

UPPER RIO GRANDE WATER OPERATIONS MODEL
RULES DOCUMENTATION
Summary

Table of Contents

DISCLAIMER 1
Introduction 1
Rulebased Simulation 2
 Functions – General Discussion 3
Reservoir Operations 4
 Heron Reservoir..... 4
 El Vado Reservoir..... 4
 Operation for Rio Grande Water 5
 Restrictions of the Rio Grande Compact..... 5
 Water Right Constraints on Operation of Rio Grande Water..... 5
 Operation for San Juan-Chama Water 5
 Abiquiu Reservoir 5
 Operation for Rio Grande Water 5
 Flood Carryover Storage..... 6
 Conservation Storage 6
 Cochiti Reservoir 6
 Operation for Rio Grande Water 6
 Operation for San Juan-Chama Project Water..... 7
 Jemez Canyon Reservoir 7
 Elephant Butte Reservoir..... 7
 Operation for Rio Grande Water: 7
 Operation for San Juan-Chama Project Water:..... 7
 Caballo Reservoir 7
General Rules Description 8
 General..... 8
 Exchanges, credits and debits..... 8
 San Juan Diversion Rules 8
 Heron Reservoir Rules 9
 El Vado Reservoir Rules..... 9
 Abiquiu Reservoir Rules 9
 Cochiti Lake Rules..... 10
 Jemez Canyon Reservoir Rules 10
 Elephant Butte RESERVOIR Rules..... 10
 Caballo Reservoir Rules 10
San Juan-Chama release priorities 10
 Priority Solution Example..... 11
 Assumptions: 11
 Solution 12
APPENDIX A – Simulation and Rulebased Simulation Overview 13
Introduction 13

Simulation 13
Example of Simulation..... 14
Rulebased Simulation 16

Lists of Tables & Figures

Table 1 - Release type assumed priorities for Abiquiu Reservoir 11
Table 2 - Account priorities for Abiquiu Reservoir 12
Table 3 - Steps Of Rulebased Simulation 19

Figure 1 - Example Model For Demonstrating The Simulation Controller In RiverWare 15
Figure 2 - Rulebased Simulation Example Model 18

UPPER RIO GRANDE WATER OPERATIONS MODEL RULES DOCUMENTATION Summary

DISCLAIMER

This (draft) document is a product of Upper Rio Grande Water Operations Model (URGWOM) and is intended to assist technical reviewers understand how the Water Operations Model logic works. The URGWOM Rules reflect the URGWOM Technical Team's interpretation of operational policy, regulations, preferences, and other decision-making logic. URGWOM is not the designer of these policies and regulations, but simply an interpreter and user. For the actual policies, laws, regulations, and implementations, the user is referred to the responsible agencies. This document, and the URGWOM Ruleset, may contain errors in interpretations, statements, and applications of such laws, regulations, and policies. URGWOM and the URGWOM Team members are not responsible for errors that might result from the use of the information provided herein.

The actual URGWOM rule text (code) is in draft form and is continually being updated to reflect changes in the physical model and improvements and upgrades in the rule simulations provided by CADSWES. A description or annotation of the most current version of the URGWOM ruleset may be found in the URGWOM models.

INTRODUCTION

URGWOM has a number of components of substantial development. Two of them are: The physical structure, and the operational logic. This document is an evolving explanation of the operational logic ("Rules"), which are also evolving. The physical structure has undergone substantial development and documentation as presented in the (annual) Technical Reviews the past few years. This is the second official draft of the Rules document published (The first dealt only with the "Rio Chama Test-Case" and came out in September of 1998.) As a "draft document", this document is not complete, but begins to communicate understanding of the "Rules" in terms other than just the code itself. The approach taken is to provide an actual printout of a current Water Operations Model Ruleset, with descriptions of each Rule, and Function alongside each. (Ideally, another method of presenting logic such as flowcharting would be used, but the resources do not allow for that type of effort.) Note that the best way to track Rules is from within RiverWare (an option not available to most reviewers) because you can track through the Rule/Function paths by clicking the next Function name. (Of course, a comprehensive review would have the reviewer track through every possible logic path, which would be a tremendous task!) Since this is a living document describing an evolving Ruleset, it may not include descriptions of every item, and it may be outdated fairly quickly. It will be updated as resources allow.

In basins where multi-objective river and reservoir system modeling is needed, such as the Rio Grande Basin extending slightly beyond New Mexico's northern and southern borders, modeling uses a set of independent policy objectives to drive a solution of releases and/or other control actions that satisfy the objectives as well as possible. The main discretionary control executed by "Rules" on this river/reservoir system is in setting the outflow amounts from the reservoirs. This makes sense, because in the real world of operation, outflow is what we directly control, not reservoir elevation, inflow, evaporation, or precipitation! Typical objectives reflect such policies as meeting downstream demands, preventing flooding, providing flows for aquatic and riparian habitat, providing flows and lake levels for recreation, protecting senior water rights, and other regulation policies. Conflicting objectives compete by a system of prioritization. For example, flood control for public safety is always at the top of priorities, and recreational requirements are often very low. Generally in RiverWare, "Groups" of Rules, as the Rules themselves, are placed in order of priority, highest to least. This is not always true, because through the use of logical

statements, you can circumvent the priority system. (We accept this inconsistency because in the real world, priorities like these are not linear or 2-dimensional, in other words, priorities vary based upon conditions and so you cannot strictly adhere to one hard priority arrangement for all cases.) However, higher-to-lower is the general ordering of prioritized Groups and Rules. The reviewer should be made aware that some lower priority objectives are placed before higher priority objectives, but in the actual process, do not govern over them. This might be made to happen by making the lower priority objective (placed at a higher priority position) check to see if certain higher-objective values are set before it sets its own value.

In RiverWare, the logic of the operating policies is included in modeling by running what is referred to as "Rulebased Simulation." In the absence of control of intakes, penstocks, slide gates, and other such managed facilities, RiverWare's "Simulation" run-mode would sufficiently model how the natural river system transports water through a system of unmanned/uncontrolled reservoirs. Conversely, with the above facilities in operation, understanding of how decisions are made must be provided for, in the form of a prioritized set of Rules. These Rules are developed using RiverWare's Rule language, "RiverWare Policy Language" (RPL). The written Rules attempt to embody and implement all of the consistent, repeatable decision-making done by water operators in actual operations. Of course, special cases and deviations are a more difficult issue, possibly not suitable for coding as Rules. The Rules themselves provide for absolute consistency in operations (although actual operations are seldom absolutely consistent), and the documentation ideally would explain all of the logic.

Some basic definitions are needed to meaningfully describe Rules (the reviewer should refer to RiverWare training and manuals to fully understand concepts): A RiverWare "object" is a graphical representation of a physical thing like a reservoir, river reach, or stream gage, that contains data, customized "methods" for doing things, and contractor "accounts". An example of an object would be El Vado Reservoir, which includes information on its elevation/storage relationship, outflow features, etc. A "slot" is the placeholder for a value, a table of values, or a set of timeseries values residing within an object, which can be filled ("set") by the user directly, by importing data, or by Rules. An example of a slot would be "Pool Elevation", which is a timeseries slot including values for the entire run period, and is generally solved for internally based upon total reservoir mass balance. User input has the highest priority in setting slots – Rules cannot overwrite what has been input by a user. In the absence of data, a slot contains a "NaN" or a series of them. This stands for "Not-a-Number", and is essentially an identifiable no-value placeholder for an empty space in a slot. Rules can recognize and overwrite NaN's. Each Rule can set one or more slots.

The need for Rules results from too many unknowns to solve the physical system. In other words, we don't know for example, the outflows from reservoirs because they are controlled by operators, and are not just dependent upon inflows and reservoir elevations. Rules step in to make decisions that physical system cannot solve.

Note that the Rules are developed as much as possible to avoid "hard-coding" numbers into them. That is, wherever a number like channel capacity flow is needed, a data object/table slot is created to hold that number, instead of typing the number into the Rule or Function code. Therefore, these numbers can be changed as studies determine necessary, to study "what-if" scenarios, and to primarily give the normal user the ability to make the change, not a Rules-Coding person. This practice is a standard good programming practice we attempt to employ, to the best usability and proper development of the Water Operations Model.

