**: **Muddy Data**, **Slot**: **Min Fish Flow**, then click **OK**.



```
= IF ( Muddy Reservoir.Outflow [ ] < Muddy Data.Min Fish Flow [ ] ) THEN
    <numeric expr>
  END IF
```

The conditional statement is complete. Now we will specify the value of the Outflow slot using a predefined RiverWare function.

☞ Highlight **<numeric expr>** and navigate to the **Predefined Functions** tab of the Palette.

☞ At the bottom of the Palette window, check the box next to "**Show only functions with a return type matching the selected expression**"

☞ Scroll through the alphabetical list to the **Min** function. Double click on it to insert it into the rule.

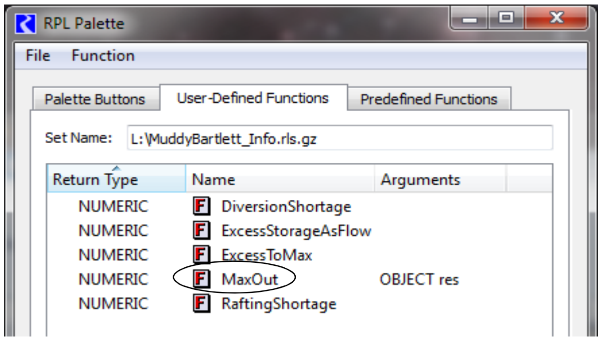
The rule now reflects that we want the rule to compare two values and choose the smallest.

The purpose of the rule is to set outflow to a level that protects the fish downstream, but that should not conflict with the maximum outflow possible for the reservoir, so the smaller of the two values will be used.

☞ Highlight the first **<numeric expr>** and make it a slot at the current timestep: **Slot [ ]**, then highlight **<expr>** and in the Slot Selector specify **Object Type**: **DataObj**, **Object**: **Muddy Data**, **Slot**: **Min Fish Flow**, then click **OK.**

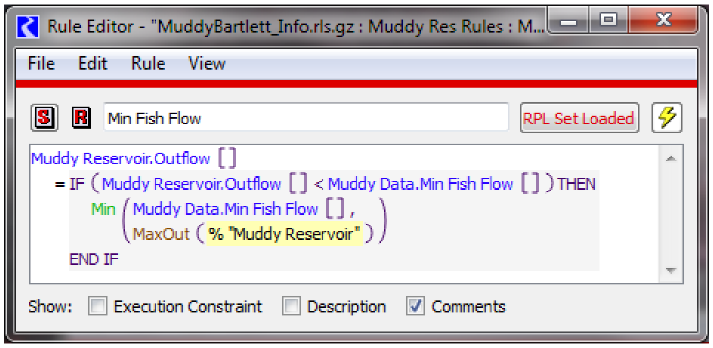
☞ Highlight the remaining **<numeric expr>** and navigate to the **User-Defined Functions** tab of the Palette. Double click on **MaxOut**.

☞ Highlight the <object expr> argument to specify which reservoir you want to look at and choose Object Selector (from the Palette Buttons tab). In the selector dialog choose **Object Type**: **Reservoir**, **Object**: **Muddy Reservoir**, then click **OK.**



Set Name: L:\MuddyBartlett_Info.rls.gz

Return Type	Name	Arguments
NUMERIC	DiversionShortage	
NUMERIC	ExcessStorageAsFlow	
NUMERIC	ExcessToMax	
NUMERIC	MaxOut	OBJECT res
NUMERIC	RaftingShortage	


Rule Editor - "MuddyBartlett_Info.rls.gz : Muddy Res Rules : M...
File
Edit
Rule
View
RPL Set Loaded

Min Fish Flow

```
Muddy Reservoir.Outflow [ ]
  = IF ( Muddy Reservoir.Outflow [ ] < Muddy Data.Min Fish Flow [ ] ) THEN
      Min ( Muddy Data.Min Fish Flow [ ] ,
            MaxOut ( % "Muddy Reservoir" ) )
    END IF
```

Show: ☐ Execution Constraint ☐ Description ☑ Comments

Your finished rule should look like the above figure.

☞ Close the Rule Editor window and return to the Ruleset Editor window.

