Supporting Alternate Execution Times for Object Level Accounting Methods

Requirements and Design

Authors: Neil Wilson, Edie Zagona, Phil Weinstein, Nancy Hall

1.0 Purpose and Approach

RiverWare is used for water rights allocation, a scenario in which the accounting solves first, and the physical system values are primarily based on the results of the accounting solution. The existing controllers were designed to have the physical system solve before the accounting. This order is "managed" by the Object Level Accounting Methods (OLAMs), specifically the SlotInflow Category which sets the values of the hydrologic inflows into the accounting system. The methods execute after the simulation or rulebased simulation has completed at each timestep. Since the Slot Inflow values are required for the accounts to solve, the accounts typically solve after the physical system has solved.

In some models, for example, models that use the water rights solver, the accounting must solve before the physical system. The accounts cannot solve until the Slot Inflow values are set in the accounts. The Slot Inflow OLAMs don't execute until after the simulation at each timestep. So, for these models, rules are being used to move hydrologic inflows from the physical objects to the account Slot Inflows. This is awkward (rules are meant for policy, not for modelling physical processes), and it imposes a performance penalty (methods are more efficient than rules).

One approach would be to have a different accounting controller that would execute the OLAMs before the physical simulation controller. However, it would be a more flexible system if some methods could execute before the simulation and others after. Hence the idea to allow the user to decide when the OLAMs will execute.

This document proposes the design of enhancements that meet the needs of water rights models and similar accounting-driven models by changing the behavior of the OLAMs and also the Accounting Controller to allow OLAMs to execute at different times relative to the simulation at each timestep. The execution times for an OLAM will be selectable by the user. In addition, two new OLAMs for use in accounting-driven models are proposed that will take advantage of new execution times.

2.0 Execution TImes

All timestep-based run controllers (as opposed to, for example, optimization controllers) execute the following phases in a run: pre-initialization, initialization, begin run, begin block (timestep), execute block (timestep), end block (timestep), end run. Each phase can be considered a state of the controller. Contrary to what one might expect, when controllers are combined and driven with a meta-controller, the BeginBlock, ExecuteBlock, and EndBlock phases are not interleaved. The MetaControl::BeginBlock and EndBlock methods do nothing; while its ExecuteBlock phase calls each controllers' Begin, Execute, and End block methods in series before moving to the next controller. This means that introducing new execution times will require some changes in to controller methods as is described in the later implementation section.

The following possible execution times for OLAMs were considered:

1. Beginning of Run: execute once for each timestep of the run, but do so at beginning of run. No dependencies on any slots will be respected. Controller will loop through the timesteps, executing the selected method for each timestep. Setting of accounting slots may cause accounts to solve at this time. If these methods set simulation object slot values, it will have no effect (on dispatching) as long as it is the accounting controller that is executing these methods, since it is the controller's slotChanged() method that determines what happens as a result of setting the slot's value.
2. End of Run:execute once for each timestep of the run, but do so at end of run. This is probably not useful and we propose to omit this.
3. Beginning of Timestep, current timestep only, run before simulation or rules, no dependencies on accounting slots or simulation slots, hence, no re-execution. This would be useful for initializing accounting or simulation slot values based on prior timestep's simulation and/or accounting results.
4. Beginning of Timestep and as Dependencies Change, current timestep only, run before simulation or rules, and after simulation during the accounting timestep if dependent accounting input slot values have changed
5. End of Timestep, current timestep only, no dependencies on accounting or simulation slots, hence, no re-execution. This would be the same as the above case (#3), but the methods would have to be written to look at current-timestep values and set next-timestep values, and would have to deal with end-of-run cases; the result is that the programming would be more awkward, so we suggest that this would not be useful and propose to omit this.
6. Beginning of Accounting Timestep and as Dependencies Change. This is the behavior of OLAMs now. Run after simulation at the beginning of the accounting-phase timestep, and during the accounting timestep if dependent accounting input slots have changed.
7. Never: This can be used to avoid ever executing the "None" or "No Method" methods.