RULEBASED SIMULATION

"Rulebased Simulation" is a RiverWare term for computerized reproduction in which values undeterminable by pure "Simulation", are set by prioritized Rules. Rules typically have an IF-THEN-ELSE structure that examines the state of the system in the antecedent, and then sets values in slots accordingly. The "state of the system" is usually the result of input from forecasting. Higher priority Rules can overwrite values set by lower priority Rules, but not vice-versa. This summary provides just a general

overview on how the Rulebased simulation controller works. (A more detailed description is provided in Appendix A.)

As discussed above, the Rules are listed in order of priority. Each Rule is assigned a numerical integer value, with the lowest (1) being the highest priority Rule. The Rules are subdivided into “policy Groups,” typically by reservoir and by special functional needs. The model and Rules execute upstream to downstream, therefore, the upstream reservoir Rules have higher priority than downstream reservoir Rules. The implications of this are for example: Abiquiu’s top priority Rule is actually a lower priority than El Vado’s lowest priority Rule. This is not a problem though, because reservoirs operate somewhat independently and do not set each other’s slots. Also, another convention is used to negate priority problems: Assign responsibility to the upstream reservoir to check the downstream reservoirs’ conditions prior to doing something detrimental to downstream operations.

Rules “fire”, or in other words “execute” in an iterative fashion. The priority system gives higher Rules the ability to overwrite slots set by lower Rules, or conversely, to have the slots remain as they set them if and when lower priority Rules attempt changes. But normally, higher Rules have insufficient information to solve on the first iteration, and must re-fire after lower Rules have set some slot values. However, if a higher Rule can solve on the first pass, the lower Rules must recognize this prior to attempting to set the same slots, if they wish to set other slots the higher Rule did not set. For any Rule to be successful setting any of the slots, it must not attempt to set any slots that have been set by higher Rules. If it does, it will fail not only on that slot, but on all slots it addresses due to its junior priority. The way around this is; for each slot setting assignment in the Rule, an “IsNaN” (“Is it a NaN”) check is used to see if a number has been assigned to that slot. If it “IsNaN”, then an assignment will be attempted, but if it is a number, no attempt will be made and the Rule will move on to the remaining slots (instead of completely failing and kicking completely out of the Rule). For both the checked former slot and the subsequent slots it will be fully successful. Set slots remain set until the next timestep, unless overwritten by higher priority Rules, and assume the priority of the Rules that set them. Rules are best named descriptively; for readability of the code, for example; “AbiquiuFloodControl”.

FUNCTIONS – GENERAL DISCUSSION

Functions are named logical and mathematical expressions of code called by Rules or other Functions, which return one value each time they are called. The values they return reside only temporarily in memory, and disappear as soon as they provide their values to the calling Functions or Rules. Functions cannot set slots, but are used by Rules to do so. For example, if the Function “ElVadoGRRelease ()” is called, its logic goes to work using current values in slots it refers to and using the results of other Functions it calls to return for instance; “500 CFS”. This number is invisible to the user, unlike if written to a slot, and is just provided to the caller, and purged. A Function is also named (by the user) descriptively, so the user can understand what the number returned is, (in the example case; the amount of Rio Grande water to be released from El Vado).

There are two types of “Internal Functions” used by RiverWare: “Pre-Defined Functions” provided by CADSWES for general RiverWare use, such as; “FlowToVolume ()”, which converts a flow in CFS to a volume per timestep in AF/timestep; and “User-Defined Functions” such as “HeronOutflow ()”. The Functions we are documenting here are User-Defined Functions. User-Defined Functions were written by the URGWOM development team (including consultants). (For documentation on RiverWare Pre-Defined Functions, refer to RiverWare help, which is launched from RiverWare.)

Using Functions is often optional, as the Rules can usually include all of the lines of code. (In fact, the code syntax in Functions is almost identical to that used in Rules.) However, this would result in long Rules that are difficult to read and decipher. Functions allow the subdivision of Rules into shorter, more readable groupings of code. More importantly, for code that is used repeatedly, having one Function called by name numerous times is much more efficient than rewriting all its code in every spot it is needed. Use of Functions also makes debugging of Rulesets easier, as you can trace the calling of

Functions easier than through lengthy Rule statements. Functions and Rules are the fundamental building blocks of a Ruleset.

RESERVOIR OPERATIONS

This section is dedicated to explaining the workings of the system from the operators' perspective. Following are the current policies and typical operations for each reservoir project that are implemented in the model.

HERON RESERVOIR

Heron Reservoir is operated in accordance with P.L. 87-483 and in compliance with the Rio Grande Compact. Two basic principles control the water release schedule from Heron Reservoir. The first is the authorized development of San Juan-Chama Project supplemental irrigation and municipal and industrial water demands that result in increased depletion of the Rio Grande. These depletions are offset by releases of San Juan-Chama water from Heron Reservoir sufficient to assure that no residual effect occurs to natural waters of the Rio Grande due to project operations. In addition, downstream contractors such as the City of Albuquerque and the Middle Rio Grande Conservancy District (MRGCD) convey other project waters past Otowi for use. No Rio Grande water can be stored in Heron, therefore it is released as soon as it is known of per the accounting procedures, and it is feasible to release it. (This is usually after the end of the month, because the accounting-determined amount is not known until the month ends.)

Secondly, carry-over storage is not permitted in Heron Reservoir (unless by "Waivers", as discussed later) from one year to the next. Contracted water not called to be released by December 31, would remain in Heron Reservoir as part of the project supply and would no longer belong to the individual contractor. In the past, Reclamation negotiated temporary waivers with contractors that allowed carry-over until April 30, in order to provide release rates on the Rio Chama that would enhance the fishery between El Vado and Abiquiu Reservoirs during the winter and provide flexibility in managing river flows. The no carry-over stipulation results in the various contractors seeking storage in reservoirs downstream of Heron for their unused water. El Vado, Abiquiu, Jemez Canyon, and Elephant Butte reservoirs have been used for storage of San Juan-Chama waters. Another factor that influences Heron releases is ice cover on the reservoir and the resulting safety issues. If Heron is drawn down too quickly when iced over or nearly completely iced over, hazardous conditions develop. Releases would be terminated until conditions are safe. During late March or April, any San Juan-Chama Project water not released because of unsafe winter operation conditions, will be released at a time when it is assured to meet the same purposes as if it had been released during the winter months, provided the necessary waivers have been granted.

The Natural Resources Conservation Service/National Weather Service (NRCS/NWS) coordinated runoff forecast is used to estimate the period of time during the spring runoff that the flow of the Rio Chama is expected to exceed channel capacity below Abiquiu Reservoir, thus preventing a release of San Juan-Chama Project water from Abiquiu Dam. Between the times of ice melt on the reservoir and the times when spring runoff reaches full channel capacity, San Juan-Chama replacement water requirements downstream are estimated and are released from Heron Reservoir.

EL VADO RESERVOIR

El Vado Dam was originally constructed to provide conservation storage for a supplemental irrigation supply for the MRGCD lands along the Rio Grande from Cochiti Dam to below Socorro, New Mexico. Because El Vado Dam was constructed after 1929 (completed in 1935), the operation of the reservoir for storage and release of Rio Grande water is subject to Prior and Paramount (Indian Water Rights) storage and releases, and the Rio Grande Compact. Water imported into the Rio Grande basin via the San Juan-Chama Project and stored in El Vado Reservoir is not subject to storage and release restrictions of the Rio Grande Compact.

Operation for Rio Grande Water

The basic concept in operating El Vado Reservoir involves the storage of natural inflow that is in excess of current MRGCD and other needs below El Vado Dam. The major storage season is during the spring runoff, and then storage can be released during the irrigation season to users in the Middle Rio Grande Valley as needed. In accordance with Prior And Paramount requirements, a monthly-varying minimum pool of Rio Grande water is held in storage from March through October, with higher priority than any other requirements. This amount is adjusted monthly based upon the NRCS/NWS March, April, and May Otowi forecasts, amounts of Rio Grande captured in El Vado, and calls for the water by the Pueblos.