The Min Fish Flow rule you just wrote is currently at priority 9, but it needs to be at priority 7.

☞ Click and drag the rule to just below the **Muddy Flood Control** rule and release the left mouse button. A confirmation window will appear asking if you want to finalize the move from priority 9 to priority 7. Click **OK**.

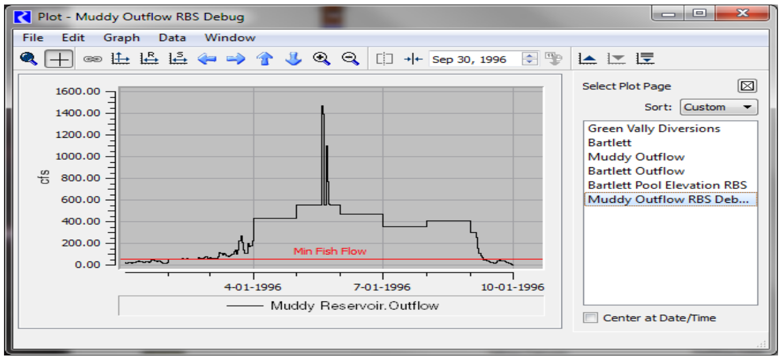
Now Min Fish Flow is priority 7 and the remaining rules have shifted down one priority.

☞ Save the new Ruleset via **File ➥ Save RBS Ruleset**

☞ Run the model.

☞ Open the same plot of Muddy Reservoir.Outflow you looked at in section Section 2.3, “Building a rule” by clicking on the plot icon and clicking on the **Muddy Outflow RBS Debug** plot.

The red line shows the minimum flow required for downstream fish health (55 cfs).

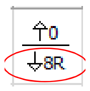


☞ Zoom in on January and February 1996 (resize the window if necessary).

After zooming in we can see that the violations were fixed starting in February, but not in January. Let’s use the Model Run Analysis to see which rules were in effect for this month.

☞ Open the **Model Run Analysis** dialog by clicking on the icon, and scroll left to the **January 1, 1996** timestep.

☞ Click on the cell for Muddy on January 1.



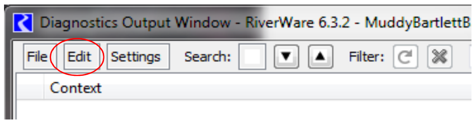
Muddy Reservoir’s Outflow was set by rule 8 for the whole month of January but the **Min Fish Flow** rule that you wrote is at priority 7, a higher priority that should have overwritten rule 8. Obviously there is a breakdown somewhere. In the next section we debug this rule to find the problem.

2.4 Debugging

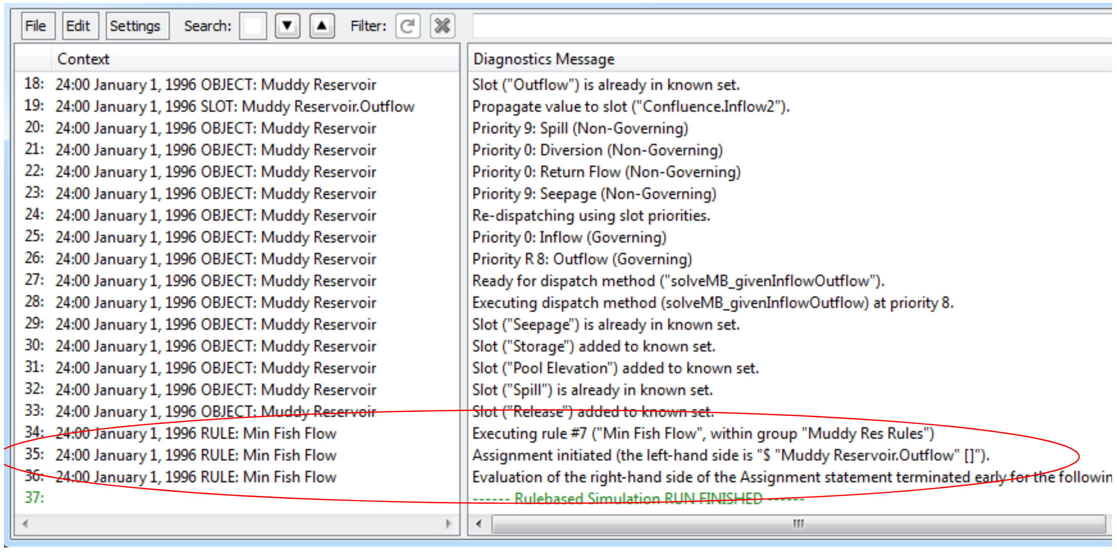
After setting up a rulebased simulation model, it is important to verify the results and debug any issues. Some problems within a model or ruleset will cause the run to abort generating an error message that can direct you to the problem. Other times a model may run to completion, but it may not produce the correct results. These problems can be more difficult to debug. This section will present two tools: Diagnostics and the RPL Debugger, which can assist in this debugging process.

