USBR Truckee/Carson RiverWare Model Report

Functional Requirements

Patrick Lynn, CADSWES

This document presents requirements for a new type of RiverWare output device type whose purpose is to produce model reports for the USBR Truckee/Carson division. Another document, “RiverWare Model Report: Functional Requirements and Prototype Description” provides background as well as more general requirements for the model report output device type. Here we focus on the needs of this particular sponsor and how they will be met by the model report output device type.

1 Needs

In this section we present and discuss the model reporting needs of the sponsor. The following section contains the proposed RiverWare functional requirements to meet these needs.

1.1 Contents of the documentation

The documentation should be organized hierarchically with the following top-level sections:

- Textual description of the model (possibly including links to external documents)
- Section for each process
- A listing of external documents associated with the model

In this context, a process is a user-defined collection of related workspace and policy objects. Inclusion in a process is non-exclusive, that is, a rule can be associated with multiple processes, and rules need not be associated with any process. Each process section should include the following:

- Textual description of the process (possibly including links to external documents)
- Section for each rule associated with the process
- List of workspace objects associated with the process (SimObj names with textual descriptions)

Each Rule section should contain the following:

- Textual description of the rule (possibly including links to external documents)
- Section for each function accessed by the rule
- List of slots set by the rule

Each Function section should contain the following:

- Textual description of the rule (possibly including links to external documents)
- List of slots accessed by the function
- The body of the rule (optional)

The following figure presents a view of the documentation contents which is organized by the level of detail. Note that in this figure, function sections have been moved two the same level as that of rules. This organization is more consistent with the existing RPL structure and helps avoid duplicated content in the model report.

Figure 1. Organization of report contents by level of detail.

Level 1 (top-level)	Level 2	Level 3	Level 4
Description			
Processes	Description		
	Rules	Description	Statements
		Functions (called)	
		Slots (set)	
	Functions	Description	Body
		Slots (accessed)	
		Functions (called)	
	SimObjs	Description	
List of external documents			

1.2 Interactive viewer

It should be possible to view the documentation online with interactive control of the level of detail. Ideally the user would be able to dynamically hide/show the contents of any section, at any level, as well as view all content to a given level of detail. For example, a particular user might want to view the contents to the second level of detail in Figure 1, then interactively choose to view more detail for certain rules in certain process sections.

It should be possible to generate a static version of the documentation with a similar degree of flexibility with respect to the visible level of detail. One way to achieve this functionality would be for the interactive viewer to support output that respects the currently visibility selections. The format of the static report should be PDF or some other common electronic document format.

From the interactive viewer it should be possible to view external documentation either in-line, at the point in the report where the document is referenced, or via an active hypertext link. Ideally, the interactive viewer would support display of external documents in a variety of formats, e.g., plain text, HTML, PDF, JPG, and PNG.

In a printed version of the report, references to external documents should be displayed as citations that refer to entries in the document bibliography, that is, in the complete list of external documents which is the last top level section in the document.

1.3 Management of external documents

External documents referenced from within the model report might be specified by absolute or relative file paths, URIs, or possibly references to another output device (e.g., a RiverWare plot page).

The following bibliographic database functionality should be available to help manage references to external documents:

- View and edit entries in a reference database (add, delete, edit entry).
- Import entries to (and export entries from) the reference database in common formats.
- Use the reference database to generate the bibliography section of the model report. The bibliography might include all entries in the database or only those cited in the report.
- Facilitate the adding of an external document reference to the model report by allowing selection from the reference database or lookup by tag.

• Flexible formatting of links to external documentations as citations in the static model report.

1.4 Discussion

1.4.1 Additional content

Several types of content not requested by the sponsor would likely be useful to some users and its omission would be inconsistent with other aspects of the RiverWare conceptual framework and user interface:

• Rule statements and execution constraint. This is RPL code at a level of detail analogous to function bodies, and can include logic as critical as that in functions.

• Rule priority. This is an important property that does not take much space to display.

• Function parameter names, types, and descriptions, and return type.

• SimObj type, slot type.

We should consider supporting the inclusion of this information in the report.

1.4.2 The process concept

Within RiverWare, where should a user create, view and edit processes? Processes contain references to objects on the workspace as well as to policy (RPL) objects. Since multiple policies can be used with the same model, but not vice versa, it seems most natural to associate processes with policies. The raison d'etre of RPL sets is accessing and setting values on workspace objects, and so RPL sets necessarily contain references to workspace objects, and adding additional references that serve to document the policy is consistent with this architecture.