Restrictions of the Rio Grande Compact

Article VII of the Rio Grande Compact provides that no storage of Rio Grande water in El Vado Reservoir can take place when usable water in Project Storage (non-credit-water and non-SJ-C Contractor-water storage in Elephant Butte and Caballo Reservoirs) is less than 400,000 acre-feet (except for Prior and Paramount waters). Article VI provides that any Rio Grande water stored in El Vado Reservoir must be held in storage to the extent of New Mexico's accrued debit under the Compact. Nearly all of the post-compact storage capacity in New Mexico is located in El Vado Reservoir.

Water Right Constraints on Operation of Rio Grande Water

El Vado is operated to store native water (Prior and Paramount) for the Six Middle Rio Grande Pueblos of; Cochiti, Santo Domingo, San Felipe, Santa Ana, Sandia and Isleta. The Bureau of Indian Affairs and the Bureau of Reclamation compute the amount of storage required and release of Indian storage is made only when the natural flow of the Rio Grande is insufficient to adequately supply irrigation to 8,847 acres of Indian lands.

Additionally, no storage of native water except for Prior and Paramount can be made at El Vado Reservoir when to do so would deprive acequias along the Rio Chama downstream of El Vado of water to which they are entitled. In 1971, the State Engineer required that El Vado Reservoir be operated during the irrigation season to pass all the natural flow of the Rio Chama up to 100 cfs, as determined at Abiquiu Dam, during the irrigation season.

Operation for San Juan-Chama Water

El Vado Reservoir operation is affected by the San Juan-Chama Project in two ways. The first is that San Juan-Chama Project water released from Heron Dam for use downstream of El Vado Reservoir is simply passed through. The second is that storage of large volumes of San Juan-Chama Project water in El Vado Reservoir may take place for extended periods of time. The MRGCD has contracted for 20,900 acre-feet per year of San Juan-Chama Project water and maintains as much of this water in El Vado Reservoir as conditions permit. In addition, the MRGCD has contracted with various contractors of San Juan-Chama Project water to allow for those contractors to store in El Vado Reservoir.

ABIQUIU RESERVOIR

Operation for Rio Grande Water

Abiquiu Dam and Reservoir is operated for flood and sediment control in accordance with conditions and limitations stipulated in the Flood Control Act of 1960 (PL 86-645). Reservoir regulation for flood control is also coordinated with the operation of Jemez Canyon Reservoir, Cochiti Lake, and Galisteo Reservoir (an ungated structure). Abiquiu Reservoir is operated to limit flow in the Rio Chama, insofar as possible, to the downstream channel capacities of 1,800 cfs for the reach below Abiquiu Dam, 3,000 cfs for the reach below the mouth of the Rio Ojo Caliente, and on the Rio Grande main stem, 10,000 cfs below the mouth of the Rio Chama. Irrigation releases from El Vado Reservoir are passed through the reservoir. Typically, if Rio Grande inflows exceed downstream channel capacities during April and May, Abiquiu captures this peak of snowmelt runoff, and releases it during June and early July. However, any Rio Grande storage remaining after the natural flow at Otowi drops below 1,500 cfs (July 1st or later) is carried over (see Flood Carryover Storage section below) and not released until November 1st or later.

Operation for San Juan-Chama Water

In 1981, PL 97-140 authorized the storage of 200,000 acre-feet of San Juan-Chama water in Abiquiu Reservoir. The City of Albuquerque has obtained a storage easement to elevation 6,220 feet. Real estate interests have not been obtained above elevation 6,220 feet to accommodate the full 200,000 acre-feet as authorized. The San Juan-Chama capacity is annually reduced due to the estimated sediment deposition. San Juan-Chama storage is held below elevation 6,220 feet and released as requested by the storing entities, except when releasing Rio Grande water at channel capacity (due to its flood-control priority). The San Juan-Chama pool also serves to increase the sediment trap efficiency, and enhance the recreation, fish and wildlife opportunities in the reservoir.

Flood Carryover Storage

Abiquiu Reservoir is required to retain any floodwater remaining in storage after July 1st of each year, when the natural inflow to Cochiti Lake is less than 1,500 cfs and so long as there is at least 212,000 acre-feet of capacity available at Cochiti Lake to control runoff from summer thunderstorms. This floodwater storage is the result of high inflows coming into the reservoir during spring runoff. As previously stated, the Corps has downstream channel capacity restrictions that prohibit passing all of the natural inflow. The Rio Chama has a channel capacity of 1,800 cfs below Abiquiu Reservoir, flows must not exceed 3,000 cfs at Chamita gage and there must be no more than 10,000 cfs at the Otowi gage. These restrictions result in temporary storage of natural water as well as releasing as much as downstream restrictions allow. Depending on the volume of water from spring runoff, Abiquiu Reservoir has either been able to safely pass inflow without any carryover or has locked-in as little as 3,500 (1994) acre-feet to as much as 212,000 acre-feet (1987).

The natural water that is locked-in must remain in storage until the end of irrigation season (November 1). Any natural inflow to the reservoir during the lock-in period is passed through the reservoir. After October 31st, the Corps can release the carryover, and release rates are coordinated with state and other federal agencies, and are generally used to maintain minimum flows during the winter.

Conservation Storage

In 2000, an agreement was formed to allow for up to 3-years of storage of natural flow in Abiquiu (and Jemez Canyon) to help maintain middle valley flows during the latter part of the summer and beyond, for silvery minnow habitat. This temporary agreement was implemented in the model by creating another Rio Grande Account in Abiquiu (and in Jemez), and by creating Rules understanding how to store and release this water. Flow diverted to the conservation water pool can only be stored when all downstream demands are met. Releases are made when there is not enough natural flow and San Juan-Chama water combined to meet downstream demands.

COCHITI RESERVOIR

Operation for Rio Grande Water

Congress authorized Cochiti Dam in 1960 for flood and sediment control. Operating Rules specified in P.L. 86-485 provide that the Dam is to be operated to bypass the maximum possible rate of flow that can be carried in the channel through the middle valley without causing flooding. Cochiti Lake flood control operations are primarily focused on the Middle Rio Grande Valley, controlling flows between Cochiti Lake and Elephant Butte Reservoir to current maximum channel capacities of 7,000 cfs in Albuquerque and about 4,500 cfs at the San Marcial Railroad Bridge.

As with Abiquiu, when inflow exceeds the capacity of the downstream channel, storage is retained in the reservoir and held until downstream channel conditions allow for its release, provided that, after July 1, there is 1500 cfs or more of native inflow and that a minimum of 212,000 acre-feet of storage is available in Cochiti Reservoir to control summer flood flows. Flood Carryover storage that is "locked-in" is released beginning November first. (See discussion under Flood Carryover storage at Abiquiu Reservoir).

An algorithm for balanced flood releases between Cochiti and Jemez, when there is flood storage in both reservoirs is given in their respective Water Control Manuals (WCM). Use of the balanced operation tends to cause fluctuations in each reservoirs release to maintain flood control pools.

Operation for San Juan-Chama Project Water

Public Law 88-293 authorized the release of 50,000 acre-feet of San Juan-Chama Project water for the initial filling of a permanent pool of 1200 acres in Cochiti Reservoir, and thereafter, sufficient water annually to offset the evaporation from such area. A portion of the release of San Juan-Chama Project water is used to offset evaporation loss from the water surface of a small wetland on the Santa Fe River above Cochiti Dam. San Juan-Chama Project water destined for Elephant Butte or as supplemental irrigation water for MRGCD is passed through Cochiti.

JEMEZ CANYON RESERVOIR

Jemez Canyon Dam and Reservoir was authorized by the Flood Control Act of 1948, and is operated in tandem with Cochiti Lake to control flows through the middle valley. In 1979, a sediment control pool was established and maintained within that portion of the reservoir capacity allocated for sediment deposition. The water stored in the sediment control was completely evacuated by 2001. Flood storage, if any, is accumulated atop the sediment control pool and released as soon as possible thereafter. Jemez Canyon Reservoir is operated to prevent carryover storage of floodwater, and in conjunction with Cochiti to prevent flows exceeding downstream channel capacities. Currently there is no continuing authorization to operate Jemez for sediment retention, so it is essentially a dry reservoir.

As stated in the Cochiti operations section, balanced flood operations are attempted between Jemez and Cochiti. Also, refer to the above write-up on Abiquiu for the temporary "Rio Grande Conservation Storage" management implemented in the year 2000.