If the user needs to have a given method execute on object A at beginning of run, and on object B at current timestep with possible re-execution, we cannot make the execution time a fixed characteristic of the method itself. An example might be a method that copies values from a simulation slot to an account's Slot Inflow slot. For some models, this would have to be run on a per-timestep basis, as the source values might be outputs. On other models, these values might be inputs, known a priori, and copied at the beginning of the run to accommodate lagged water rights. For this reason, the execution time for a method should be selectable by the user on a per-object basis.

2.1 Shortcomings of OLAMs

OLAMs are currently not powerful enough to satisfy accounting-driven models for two reasons.

Dependencies:

The models we are addressing use the results of accounting to drive the physical simulation. For these models, accounting solves based on rules (e.g., results of the RPL function SolveWaterRights()), then rules use the accounting results to set releases from reservoirs, then simulation solves. If OLAMs (as presently handled) are used to compute, say, account Slot Inflow values based on simulation object data, the accounts will not solve until after the simulation and rules run, which is not soon enough for the rules that use accounting to drive the simulation. For these models, we need to run some OLAMs before the rules run for a given timestep.

We cannot have these OLAMs depend on simulation results and be re-run during the rules execution, nor re-executed at the end of the timestep by the accounting controller. These are OLAMs that do not depend on current-timestep simulation results. They might depend on prior-timestep simulation results, however.

Timestep lags in the accounting model:

Some of these models might have lags in the accounting network; the water rights solution requires that accounts have sufficient information to solve at timesteps later than the current controller timestep. If OLAMs are executed only for the current controller timestep, then populating account slots for future timesteps with OLAMs becomes awkward at best. For such models where the entire run's flows are known a priori, the relevant account slots (primarily starting storages and slot inflows) can be populated at the beginning of the run. We propose to create a new class of OLAMs that are executed once only, at the beginning of the run. These OLAMs will not depend on simulation or accounting slots. Methods that fit into this class of OLAMs might zero all account slot inflows, or zero all account slot inflows except those that have a certain water type or account name, or copy an input series from a simulation object slot into an account slot inflow (this latter makes modeling sense only if the entire series is input via user or DMIs.

3.0 New OLAMs

The following proposed new compiled OLAMs will be written for RiverWare to help meet the needs of accounting-driven models as discussed above. These methods will be in the Slot Inflow method category (actual category name varies between object type i.e. Control Point Pass Through Slot Inflow, Storage Account Slot Inflow, etc.) The new methods will have the indicated execution times by default, although execution times will be user-selectable and can be changed by the user if needed.

TABLE 1. Proposed new compiled OLAMs

Name	Input	Execution Time	Effect
Zero Slot Inflows	None	Beginning of Run	Make all account slot inflows zero for all timesteps.
Copy Slot to Slot Inflow	Target Account	Beginning of Timestep (before simulation)	Call Zero Slot Inflows, then Copy values from the object's "local inflow" slot (slot name varies with object type) to Slot Inflow on the target account. Provide capability in Multi-Object Method Selector to set the target account on multiple objects at once by specifying an account name or a water type.

4.0 Requirements Summary

1. Allow OLAMs to have an execution time attribute on a per object basis.
2. Modify the accounting controller to execute OLAMs at the following times:
   - Beginning of Run
   - Beginning of Timestep
   - Beginning of TImestep and as Dependencies Change
   - Beginning of Accounting Timestep and as Dependencies Change
3. Allow the user to select the execution time of each instance of an OLAM on a per object basis.
4. Create new OLAMs for "Zero Slot Inflows" and "Copy Slot to Slot Inflow"

5. Extend the Multi-Object Method Selector to allow selection of OLAM execution time across multiple objects and to allow specification of the input account by either account name or by water type for the “Copy Slot to Slot Inflow” method across multiple objects.

5.0 Implementation in RiverWare

The following sections discuss implementation for the above requirements.