By allowing a view of the policy which is organized according to concerns other than priority, incorporating the process concept into RPL sets could provide utility to users whether or not they are generating external documentation.

1.4.3 Report configuration

The needs involve two points at which report content is specified:

1. Selection of content to be displayed in the interactive viewer.
2. Selection of content to be included in the static report.

Since the interactive viewer itself provides flexible control over which aspects of the content are visible, specification of the full contents for the viewer (1) does not need to provide much in the way of filters. For example, the user does not need to be able to specify which processes should be included or which rules are included for specific processes, the viewer can display all processes and, for each process, all rules.

On the other hand, the selection of static report contents needs to be highly flexible with respect to its contents. If this process is time-consuming or tedious, and if a given selection might be used to generate a report more than once, then it would be useful if support was provided for persistence of these selections.

1.4.4 The interactive viewer

An interactive document viewer could be incorporated into RiverWare, but this might involve a large software effort not central to the primary purpose of RiverWare, and furthermore this viewer would necessarily be limited in its support for various formats. Therefore, if one or more easily available third party application can meet the interactive viewing needs, this would be a preferable approach. We require a document display application that sup-

ports: 1) hypertext links to documents in various formats, 2) interactive hiding/showing of content, 3) output to a common electronic format which respects the current visibility settings.

Modern web browsers in combination with dynamic HTML are capable of meeting these needs. The following outlines how these technologies might be used in the context of RiverWare model reports:

- Within RiverWare, the user would configure an output device, specifying the report content, organization, and formatting.
- Generating output from the device configuration would create an HTML file containing the full report (i.e., all the details described in Section 1.1).
- The document could be viewed using any application capable of displaying HTML, including web browsers (e.g., Firefox, Internet Explorer) as well as word processing applications such as Microsoft Word. Many of these applications support the display of other formats, thus links within the model document could be used to dynamically access related documents in a variety of formats.
- The HTML file would use JavaScript and the document object model (DOM) to provide interactive control of the content behavior. Modern web browsers support this technology, but it might be less common for applications with a different focus.
- If a user has Adobe Acrobat Pro X or the equivalent installed on their machine, then it is possible from a web browser to either print to a file using the PDF format or to generate a set of linked PDF documents that mirrors the linking structure of a set of web pages.

Some decisions would need to be made when RiverWare generates the HTML document:

- When the model document is opened in an application, which level of detail should initially be visible.
- For each external file, should it be included in-line or as a link.

Challenges for the dynamic HTML approach:

- Displaying links in the static document differently from their display in the HTML viewer application, though dynamic HTML techniques might provide a way to accomplish this.
- Global operations from within the HTML viewer, such as: close or show all sections at a given level.
- A mechanism for persistence of the shown/hidden sections in a static report.
- Since a report is contained within a single file, displaying a report in a web browser could pose some performance problems, especially if it contains a large amount of binary data (images).

Note that all of the information in the proposed model report would also be contained within the RiverWare model file, and the RiverWare workspace and RPL editors themselves provide much of the desired interactive viewing functionality. That is, RiverWare (or the RiverWare Viewer) can currently be used to view all of the report content at a flexible level of detail. Of course this approach requires that the person viewing the model have access to RiverWare (or the RiverWare Viewer) as well as have had some RiverWare training. Furthermore, this approach does not provide a way to limit the amount of information available to the viewer. Thus while this approach can certainly supplement the model report for some users, it is not appropriate for all users.

1.4.5 Management of external documents

How should RiverWare provide the requested support for references to external documents within descriptions?

Note that currently a description can include HTML text describing a link, e.g.,

```
<a href="http://www.water.org/report.html">(Obama, et al., 2011)</a>
```

If a description containing this text were included in an HTML report document, then the text "(Obama, et al., 2011)" would be presented as an active hypertext link.

Typing in this text requires both knowledge of the relevant HTML syntax as well as error-prone typing. How could RiverWare help with the task of including such a link? One option would be to provide an “insert link” operation accessible from within description editors. This operation would present a dialog which would allow users to enter the link text and the URI separately. When edits are complete, the appropriate HTML would be generated and inserted into the description.

Ideally, links within descriptions would be functional hyperlinks from within RiverWare dialogs that display those descriptions; that is, link texts would be highlighted and clicking on a link would open an appropriate viewer for the referenced document.