ELEPHANT BUTTE RESERVOIR

Operation for Rio Grande Water:

Elephant Butte Reservoir is the principal storage facility for the Rio Grande Project, delivering stored water for downstream use under contract between the Bureau of Reclamation and the Elephant Butte Irrigation District in New Mexico and the El Paso County Water Improvement District No. 1 in Texas. Elephant Butte Reservoir is also operated to ensure that the U. S. 1906 Treaty obligation with Mexico to deliver 60,000 acre-feet per annum at the Acequia Madre headgate in Mexico can be met. The river channel below Elephant Butte has a design capacity of 5,000 cfs.

Operation for San Juan-Chama Project Water:

In 1981, Congress authorized the Secretary of the Interior to enter into contracts for storage of San Juan-Chama Project water in Elephant Butte Reservoir. This Act (P. L. 97-140) provided that the amount of evaporation loss and spill chargeable to San Juan-Chama Project water shall be accounted under procedures established by the Rio Grande Compact Commission.

San Juan-Chama Project Water may also be stored in Elephant Butte Reservoir for recreation purposes. Originally established at 50,000 acre-feet, water in the recreation pool has been substantially diminished in size because of spill of this water from Elephant Butte Reservoir. This spill resulted from the Rio Grande Compact Commission accounting requirements that SJ-C water is to spill before native water.

CABALLO RESERVOIR

Caballo Reservoir is a re-regulating reservoir that works in conjunction with Elephant Butte Reservoir. Water released from the Elephant Butte power plant during winter generation is impounded for irrigation use during the following summer. The International Boundary and Water Commission (IBWC) manages

flood control operations in and below Caballo Reservoir. Caballo is authorized for 100,000 acre-feet of flood control storage to limit flow in the Rio Grande to no more than 11,000 cfs at or below American Dam in El Paso, Texas.

GENERAL RULES DESCRIPTION

GENERAL

Following are descriptions of the general logic of the Rules for each project. More detailed descriptions (comments within the Ruleset) of each Rule and Function are given in the Specific Rules Descriptions nestled within the code.

The basic solution methodology of the water operations model is top down. In other words, the model solves in a downstream direction from the San Juan-Chama Project diversions above Heron Reservoir (Rio Chama) and the Lobatos gage (Rio Grande) near the Colorado/New Mexico border down to and including Caballo Reservoir and on to El Paso.

As a general Rule, each reservoir makes a "total release" decision based upon the physical system. In this case "total release" is defined as both San Juan-Chama (SJ-C) and Rio Grande. This total release is then reconciled with the accounting system (which keeps track of the different types of water and contractors). Reconciliation of the accounting system to the physical system is made by first releasing any Rio Grande water that was computed that needed to be released. Any remaining water is assumed to be SJ-C. In case that the physical "total release" is greater than the sum of the Rio Grande and SJ-C release, the excess release is determined to be Rio Grande. If the total release is less than the computed Rio Grande release, all release is Rio Grande. Rules then allocate and distribute each of the different types of water (SJ-C and Rio Grande) to individual contractors based upon a priority system involving contractor, release type, and in the case of Heron, destination. A discussion on how the priority system works to distribute contractor SJ-C water is presented below.

EXCHANGES, CREDITS AND DEBITS

Rules are also used to track exchanges of water for Rio Grande depletions or between accounts. These exchanges also keep track of credits and debits between various accounts. For example, Nambe Falls involves an exchange of Rio Grande water for SJ-C water. Nambe Falls is not currently modeled in URGWOM, but there is a mechanism (an account) in the model and Ruleset to release exchange water for the Nambe Falls project. Release of SJ-C water to offset depletions of Rio Grande water caused by the operation of the Nambe Falls Project must be made as soon as practical from the Nambe Falls SJ-C account in Heron to replace the depleted Rio Grande water. This water must be delivered to the Rio Grande at Otowi. This is modeled in the URGWOM model by inputting historical patterned depletions. When such an input depletion is encountered by the Rules, a series of exchanges are set up. The first exchange is set up between Otowi and the Albuquerque Abiquiu account. This exchange is seen as a demand for water by the Albuquerque Abiquiu account and if there is enough storage available in Abiquiu and the Nambe Falls Heron account has enough water to pay Albuquerque Abiquiu account back, Albuquerque Abiquiu account releases enough water to offset the depletion, by exchange. At that point, another exchange is set up between Nambe Falls account at Heron and Albuquerque Account in Abiquiu that will entirely repay the Albuquerque Abiquiu account. This second exchange is examined by the Rules computing Heron Reservoir's release and eventually water is released from Nambe Falls Heron account in order to repay its debt to Albuquerque Abiquiu. This same process is used to propagate demands for every account with the exception of MRGCD, Reclamation's Supplemental Water Program (SWP), and Cochiti Recreation Pool. Demands for these accounts are directly input or computed by the model.

SAN JUAN DIVERSION RULES

The SJ-C diversions Rules are the first to execute because they are at the top of the system. These Rules determine how much water will be diverted from the San Juan River Basin into the Rio Grande Basin based on statutory conditions and operating practices and priorities. Statutory conditions providing the bases for Rules set are, water availability based on bypass requirements, annual diversion volume limitation, and decade diversion volume limitation. Operating practices and priorities that are included as Rules are tunnel and feeder capacities and operating priorities based on diversion levels, and available storage capacity available in Heron. After the SJ-C diversion Rules execute, both the SJ-C and Rio Grande or “total inflow” into Heron may then be determined.

HERON RESERVOIR RULES

The Rules for Heron then determine what the “total outflow” (the physical outflow) from Heron will be. Initially this is computed as the sum of the Rio Grande Release plus any SJ-C releases. The Rio Grande release is determined by looking at the Rio Grande inflow into the reservoir as well as any incidental storage of Rio Grande water in Heron. The goal for the operation of Heron is to avoid impairment of rights to native water. In other words, release any Rio Grande water as soon after it comes into the reservoir as possible. In the absence of knowing the actual daily Rio Grande storage amounts, the Rules implement a system of releases during the course of the month under a variety of considerations to ensure Rio Grande storage is released or maintained at very low levels.

The total SJ-C release is determined by several factors. The SJ-C release factors are; maintaining the minimum release below El Vado, rafting releases, whether a waiver has been granted to allow carry-over storage and delivery of contractor allocations. After the initial outflow is determined, the higher priority Rules for Heron check other physical factors such as the need to spill, minimum and maximum storage, maximum allowable draw-down conditions, reservoir ice cover and maximum release. This checked physical release is then reconciled against the accounting system. These reconciled Rio Grande and SJ-C releases are then distributed to specific accounts based upon the priority system described in the San Juan–Chama Release Priorities section below.

EL VADO RESERVOIR RULES

After Heron Rules have executed, the El Vado total inflow and the breakdown of the total inflow between native and SJ-C water is known. Once again, an initial total outflow is computed by the sum of the Rio Grande and SJ-C releases. The Rio Grande release is computed as the maximum of the MRGCD demand (not met by downstream Rio Grande inflow) and a target storage algorithm. Target storages are input by the user. To meet a storage target, reservoir inflow is forecasted and the Rules release water in excess of storage targets. The Rio Grande release is also constrained by Pueblo Indian water storage and release requirements and time dependent (seasonal or monthly) minimum releases. The SJ-C release is computed by the summation of any flow through accounts, exchange debts to Otowi, exchange debts to accounts in Abiquiu, rafting releases, and minimum flows.

This initial total outflow is then checked by higher priority Rules to ensure compliance with flood control criteria based on maximum pool elevations, outlet works capacity, downstream channel capacity, available capacity of Abiquiu Reservoir and the Rio Grande Compact requirements of Article VI, (retain debit water in storage in post compact reservoirs), Article VII (no increase in storage when “Usable Water in Project Storage” is less than 400,000 acre-feet) and Article VIII (Texas call for release of debit storage) to determine a final total release. The total outflow is then reconciled to the accounting system and the native and SJ-C releases for individual contractors are determined by a priority system.