☞ Open the **Diagnostics Output Window** by selecting **Utilities ➥ Diagnostics Output...** from the menu bar in the RiverWare workspace.

☞ Select click **Edit ➥ Clear Messages** from the menu, to clear the messages from all the previous runs.


Diagnostics Output Window - RiverWare 6.3.2 - MuddyBartlettB
File
Edit
Settings
Search:
Filter:
Context

Since we only want to look at diagnostics for how rules set the Muddy Reservoir Outflow in January, 1996, the diagnostics have been pre-configured using context filters.



	Context	Diagnostics Message
18:	24:00 January 1, 1996 OBJECT: Muddy Reservoir	Slot ("Outflow") is already in known set.
19:	24:00 January 1, 1996 SLOT: Muddy Reservoir.Outflow	Propagate value to slot ("Confluence.Inflow2").
20:	24:00 January 1, 1996 OBJECT: Muddy Reservoir	Priority 9: Spill (Non-Governing)
21:	24:00 January 1, 1996 OBJECT: Muddy Reservoir	Priority 0: Diversion (Non-Governing)
22:	24:00 January 1, 1996 OBJECT: Muddy Reservoir	Priority 0: Return Flow (Non-Governing)
23:	24:00 January 1, 1996 OBJECT: Muddy Reservoir	Priority 9: Seepage (Non-Governing)
24:	24:00 January 1, 1996 OBJECT: Muddy Reservoir	Re-dispatching using slot priorities.
25:	24:00 January 1, 1996 OBJECT: Muddy Reservoir	Priority 0: Inflow (Governing)
26:	24:00 January 1, 1996 OBJECT: Muddy Reservoir	Priority R 8: Outflow (Governing)
27:	24:00 January 1, 1996 OBJECT: Muddy Reservoir	Ready for dispatch method ("solveMB_givenInflowOutflow").
28:	24:00 January 1, 1996 OBJECT: Muddy Reservoir	Executing dispatch method (solveMB_givenInflowOutflow) at priority 8.
29:	24:00 January 1, 1996 OBJECT: Muddy Reservoir	Slot ("Seepage") is already in known set.
30:	24:00 January 1, 1996 OBJECT: Muddy Reservoir	Slot ("Storage") added to known set.
31:	24:00 January 1, 1996 OBJECT: Muddy Reservoir	Slot ("Pool Elevation") added to known set.
32:	24:00 January 1, 1996 OBJECT: Muddy Reservoir	Slot ("Spill") is already in known set.
33:	24:00 January 1, 1996 OBJECT: Muddy Reservoir	Slot ("Release") added to known set.
34:	24:00 January 1, 1996 RULE: Min Fish Flow	Executing rule #7 ("Min Fish Flow", within group "Muddy Res Rules")
35:	24:00 January 1, 1996 RULE: Min Fish Flow	Assignment initiated (the left-hand side is "$ "Muddy Reservoir.Outflow" []").
36:	24:00 January 1, 1996 RULE: Min Fish Flow	Evaluation of the right-hand side of the Assignment statement terminated early for the followin
37:		------ Rulebased Simulation RUN FINISHED ------

☞ Scroll to lines which show "**24:00 January1, 1996 RULE: Min Fish Flow**" in the context panel.