5.1 OLAM Execution TIme

An enumerated type for execution time will be added to the Method class and will include values for the four execution times (Beginning of Run, Beginning of Timestep Once, Beginning of Timestep (and as dependencies change), and After Simulation (beginning of accounting timestep and as dependencies change). Public methods to get and set the execution time on the Method class will be added. In addition, a new ObjectMethod constructor will be created to allow specification of the desired execution time when the ObjectMethod is created.

TABLE 2. OLAM Method Execution Times

Symbol	Displayed Text	Tooltip Text
OLAM_NEVER	Never	Method does not execute.
OLAM_BEG_BLOCK_AND_EXEC	After Simulation	Execute the method after each timestep's simulation is complete and as dependencies change.
OLAM_BEG_RUN	Beg. of Run	Execute the method once per timestep at the beginning of the run.
OLAM_PRE_BLOCK	Beg. of Timestep Once	Execute the method once before each timestep's simulation.
OLAM_PRE_BLOCK_AND_EXEC	Beg. of Timestep	Execute the method before each timestep's simulation and as dependencies change.

Each simulation object will need to save and load this attribute for each of its OLAMs. Currently there are no attributes saved per method with an object. The MethodTable class will be modified so that when it dumps the selected methods for an object to the model file, it will also write a string representing its execution time. During model load, the SimObj class will set the methods execution time when it sets the selected method from the model file.

5.2 Controller Modifications

We noted earlier that the BeginBlock, ExecuteBlock, and EndBlock phases are not interleaved by the meta controller. This means that if we want to execute a new class of OLAMS before the rules or simulation controller is called, some restructuring of the controllers' methods is needed.

We will add another virtual method (preBlock) to every controller, for calling before the beginning of each timestep. The RunInfo::executeBlock will have to invoke this new method. The RunInfo's executeBlock method will now look like this:

```
void RunInfo::executeBlock()
{
    _runTimer.start();
    if ( moreBlocks && shouldContinue() )
    {
        RI_CALLBACK( RI_ADVANCE_BLOCK );
        setState( RUNNING );
        controller->preBlock(); // NEW LINE
        controller->beginningOfBlock();
        controller->executeBlock();
        controller->endOfBlock();
        moreBlocks = controller->advanceBlock();
    }
    _runTimer.stop();
}
```

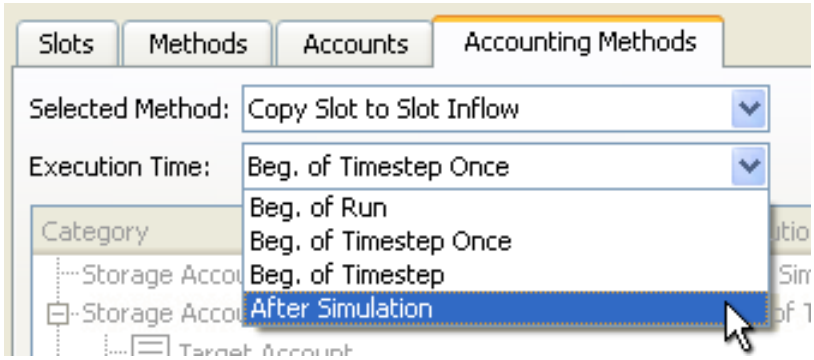
The meta controller's preBlock() method will invoke each of its sub-controllers' preBlock() methods, and the accounting controller's preBlock() method will execute the OLAMs whose execution time is Beginning of Timestep Once or Beginning of Timestep.

The accounting controller's beginningOfBlock() method will be modified to execute only those OLAMs whose execution time is Beginning of Accounting Timestep.