ABIQUIU RESERVOIR RULES

After El Vado Rules have been executed and it’s outflow computed, Abiquiu total inflow might then be determined. The initial total outflow is determined by summing the Rio Grande release and the SJ-C release. As Abiquiu cannot (in normal circumstances) store any Rio Grande water, the Rio Grande outflow is computed by looking at the amount of Rio Grande inflow and releasing the inflow and any

incidental Rio Grande storage. In instances where releases would exceed downstream channel capacity, Rio Grande storage is induced. This storage is "locked-in" when the flood control criteria of Public Law 86-645 are invoked. These are, that during the July 1st through October 31st period, flood waters are retained in storage when there is a minimum of 212,000 acre-feet of vacant capacity in Cochiti Reservoir, and the natural inflow to Cochiti Reservoir (sum of Rio Chama near La Puente and Rio Grande at Embudo) is less than 1,500 cfs. This flood carryover storage is released after October 31, and must be completely evacuated by March 31 of the following year. The default in the Rules is to compute a constant release for the period of November 1 through March 31 of the flood carryover storage available on October 31. The user can override the default method by directly inputting the flood carryover release per preferred schedule.

The SJ-C release for Abiquiu is computed by summing any flow through accounts, meeting any exchange demands, and by meeting computed MRGCD, Albuquerque, and Reclamation SWP demands below Cochiti. The SJ-C release is also constrained by a user input maximum SJ-C release. After the initial total outflow is computed, it is checked to see that it conforms to flood control criteria spillway, (outlet works and downstream channel capacity), outflow restrictions (stepped release etc.), and minimum flow requirements for tailwater fishery. This checked total outflow is then reconciled to the accounting system and distributed to individual accounts by the priority system.

COCHITI LAKE RULES

Cochiti's operation is very similar to Abiquiu's. Its main criterion is to release any Rio Grande water and incidental storage and meet user input demands for MRGCD, Albuquerque, and Reclamation's SWP. It also uses the priority system to distribute water to individual accounts, such as a pass-through account for delivering water to downstream users such as MRGCD.

JEMEZ CANYON RESERVOIR RULES

Jemez's operation is very similar to Abiquiu and Cochiti's. Its main criterion is to release any Rio Grande water and incidental storage. It is also set up to operate to capture water for a sediment pool in the allocated sediment space. Water for the earlier sediment pool and evaporation replacement were exchanged from RG to SJ-C, along with a release from Heron that's exchanged from SJ-C to RG at the RioGrandeJemez confluence. As of 2000, the authorization for the sediment pool has expired, and it is not currently in operation. The Rules are still set up allow a sediment pool, if so desired.

ELEPHANT BUTTE RESERVOIR RULES

Since only flood control operations are allowed to be incorporated in the Rules at and below Elephant Butte at this time, irrigation releases are directly input (in a data object). The flood control Rules will override these releases when the content of Elephant Butte Reservoir exceeds the prudent flood space criteria (50,000 AF vacant from April through September and 25,000 AF from October through March).

CABALLO RESERVOIR RULES

Caballo's operations are set up similarly to Elephant Butte's; irrigation releases are directly input (in a data object), and flood control Rules only override releases when flood conditions exist. Caballo's Flood Operations Criteria is a release diagram based on 2-hour instantaneous inflow rates, compromising its application in a daily model. Therefore, the current Ruleset releases one-half of the inflow when in flood operations. (It would be preferred to use a release diagram that approximates the 2-hour release diagram releases on a daily basis, but currently only one timestep period can be used per instance of RiverWare).

SAN JUAN-CHAMA RELEASE PRIORITIES

As mentioned earlier, reconciled total accounting releases are distributed by a priority system. This priority system uses 2 to 3 different sets of priorities (depending upon the reservoir) to determine which account(s) will release water. These sets of priorities include release type, destination, and account.

The highest priority of these sets of priorities is the release type priority. For example, the release type priorities for Abiquiu are FlowThrough, OtowiPaybacks, AlbuquerqueLoan (Albuquerque will let other contractors borrow from them), CochitiRecPoolPayback, MRGCD, and Reclamation. Each of these release types has a priority that can change through the year and has from one to fifteen different accounts associated with it. For example, the OtowiPaybacks release type has seven different accounts associated with it (Albuquerque, Bernalillo, Espanola, Los Alamos, Santa Fe, Taos, and Twining). Each of these associated accounts has a storage pool in Abiquiu and therefore, they can be used to pay any debt to Otowi from Abiquiu directly.

The next highest priority set (if used) is the destination priority set, of which only Heron Reservoir uses. The destination priority set is used on Heron to determine whether it is a higher priority to send water to accounts in El Vado or in Abiquiu. Heron releases are somewhat unique, in that there are no direct demands for releases from Heron (all direct demands are made from El Vado or Abiquiu and releases from Heron are made to repay El Vado or Abiquiu. It can be set up to, in effect, cause the demands to be met directly from Heron though). Releases from Heron are generally timed to help El Vado to maintain its minimum release, help produce rafting releases, or to make specified delivery of a contractors allocation, if maintaining minimum releases and rafting releases do not require enough water to force contractors water out of Heron. This priority set, like the others, can change up to 12 times through the year.

The lowest priority set is the account priority set. Each storage account on a reservoir has a priority that can change up to 12 times through the year. This priority set is used to determine which particular account will receive water if SJ-C releases are not sufficient to meet all demands for a release priority. To illustrate how the priorities work, several assumptions about the state of the model are made, as described as follows.

PRIORITY SOLUTION EXAMPLE

Assumptions:

Assume on a given day that the reconciled SJ-C release from Abiquiu is 500 cfs, of which 300 cfs is flow through water (water released from either El Vado or Heron going to a destination downstream of Abiquiu) passing through Abiquiu, MRGCD owes 150 cfs to the CochitiRecPool, Albuquerque owes 10 cfs to the CochitiRecPool, Bernalillo has a debt at Otowi of 150 cfs, and Espanola has a debt of 25 cfs at Otowi. Additionally, there is no borrowing from the Albuquerque pool. The priorities of each release type and account for Abiquiu are shown in Tables 1 and 2, respectively.

Table 1 - Release type assumed priorities for Abiquiu Reservoir

RELEASE TYPE	Effective through Julian Day				
	74	90	212	304	365
	<u>Priority</u>	<u>Priority</u>	<u>Priority</u>	<u>Priority</u>	<u>Priority</u>
Reclamation S.W.P.	6	6	6	6	6
OtowiPaybacks	4	4	4	4	4
Albuquerque Borrow	2	2	2	2	2
FlowThrough	1	1	1	1	1
NMISC	5	5	5	5	5
CochitiRecPoolPayback	3	3	3	3	3
MRGCD	7	7	7	7	7

Table 2 - Account priorities for Abiquiu Reservoir

ACCOUNT	Effective through Julian Day				
	74 Priority	90 Priority	212 Priority	304 Priority	365 Priority
Albuquerque	2	2	2	2	2
Bernalillo	5	5	5	5	5
Espanola	3	3	3	3	3
Los Alamos	4	4	4	4	4
MRGCD	1	1	1	1	1
Reclamation S.W.P.	6	6	6	6	6
Santa Fe	7	7	7	7	7
Taos	8	8	8	8	8
Twining	9	9	9	9	9
Belen	10	10	10	10	10
Los Lunas	11	11	11	11	11
Nambe Falls	12	12	12	12	12
Red River	13	13	13	13	13
Uncontracted	14	14	14	14	14

Solution

The first thing to happen is that all debts or demands associated with a particular release type are met in order of priority. From Table 1, it can be seen that the first priority for release type in Abiquiu is FlowThrough. From the assumptions, there is 300 cfs of flow through water and therefore, all 300 cfs of flow through releases would occur. This leaves 200 cfs to distribute to the other priorities (the total of 500 cfs less the 300 cfs of flow through). The second release type priority is Albuquerque Borrow, which has been set to zero acre-feet, so 200 cfs still remains to be distributed. The third release type priority is CochitiRecPoolPayback, which has two accounts associated with it (MRGCD and Albuquerque). MRGCD has a debt of 150 cfs to the CochitiRecPool, and from Table 2 it can be seen that MRGCD has the highest account priority (between MRGCD and Albuquerque), so the entire 150 cfs debt would be met leaving 50 cfs to be distributed. Albuquerque also owes the CochitiRecPool 10 cfs, therefore, it would be the next priority and its debt would be met in its entirety. This would leave 40 cfs to be distributed to other priorities. From Table 1, the fourth priority release type is OtowiPaybacks. For Abiquiu this priority has seven accounts associated with it (Albuquerque, Bernalillo, Espanola, Los Alamos, Santa Fe, Taos, and Twining). Espanola has a debt of 25 cfs to Otowi, and from Table 2 it can be seen that Espanola has a priority of 3, which is higher than the priority of 5 for Bernalillo. Therefore, Espanola would be met first and completely, leaving just 15 cfs to use toward meeting the 150 cfs debt that Bernalillo has at Otowi. So on the next time step, the Bernalillo debt at Otowi would be 150 cfs less 15 cfs (or 135 cfs) plus any additional debt that it may accumulate on the next time step.