These entries show the process RiverWare went through to implement the Min Fish Flow rule. The third diagnostic message for the Min Fish Flow rule specifically illuminates the problem:

36: 24:00 January 1, 1996 RULE: Min Fish Flow: Evaluation of the right-hand side of the Assignment statement terminated early for the following reason: Encountered invalid value in the series slot "Muddy Data.Min Fish Flow" at the date 24:00 January 1, 1996. This occurred at the following location within the expression: $ "Muddy Data.Min Fish Flow" [].

The first piece of important information that we get from this message is that the rule "terminated early." A rule terminates early when it does not have enough information to evaluate the rule, i.e. it finds a NaN ("Not a Number") in a referenced slot. When it terminates early, it does not set any values. This explains why rule 7 did not set the **Muddy Reservoir Outflow** for January, 1996. The message then tells us why the rule terminated early: **Encountered invalid value in the series slot "Muddy Data.Min Fish Flow" at the date 24:00 January 1, 1996.**

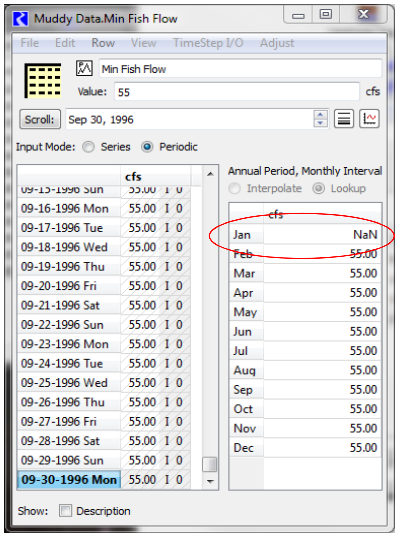
The diagnostic message suggests that we need to look at **Muddy Data.Min Fish Flow.**

☞ Open the **Muddy Data** object, and open the **Min Fish Flow** slot.

When the model was set up the January cell was left without a value, so for the whole month there was no minimum flow data for RiverWare to meet.

☞ Click in the cell, type "`55`" and press Enter.

☞ Rerun the model and see that the rule was successfully implemented.

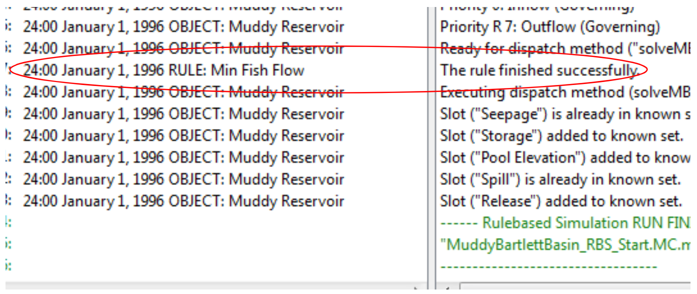


Min Fish Flow
Value: 55 cfs
Scroll: Sep 30, 1996
Input Mode: Series [✓ Periodic]

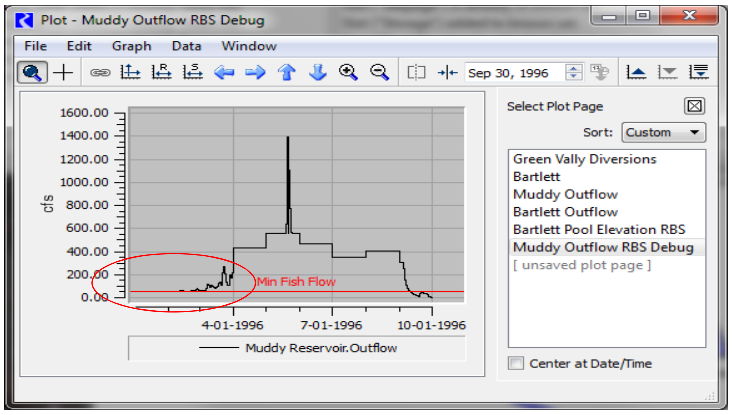
	cfs		
09-15-1996 Sun	55.00	I	0
09-16-1996 Mon	55.00	I	0
09-17-1996 Tue	55.00	I	0
09-18-1996 Wed	55.00	I	0
09-19-1996 Thu	55.00	I	0
09-20-1996 Fri	55.00	I	0
09-21-1996 Sat	55.00	I	0
09-22-1996 Sun	55.00	I	0
09-23-1996 Mon	55.00	I	0
09-24-1996 Tue	55.00	I	0
09-25-1996 Wed	55.00	I	0
09-26-1996 Thu	55.00	I	0
09-27-1996 Fri	55.00	I	0
09-28-1996 Sat	55.00	I	0
09-29-1996 Sun	55.00	I	0
09-30-1996 Mon	55.00	I	0