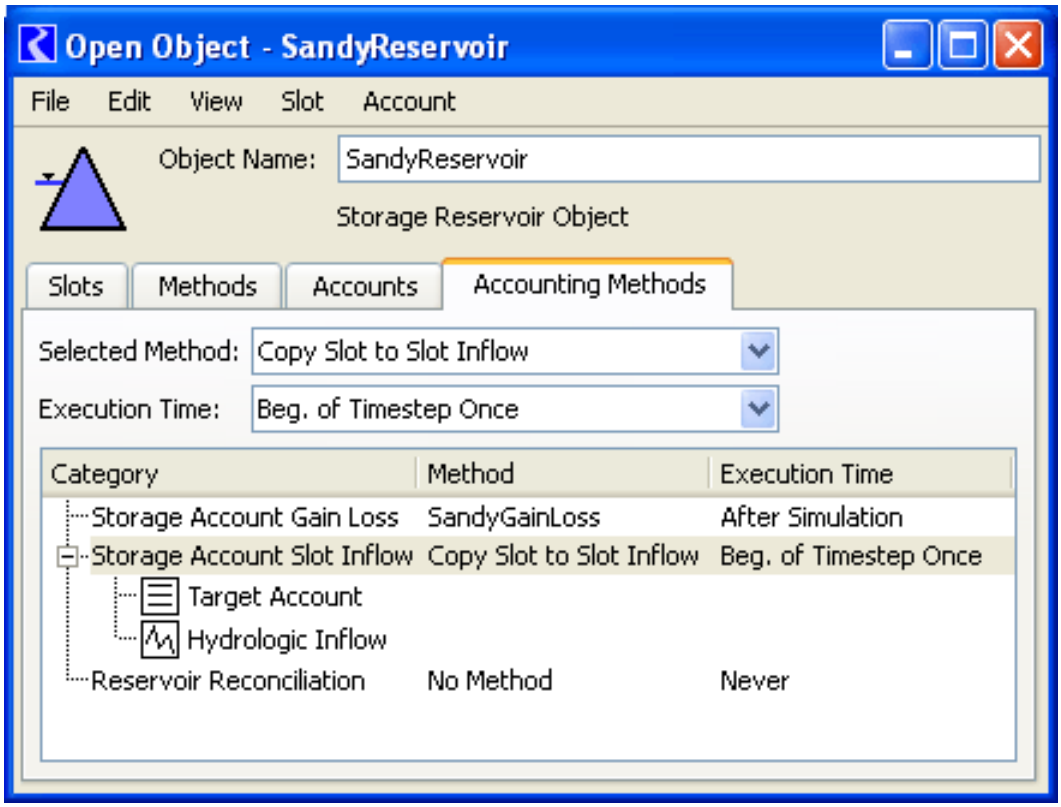
The account controller's processQueue method will be modified to only execute entries whose methods have execution times of Beginning of Timestep, and After Simulation. This will allow these methods to re-execute if dependent accounting input slots change.

5.3 User Selection of OLAM Execution TImes

Execution times for OLAMs will be selected on a per object basis, so this will need to occur in the open object dialog. A new tab will be added to this dialog called Accounting Methods that will only be enabled in accounting models.



OLAMs, which are now shown along with simulation methods in the existing methods tab, will be moved to the new tab. The new tab will appear like the Methods tab, including the **Selected Method:** combo box at the top of the tab. A new adjacent combo box labeled **Execution Time:** will be added where the user can choose the execution time for the selected method. The signal from the combo box will be connected to a function that will set the execution time on the selected method when the combo box is changed. The category / method list on this tab will contain a new "Execution Time" column showing the currently set execution time for each selected method.



5.4 Create New OLAMs

5.4.1 Zero Slot Inflows

This new OLAM will be created on ten simulation object types:

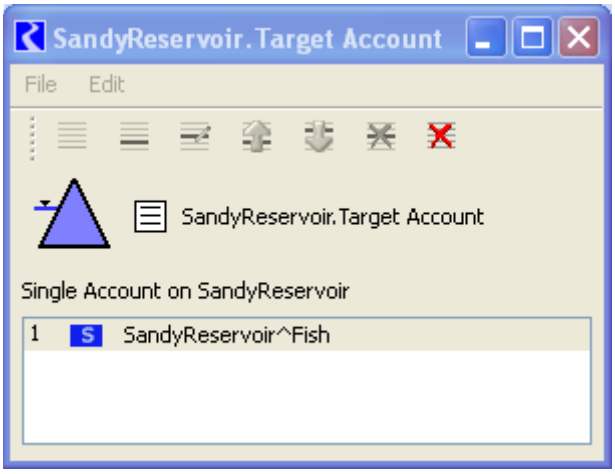
- Bifurcation
- Confluence
- Control Point
- Distribution Canal
- Inline Pump
- Pipe Junction
- Pipeline
- Reach
- Reservoir
- Stream Gage