APPENDIX A – SIMULATION AND RULEBASED SIMULATION OVERVIEW

INTRODUCTION

Rulebased Simulation can only be understood once the fundamentals of basic Simulation have been mastered. It would be impossible to describe how the URGWOM model will be implemented within the RiverWare framework without an understanding how both simulation and Rulebased simulation work within RiverWare. This section is a brief introduction to how these two *controllers* are *currently* implemented.

A RiverWare model of a river system consists of objects such as reservoirs, river reaches, diversions, confluences and water users, which are linked together to form a network. Each object “contains” physical process models and data to support those models. The links between the objects represent flow continuity, and in some cases, water surface elevation dependencies between the objects.

SIMULATION

In simulation, the physical process models range from simple linear mass balance equations to complex nonlinear processes such as dynamic routing, flow-varying losses, hydropower generation, release and spill Functions, consumptive use, and others. The user tailors the process models to meet the needs of the model and to be consistent with the size of the computational timestep. The tailoring is accomplished by selecting methods from a menu of possible model methods. For example, on a river reach object, a method of routing is selected. The simulation solution is driven by input data that provides enough known values to solve for the unknowns.

There are various solution methods (Functions) on an object corresponding to the various combinations of inputs and outputs that can solve the basic equation of the object. For a reservoir, the simplified basic equation of conservation of mass is:

$$\text{Storage}_t = \text{Storage}_{t-1} + \text{Inflow}_t \Delta t - \text{Outflow}_t \Delta t$$

Assuming that the previous storage (Storage_{t-1}) is known, the equation has three unknowns: Inflow, Outflow and Storage at the current timestep. Three solution methods are needed, each with solution conditions consisting of the known and unknown slots as follows:

Solve for Storage given Inflow and Outflow
Solve for Inflow given Storage and Outflow
Solve for Outflow given Storage and Inflow

Each object “knows” about its own data and solution methods and recognizes when it receives a new value in any one of its slots. A slot can get a new value from three sources: user input, a value propagated across a link from another object, and output set by the physical process methods. Whenever an object gets a new slot value, it compares its current set of known and unknown slots with the solution conditions for all the solution methods. If a set of solution conditions are met, that solution method is executed. For example, if a reservoir’s storage slot is input and it receives inflow from a link propagation from an upstream object, then it will execute a method which solves for outflow. When the outflow slot is set, the value will propagate downstream if that slot is linked to the inflow of a downstream object.

Too much (conflicting) information results in an error which terminates the run; not enough information results in parts of the model left unsolved. Diagnostic and run analysis utilities alert the user to these situations and help to identify the exact location of the data problems. In the cases where multiple slot linkages make the objects mutually dependent on a solution, the objects iterate until a solution meets convergence criteria.

The objects solve at whatever timestep they receive new information, allowing some flexibility in specifying models in which the solution is not strictly propagating from upstream to downstream and forward in time. River reaches with time lags may solve for inflow given outflow, setting the inflow value at a previous timestep and propagating that value upstream. Target operations on reservoirs are solved to meet a future target storage by adjusting the reservoir's outflow over a specified range of timesteps.

Simulation allows the user to control the modeled operations by inputting reservoir releases, storage levels, or other data, directly resulting in a solution. This scenario-based modeling is especially useful for fine-tuning daily operations or predicting the response of the system to particular inputs for special studies.

EXAMPLE OF SIMULATION

Simulation in RiverWare is best explained by the use of a simple example model. Consider the model illustrated in Fig 1. The model consists of four objects, 2 reservoirs (StorageReservoir0 and StorageReservoir1), Reach0, and Confluence0. The outflow of StorageReservoir0 is linked to the inflow of Reach0, the outflow of Reach0 is linked to inflow1 of Confluence0 and the outflow of StorageReservoir1 is linked to inflow2 of Confluence0. Linking in RiverWare means that if a value is set either by user input or by the solution of the objects mass balance, that value is propagated along the link to whatever object that it is linked to. An example would be that if the outflow of StorageReservoir0 gets set, that same the value would then be propagated to the inflow of Reach0 or visa versa. To successfully run simulation within RiverWare, the model must be completely determined, meaning that the user must supply enough information so that all objects in the workspace have enough information or will obtain enough information to compute its mass balance.

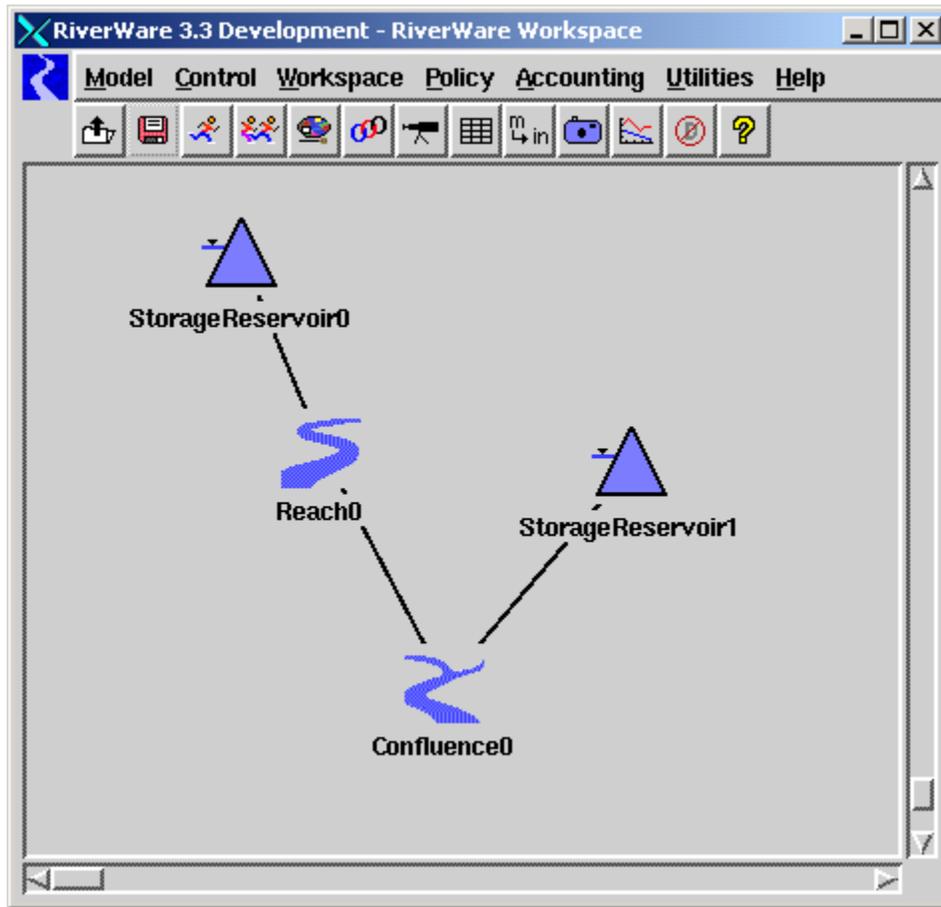
To illustrate how the simulation controller in RiverWare works, two different situations relating to the model in **Fig. 1** will be presented:

1. Solving Downstream
2. Solving Upstream.

Solving Downstream - To illustrate solving downstream, suppose that a user wants to use the model in **Fig. 1** to compute what the flow at Confluence0 would be, given that the user knows the inflow into each reservoir and there are storage targets on each reservoir. To solve this problem the user would have to supply both reservoirs, their inflows, initial storages, and their current storages (or storage targets). The model would then be completely determined and would solve in a downstream direction. The model would solve in a fashion similar to the following:

Since, both StorageReservoir0 and StorageReservoir1 reservoirs now know what their inflow, and previous and current storages are, both reservoirs would have enough information to mass balance and they would both solve for their outflows. The outflow of StorageReservoir0 would then propagate to the inflow of Reach0 and the outflow of StorageReservoir1 would propagate to inflow2 on Confluence0. Since Reach0 now knows its inflow, it has enough information to mass balance and compute its outflow, which is then propagated to inflow1 on Confluence0. Now Confluence0 knows inflow1 and inflow2 so it can compute its outflow and the model is completely solved.