Annual Period, Monthly Interval
Interpolate [✓ Lookup]

	cfs
Jan	NaN
Feb	55.00
Mar	55.00
Apr	55.00
May	55.00
Jun	55.00
Jul	55.00
Aug	55.00
Sep	55.00
Oct	55.00
Nov	55.00
Dec	55.00



```
24:00 January 1, 1996 OBJECT: Muddy Reservoir    Priority R 7: Outflow (Governing)
24:00 January 1, 1996 OBJECT: Muddy Reservoir    Ready for dispatch method ("solveMI
24:00 January 1, 1996 RULE: Min Fish Flow        The rule finished successfully.
24:00 January 1, 1996 OBJECT: Muddy Reservoir    Executing dispatch method (solveMB
24:00 January 1, 1996 OBJECT: Muddy Reservoir    Slot ("Seepage") is already in known s
24:00 January 1, 1996 OBJECT: Muddy Reservoir    Slot ("Storage") added to known set.
24:00 January 1, 1996 OBJECT: Muddy Reservoir    Slot ("Pool Elevation") added to know
24:00 January 1, 1996 OBJECT: Muddy Reservoir    Slot ("Spill") is already in known set.
24:00 January 1, 1996 OBJECT: Muddy Reservoir    Slot ("Release") added to known set.
                                                 ------ Rulebased Simulation RUN FIN
                                                 "MuddyBartlettBasin_RBS_Start.MC.m
                                                 --------------------------------
```



Looking at the right side of the Muddy Outflows plot, we see that the Min Fish Flow is not being met at the end of the simulation period either. It appears that there could be another problem with either the policy or the date. We can use the RPL Debugger to find out why the minimum flow is not being met.

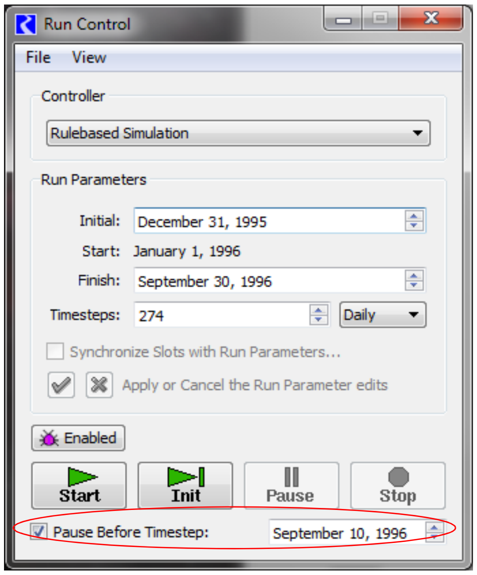
☞ In the RiverWare workspace menu bar, select **Policy ➥ RPL Debugger...**

This opens the RPL Debugger dialog. At this point the debugger is not yet enabled.

Rulesets can be very large and complicated, and consequently it can be difficult to determine why a set is doing what it is doing. The RPL debugger is designed to help understand the behavior of rulesets by pausing execution, looking at the values of RPL expressions as they are evaluated, and stepping through RPL execution.

☞ In the RPL Debugger dialog, select **Debug ➥ Enable RPL Debugging.**

We will set a breakpoint in the Min Fish Flow rule, but we only want to debug on a single timestep. If we set the breakpoint now, the debugger will stop the run once every timestep when it gets to the rule. Instead we will set the **Run Controller** to pause the run before the timestep we want. Then we can set our breakpoint.