There will be no slots associated with this method. It's code will find the Slot Inflow slot for all accounts on the object and set the current timestep value to zero. By default, this method will be given an execution time of Beginning of Run. Note that with this execution time, the controller's begin run method will iterate and call Zero Slot Inflows for all timestep's in the run at the beginning of the run.

5.4.2 Copy Slot to Slot Inflow

This new OLAM will be created on three simulation object types:

- Control Point
- Reach
- Reservoir



"Copy Slot to Slot Inflow" will require the following new slot when the method is selected:

- Target Account- a slot whose type is ListSlot<Account>, which will contain one account on the object which will receive copied values.

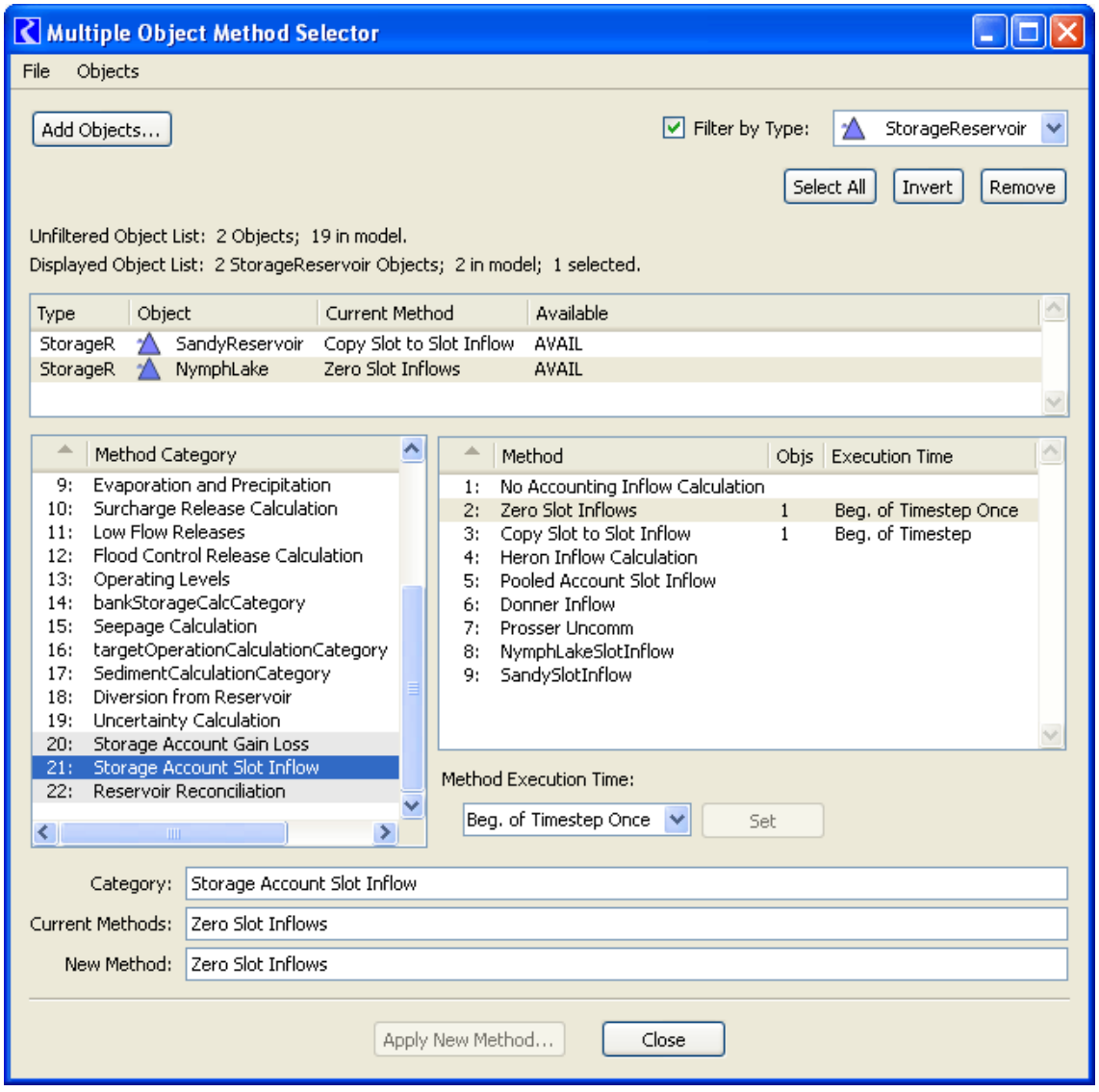
Note that ListSlot<Account> will be a new instance of the templated ListSlot class. ListSlot behaviors associated particularly with Account will be encapsulated in a templated ListSlotTraits class. The twelve methods for ListSlotTraits will need to be coded for Account.