Figure 1 - Example Model For Demonstrating The Simulation Controller In RiverWare



Solving Upstream - To illustrate solving upstream, suppose that a user wants to use the model shown in Fig 1 to compute what the inflow into each reservoir would need to be in order to meet a particular outflow and inflow1 requirements at Confluence0. Additionally, the storage at each reservoir is fixed due to storage restrictions. From these assumptions, it should be apparent that the only free variables are the inflows to each reservoir. To solve this problem the model needs solve upstream. The model would solve in a fashion similar to the following:

Since, Confluence0 knows its outflow and inflow1 it has enough information to mass balance and solve for inflow2. Inflow2 would get propagated to the outflow of StorageReservoir1, now StorageReservoir1 knows both its outflow and storage, so it would solve for its inflow. Inflow1 from Confluence0 would get propagated up to the outflow of Reach0 giving Reach0 enough information to solve for its inflow which would then be propagated up to the outflow of StorageReservoir0 which now knows its outflow and storage and so could solve for its inflow and the model is completely solved.

NOTE: Depending on which routing method is selected for Reach0, the model may not be able to solve upstream. Some routing methods (such as the variable time lag) cannot solve upstream.

From the above two examples of how the simulation controller within the RiverWare works, it should be readily apparent RiverWare can easily solve in any direction depending upon what inputs that the user provides to the model. It should also be apparent that this type of simulation has limited applications, as the user, via inputs to the model, determines all of the operational policy and a workable solution is

usually found by many iterative runs with the user changing inputs until the desired behavior of the river basin system is achieved. The use of straight simulation makes it very difficult, if not impossible, to compare how two different operational criteria would change the operation of a river basin over a long period of time because the user essentially has to supply every decision at every timestep through the entire simulation.

RULEBASED SIMULATION

Rulebased simulation solves in a similar fashion to simulation except that before the run, the system is under-determined, i.e., there are not enough known variables to solve the system. Whenever no objects can solve, control is passed to the Rule processor, which executes the prioritized Rules one at a time. Each Rule has the opportunity to examine the state of the system and set values of decision variables according to the logic of the operating policies. As a result, the objects have additional information, which can be used to further the simulation. Even after the objects have solved one or more times, higher priority Rules can reset the values of variables that have already been set by lower priority Rules or by simulating the effects of other Rules.

Rulebased simulation is the controller in RiverWare that directs the simulation of a river basin network based on user inputs and specified operating policies or Rules. With Rule based simulation, the simulation behavior of the river basin network can be modified via the Rules without recompiling the RiverWare framework code.

There are three main components that make up Rulebased simulation within RiverWare; a Rule language, a Rule processor and a Rulebased simulation controller. The first component is used directly by RiverWare to create, modify and debug Rules. The remaining two components are used internally by RiverWare to execute Rules and control the simulation. The following is a brief description of these three main components.

Rule Language

The Rules are logical statements formulated by the modeler and written within RiverWare, in a special language, RiverWare Policy Language (RPL), which is interpreted at runtime. Thus, the Rules are data which can be saved and modified without having to recompile a program.

One of the major features of Rulebased simulation is the use of a structured editor that allows the user to quickly develop customized Rules and Functions to emulate the desired behavior in the model. Each Rule has a unique priority and can set one or many slots in the model workspace. Rules are basically formatted as "if-then-else" statements where the "if" often refers to the current state of the system (values of slots), and the "then else" sets values (slots) on the system.

Rule Processor

The Rule processor is responsible for the management and execution of the Rules and, as such, is the "traffic cop" between the user-specified Rules and the RiverWare simulator. It loads, unloads, stores and updates the Rules. It retrieves data from RiverWare that the Rules need, including the values of any variables in the RiverWare workspace, and can also set any value in the RiverWare workspace based on what the user specifies in the Rules.

Execution of Rules consists of keeping a list (referred to as the agenda), ordered by priority, of all Rules that need to execute and then executing or "firing" the highest priority Rule from the agenda. The agenda is updated during Rulebased simulation as Rules are executed and as variables on the RiverWare workspace change. A Rule is added to the agenda at the start of every timestep and whenever the value

of any of its dependencies change during the timestep. A Rule is removed from the agenda after it is executed.

Rulebased Simulator Controller

The final component is the Rulebased simulation controller itself. The Rulebased simulation controller orchestrates the alternation between execution of Rules and solving of the simulation during a Rulebased simulation run. It manages the dispatching of objects on the dispatch queue, just as the simulation controller does. When no objects can dispatch, the controller invokes the Rule processor. The Rule processor fires Rules on the agenda, beginning with the highest priority, until a Rule is successful. Then, the controller again processes the dispatch queue.

There are more steps involved, but generally the sequence of the Rulebased simulation controller is as follows:

1. Dispatch objects without using Rules until the system is completely solved or the dispatch queue is empty (no objects have enough information to dispatch).
2. Execute the highest priority Rule on the agenda and return to step 1.
3. If no Rules exist on the agenda, simulation for the timestep is complete.

In Rulebased simulation, objects will solve, just as in simulation if they are exactly specified. However, at the point that no more objects can dispatch (i.e., some objects are under determined), the Rule system is consulted for additional information. If a candidate Rule applies, it will fire and one or more slots in RiverWare may be changed. If the object (or others due to links) can dispatch, they will. This connection between Rules and the simulator continues until either the Rule agenda is empty or all objects have dispatched. In the former case, some objects are still under determined and warnings will be issued and the simulation will continue. In the latter case, the Rule agenda is consulted one last time to ensure that the priorities of the Rules have been satisfied. If not, the highest priority Rule will fire and the process will attempt to continue.

It should be noted that the Rule processor is **NOT** allowed to overwrite any user-supplied inputs; i.e., user-inputs are implicitly the highest priority. This premise forces the simulator to rely on some Rules to provide the necessary information to initially dispatch certain objects. Higher priority Rules may eventually override the effects of initial Rules, but under all circumstances, user-inputs are preserved.

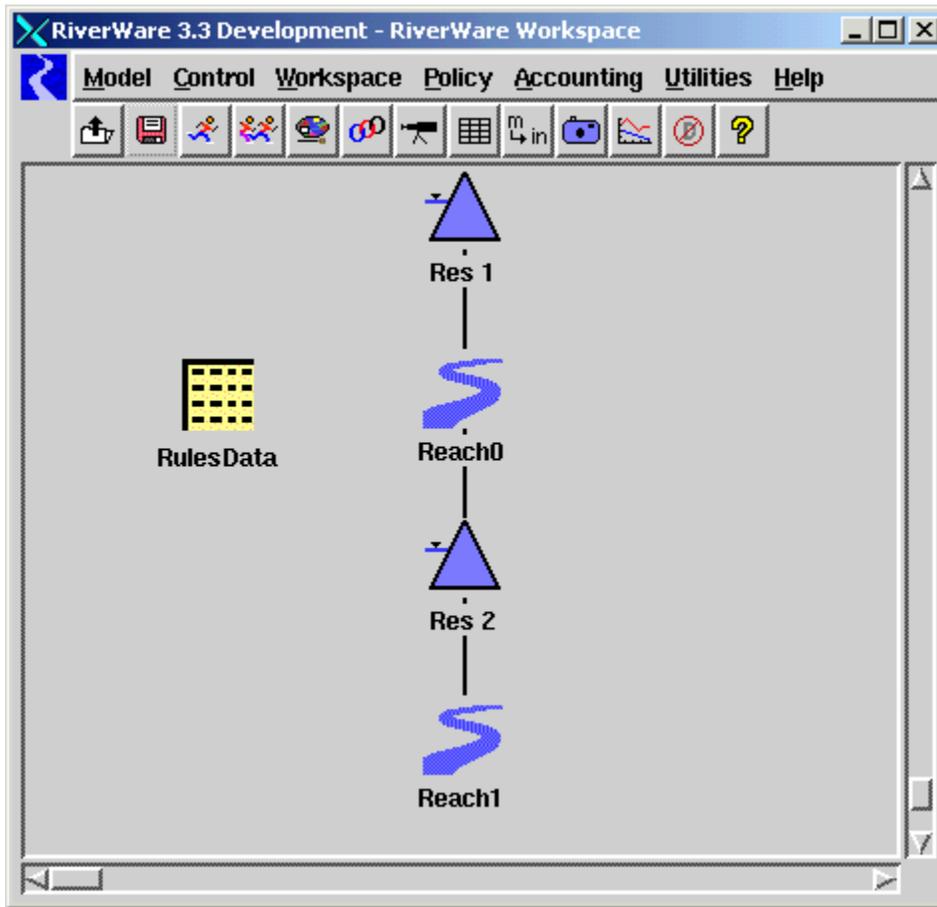
Example of Rulebased Simulation

To further illustrate how Rulebased simulation works within RiverWare, consider the model in Fig. 2. This simple model consists of two reservoirs (Res1 and Res2), two reaches (Reach0 and Reach1) and one data object that contains information used by the Rules.