Run Control

Controller: Rulebased Simulation
Run Parameters
Initial: December 31, 1995
Start: January 1, 1996
Finish: September 30, 1996
Timesteps: 274 Daily
Synchronize Slots with Run Parameters...
Apply or Cancel the Run Parameter edits

Pause Before Timestep: September 10, 1996

☞ In **Run Control** dialog (click to open), check the box for **Pause Before Timestep** at the bottom of the window.

☞ In the timestep field type "`Sept 10, 1996`" and press Enter.

☞ Click Start to run the model

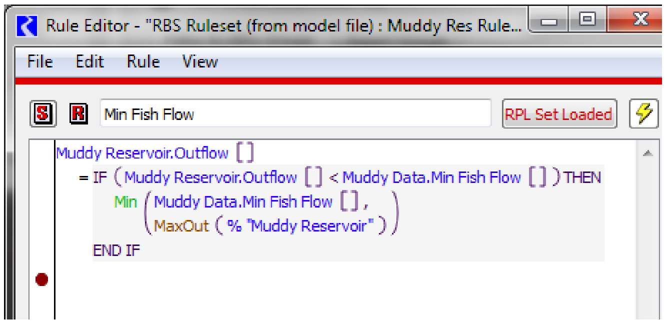
The run pauses at 93% completion, before September 10th.

☞ Open the **Min Fish Flow** rule (in the main workspace, select **Policy ➥ MuddyBartlett_Info_RBSstart** if the ruleset windows have been closed).

Note that there is now an added margin on the left of the **Rule Editor***.*

☞ Click at the bottom of the column on the left of the **Rule Editor** (below all of the expressions in the rule) to set a breakpoint at the end of the rule.

A red stop sign (octagon) appears indicating that a breakpoint has been set. If the octagon appears at the top of the rule accidentally, click on it twice to remove it and try again.


Rule Editor - "RBS Ruleset (from model file) : Muddy Res Rule...

```
Muddy Reservoir.Outflow [ ]
  = IF ( Muddy Reservoir.Outflow [ ] < Muddy Data.Min Fish Flow [ ] ) THEN
      Min ( Muddy Data.Min Fish Flow [ ] ,
            MaxOut ( % "Muddy Reservoir" ) )
    END IF
```

☞ In the **Run Control** dialog, click **Step**.

The run will pause at the breakpoint, and the **RPL Debugger** *dialog will come to the front. Note that the middle panel of the debugger lists the single breakpoint. The top panel shows the current location in the evaulation of rules and functions, called the call stack.*

Now we can see what values the different parts of the rule produce:

☞ First, in the **Min Fish Flow** rule, click on the brackets **[ ]** after **Muddy Data.Min Fish Flow**

The bottom panel in the Debugger reads "55 cfs".

☞ Next, click on the **MaxOut** function.

The bottom panel in the Debugger reads "49.97 cfs".

☞ Finally, click on the brackets **[ ]** after **Muddy Reservoir.Outflow**

The bottom panel in the Debugger reads "49.97 cfs".

As the rule dictates, **Muddy Reservoir.Outflow** is set to the minimum of the two values. Although we wanted to release 55cfs, the **MaxOut** function computed that only 49.97cfs could be released based on the amount of water currently in the reservoir. This is the purpose of including the **MaxOut** function in the rule. It assures that the rule will not try to release more water than is physically possible, so an Outflow below the Min Fish Flow is the correct outcome of the operating rules.

☞ Stop the debugging process by clicking **Stop** in the **RPL Debugger**, then disable the Debugger in the **Run Control** dialog by clicking Enabled.

☞ Un-check the box next to **Pause Before Timestep** and click **Start** to complete a run before ending the RiverWare session.

The run status reaches 100%.

Although debugging is a very advanced RiverWare topic, it is useful for even the novice user to know that it is possible to understand exactly what the software is doing. Debugging is critical for determining how the simulation proceeds. This section presented two tools for accessing the details within a rulebased simulation:

- **Diagnostics** provide a play-by-play of every step in the simulation. Once context filters are applied, you can pinpoint the exact interaction you want to analyze. More information on this topic can be found in the **Diagnostics** section of the help.
- The **RPL debugger** takes you inside a rule to see what values are produced, in order to check that you have the proper logic. More information on this topic can be found in the **RPL Debugging Tools** section of the help.