For generating the bibliography section, RiverWare could scan the descriptions, identifying the hypertext links and create a list of the citations. The full text description of the cited documents could be gleaned from the “title” attribute of the <a> HTML elements or from a bibliographic database. Further research is required to explore how RiverWare might directly integrate with existing bibliographic database applications to facilitate insertion of citations into descriptions or the creation of the bibliography.

2 Requirements

2.1 New type of RPL group (Report Group)

Add support for a new type of RPL group, tentatively called a Report Group, which provides and organizational grouping of related RPL and workspace objects. Membership in a Report Group is determined by the user and does not affect behavior during a run. A Report Group contains a set of members which can be one of the following types:

- RPL rule (or method, goal, or initialization rule, as appropriate for the RPL application)
- RPL function
- SimObj

Report Groups can contain only rules and functions that are in the same RPL set. A rule can be contained in zero, one, or more Report Groups. Report Groups automatically update if a members name is changed or a member is deleted.

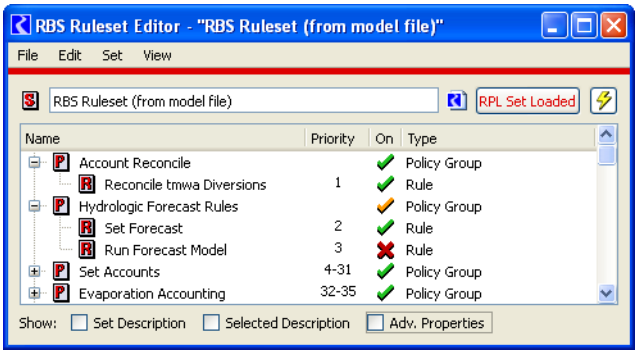
Like other types of RPL groups, Report Groups have a name which must be unique within the set and a description.

The RPL set editor displays a Report Groups as a tree view with the tree’s children being the group’s rules, functions, and SimObjs. New editor operations include adding, removing, and reordering members. Removing a rule or function from a Report Group does not remove that object from its parent policy or utility group, and similarly, removing a SimObj does not remove that object from the workspace. Double-clicking on a Report Group member opens the associated editor.

For reference, Figure 2 shows the current RPL set editor with two expanded policy groups in view. If these were instead Report Groups, the primary differences would be:

- Rule priorities would not necessarily be sequential.
- SimObj’s could be included as well.
- The Edit menu would include the ability to add SimObj’s when a Report Group is selected.

Figure 2. The current RPL set editor



If the need arises in the future, we could extend Report Group membership to additional types of workspace objects, such as accounts, supplies, and exchanges, and slots.

Because Report Groups are very different from policy and utility groups, we should consider showing them on a separate tab within the RPL set editor, one that is perhaps optionally visible.

2.2 New output device type (Model Report)

As part of work for another sponsor, requirements have been created for a new model report output device type, and a prototype was implemented. Those requirements will be revised based on experience with the prototype, and the requirements will also be adjusted to support the requirements discussed here. In particular, this device will need to support:

- The RPL set output item type.
  Output of a RPL set will generate a listing of the set's groups, organized hierarchically as illustrated in Figure 3. Formatting options will control which classes of groups are included in the report (e.g., only Report Groups). This approach is simpler than specifying each Report Group separately in the configuration, though in the future we might want to support that as well. Groups will be output in the order in which they appear in the set (this ordering can be changed in the set editor).
- For RPL set output items, RPL code (e.g., rule statements, function bodies) will be generated as images. A formatting option will control whether these images are included in-line or as links.
- A report level HTML formatting option will control whether or not sections can be interactively hidden/shown. When this option is selected, each section in the output HTML will be accompanied by a show/hide button with an appropriate icon. When the button is clicked, the contents of that section will be shown/hidden. This will be achieved through application of dynamic HTML technology (DHTML, CSS style sheets, JavaScript, DOM). The following HTML document illustrates this approach:
  http://cadswes2.colorado.edu/~lynn/Doc/2011/ShowHideDemo.3.htm.

The primary HTML file will be written to the model report device's output file, but additional files might also be created, including: image of RiverWare icons, images of RPL code, and cascading style sheet files. The names of these files will be based on the primary output file name.

2.2.1 Report Group report content

The following figure illustrates the Report Group content available for inclusion in a model report.

Figure 3. Content of Model Report Report Group output item type.