There are three Rules that control how this model will solve; these three Rules along with their dependencies are listed below in order of priority:

1. Check the outflow of Res1 against a minimum flow, if the outflow is less than the minimum flow, set the outflow of Res1 to the minimum flow. The dependency for this Rule is Reach1 outflow.
2. Set a target storage for Res1. The dependency for this Rule is Res1 storage.
3. Set a target outflow for Res2. The dependency for this Rule is Res2 outflow.

Figure 2 - Rulebased Simulation Example Model



The four inputs into the model are shown below:

1. Inflow into Res1 (this is input on the inflow slot on Res1)
2. Minimum flow for Res1 (this is input in the RulesData data object)
3. Target storage for Res1 (this is input in the RulesData data object)
4. Target outflow for Res2 (this is input in the RulesData data object)

Given the model in **Fig. 2** the three operation Rules and the inputs into the model, the Rulebased simulation would precede as shown in **Table 3**. **Table 3** shows the steps that would take place during one timestep when solving this model. First the action is listed, then the Rules that are on the agenda (candidates for firing), any slots that may get set by the Rule, and finally a list of slots and their current priority.

Table 3 - Steps Of Rulebased Simulation

Action	Rules on Agenda	Values Set with Rules	Priority of Values
At the first of the timestep all Rules get added to the agenda, since only the inflow into Res1 is known, no object has enough information to dispatch, so the Rule controller will execute the highest priority Rule on the agenda which is Rule 1.	Rule 1 Rule 2 Rule 3	nothing set	inflow Res1 = 0 outflow Res1 = storage Res1 = inflow Reach0 = outflow Reach0 = inflow Res2 = outflow Res2 = storage Res2 =
Execute Rule 1, which is to see if the outflow of Res1 is greater than the minimum release. Because, the outflow has not been set the Rule will fail at this time.	Rule 1 Rule 2 Rule 3	nothing set	inflow Res1 = 0 outflow Res1 = storage Res1 = inflow Reach0 = outflow Reach0 = inflow Res2 = outflow Res2 = storage Res2 =
Execute highest priority Rule on the agenda (Rule 2) Rule 2 sets the storage of Res1 and now Res1 has enough information to dispatch. Res1 will now solve for its outflow. Note Res1 outflow and storage are now set with a priority of 2, corresponding to the priority of the Rule which set them. Also note that the storage of Res2 is priority 2R, indicating it was set by a Rule. Rule 2 is now removed from the agenda, but is added on again because its dependency Res1 storage has changed.	Rule 2 Rule 3	storage Res1 outflow Res1	inflow Res1 = 0 outflow Res1 = 2 storage Res1 = 2R inflow Reach0 = outflow Reach0 = inflow Res2 = outflow Res2 = storage Res2 =
The outflow of Res1 is now propagated to the inflow of Reach0. Reach0 now has enough information to dispatch and solves for its outflow. The outflow of Reach0 is then propagated to the inflow of Res2. Note that the priority of the Rule that set the outflow of Res1 is propagated (priority 2) along with the value. Also note that the dependency for Rule 1 (Reach0 outflow) is changed and so Rule 1 gets added back on the agenda.	Rule 2 Rule 3	nothing set with Rules, all values are set with propagation	inflow Res1 = 0 outflow Res1 = 2 storage Res1 = 2R inflow Reach0 = 2 outflow Reach0 = 2 inflow Res2 = 2 outflow Res2 = storage Res2 =
Res2 only knows its inflow and so it still does not have enough information to dispatch. Therefore the highest priority Rule on the agenda is fired (Rule 1). Lets assume that the outflow set with Rule 2 was less than the minimum release and so Rule 1 (since it is a higher priority than Rule 2) sets the outflow of Res1 to the minimum release with a priority of 1. Now Res1 is overdetermined. However this is where the unique priority of the Rules come into play. Notice that the inflow of Res1 has a priority of 0, the storage has a	Rule 1 Rule 2 Rule 3	outflow Res1	inflow Res1 = 0 outflow Res1 = 1R storage Res1 = 2R inflow Reach0 = 2 outflow Reach0 = 2 inflow Res2 = 2 outflow Res2 = storage Res2 =

<p>priority of 2R and the outflow a priority of 1R, therefore Res1 will solve for its storage since it is the lowest priority. Rule 1 is now removed from the agenda.</p>			
<p>The outflow of Res1 is now propagated to the inflow of Reach0. It overwrites the previous value of inflow to Reach0 because it has a higher priority. Reach0 now resolves for its outflow. The outflow of Reach0 is then propagated to the inflow of Res2. Note that the priority of the Rule that set the outflow of Res1 is once again propagated (priority 1) along with the value. Also note that the dependency for Rule 1 (Reach0 outflow) is changed and so Rule 1 gets added back on the agenda.</p>	<p>Rule 2 Rule 3</p>	<p>nothing set with Rules, all values are set with propagation</p>	<p>inflow Res1 = 0 outflow Res1 = 1R storage Res1 = 1 inflow Reach0 = 1 outflow Reach0 = 1 inflow Res2 = 1 outflow Res2 = storage Res2 =</p>
<p>Res2 still only knows its inflow and so it still does not have enough information to dispatch. Therefore the highest priority Rule on the agenda is fired (Rule 1). This time the outflow of Res1 is greater than or equal to the minimum release so it fails. Rule 2 then fires, and set the storage of Res1, however note the priorities, inflow is 0 and can't be changed and the outflow is priority 1 that was set with a Rule therefore Res1 will redispach solving for storage with priority 1. This gives the same results as before and so Rule 2 will fail. Finally Rule 3 fires and sets the outflow of Res2 with a priority of 3. Note that the dependency for Rule 3 is the outflow of Res2 therefore Rule 3 is added back on the agenda.</p>	<p>Rule 1 Rule 2 Rule 3</p>	<p>storage Res2</p>	<p>inflow Res1 = 0 outflow Res1 = 1R storage Res1 = 1 inflow Reach0 = 1 outflow Reach0 = 1 inflow Res2 = 1 outflow Res2 = 3 storage Res2 =</p>
<p>Now Res2 has enough information to dispatch (it knows its inflow and outflow) and so it will solve for its storage and set it with a priority of 3. Rule 3 will fire one more time since it is on the agenda, however, it will fail because the value of the outflow of Res2 is already set to the same value. Rule 3 is now removed from the agenda.</p>	<p>Rule 3</p>	<p>nothing set with Rules</p>	<p>inflow Res1 = 0 outflow Res1 = 1R storage Res1 = 1 inflow Reach0 = 1 outflow Reach0 = 1 inflow Res2 = 1 outflow Res2 = 3R storage Res2 = 3</p>
<p>All objects have now dispatched and there are no Rules left on the agenda, so this will conclude the timestep. The controller would then execute the next timestep.</p>	<p>no Rules left on agenda</p>	<p>end of timestep</p>	<p>inflow Res1 = 0 outflow Res1 = 1R storage Res1 = 1 inflow Reach0 = 1 outflow Reach0 = 1 inflow Res2 = 1 outflow Res2 = 3 storage Res2 = 3</p>