Description		
Rules (icon + name)	Description	Statements (image)
	Functions called (static analysis)	Execution constraint (image)
	Slots set (during last run)	
Functions (icon + name)	Description	Body (image)
	Return type	
	Argument types	
	Argument names	
	Functions called (static analysis)	
	Slots accessed (during last run)	
SimObjs (icon + name)	Description	

Report Group output item formatting options can be added as necessary to allow further control over the content. For example, users could indicate whether or not the function sections of a report should include the list of argument names, or the rule sections should list the slots set by that rule in the last run.

Note that the dynamic information above (slots set by a rule, slots accessed by a function) is not currently being collected. Collecting it will incur a computational overhead and so should be optional. If it is not collected, then of course it should not be included in the report. At some point, we should make this information directly available from within RiverWare, e.g., in the RPL set analysis tool.

2.3 Enhanced descriptions

Support for descriptions will be extended to all SimObjs (currently this is supported only for DataObjs).

3 Notes

Instantaneous hide/show of content can make it difficult to understand what is happening. One approach to addressing this problem is to implement a “sliding” effect for hiding/showing content whereby the vertical size of the hidden/shown text is gradually changed, in effect animating the transition. For a demonstration, see:

http://www.dhtmlgoodies.com/scripts/show-hide-content-slide/show-hide-content-slide.html

Additional testing should be conducted to verify that the proposed approach for dynamically hiding/showing HTML content scales to large documents. Probably this would only be an issue for documents containing large images.

Collecting dynamic information about RPL evaluation can be tricky. For example, the list of slots set by a rule. Some users might want to know the date of execution, the date at which the value was set, or how many times the value was set by the rule. Similarly, a user might want to collect the dynamic information once, then retain it to avoid the collection overhead in the future (but risk correctness problems). It’s not really possible or desirable to support all of these options.

In general, any of the columns in the RPL set analysis tool are candidates for inclusion in the model report, and could be added in the future via model report RPL set output item formatting options.

Supporting RPL objects as model report item types requires a RPL object selector. We might be able to adapt or generalize an existing dialog for this purpose, such as the RPL set import dialog.

Another application for a RPL object selector would be the “Add rule/function to Report Group” operation. This could be accomplished using drag/drop, but we probably don’t want to rely solely on that mechanism.

The Qt WebKit module (including the QWebView class) provides an up-to-date, powerful HTML rendering widget. Thus it might well be feasible to provide a model report browser within RiverWare. This could allow some additional functionality, but is probably not necessary for the current needs.

Identifying RPL sets by name is not very convenient, but currently there are not any great alternatives. RPL set names are usually just the full path name of the file from which the set was loaded. Configuring a model report by including this full path is inflexible, requiring a different configuration for different policies. We could provide a formatting option that uses the currently loaded RBS/Opt file if there is one. This could be presented as a combo-box listing the open sets:, e.g., “RPL Expression Set” “Loaded RBS Set”, “Loaded Opt Set”, “C:/d1/d2/SetA.rls”, and “C:/d1/d2/SetB.rls”.

4.0 Development Schedule

Days	Task
	New type of RPL group (Report Group)
3	Add new RplObj classes (RplReportGroup, RplReportItem, RplGroup) with skeleton implementations
2	RplReportItem support for various types of references (RplBlock, RplFunction, SimObj, Slot)
2	RPL set and group editors: add display of Report Groups
3	RPL set and group editors: add Report Group editing operations
1	Testing
2	(optional) Support display of Report Groups in a separate set editor tab
2	(optional) Support adding multiple blocks/groups to a Report Group
2	Documentation
	New Model Report output device type (finish implementation of framework)
1	(GUI) Improve geometry management
2	(GUI) Improve editing of contents tree
1	(GUI) Improve display of output item panel
2	(GUI) Improved editing of format settings
2	Design and implement format setting dependency mechanism
2	Add top-level HTML format settings for control over fonts, colors, indentation, etc.
1	Improve error handling
2	Documentation
	Model Report: Support for output of RPL Report Groups
3	Add RPL Group output item type (with format settings to control content and display)
3	Support for RPL block and function code images
2	Add collection and output of dynamic information (slots set or accessed)
	Model Report: Support for dynamic hide/show of HTML content
2	Implement basic machinery (e.g., Javascript header, formatting option)
1	Additional refinements (e.g., develop button images)
	Descriptions
1	Support for all SimObj types
3	Support for editing references to external documents
2	Model Report: Automatic generation of bibliography from descriptions
2	Additional testing and refinement