The "Copy Slot to Slot Inflow" method will first assign zero to the current timestep value of the Slot Inflow slot of all accounts on the object. It will then find the current timestep value in the object's "local inflow" type simulation slot, and will copy that value to the current timestep of the Slot Inflow slot of the specified Target Account. By default this method will be given an execution time of Beginning of Timestep, meaning it will be executed once at each timestep before simulation starts.

5.5 Extended Multi-Object Method Selector

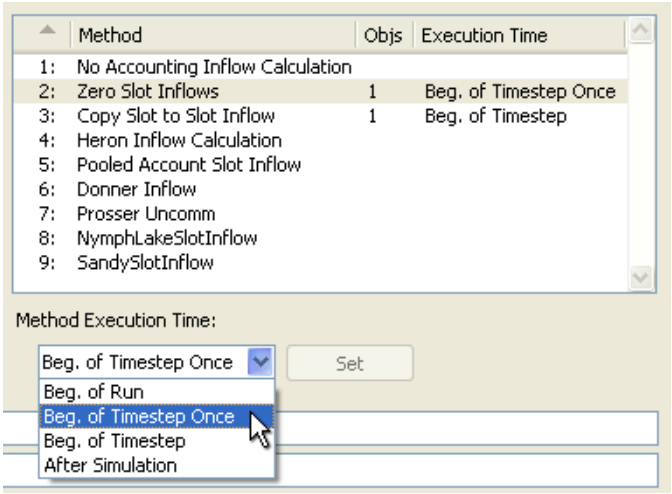
The Multi-Object Method Selector dialog has several enhancements to support the user-settable OLAM execution times, plus special support for the new "Copy Slot to Slot Inflow" OLAM. The image below shows two of the general OLAM enhancements:

- OLAM category items are shown with a grey background. *In the image below, see the bottom three rows in the method category list.*
- When an OLAM category is selected, additional controls are shown below the method list. *In the image below, see the "Method Execution Time" combo box and "Set" button.* These controls are described on the following pages.



5.5.1 Setting the Execution Time of OLAMs.

When an object level accounting category is selected in the method category list *(see above)*, that category's method options are shown in the methods list *(shown here)*.



Methods which are set on at least one of the simulation object being edited in this dialog are indicated with an object count (1 or greater -- in the "Objs" column), and the execution time of those particular methods (on objects having those methods selected) is indicated in the new "Execution Time" column. If not all of those object methods have the same execution time, then "Multiple Execution Times" is shown in the Execution Time column for that item.

When clicking on a method item selected on at least one simulation object, the Method Execution Time combo box is set to the indicated execution time -- or, if multiple execution times are indicated for that method item, one of those several execution times are used. The user can selected a different execution time in the combo box for the purpose of changing the execution times of the selected methods.

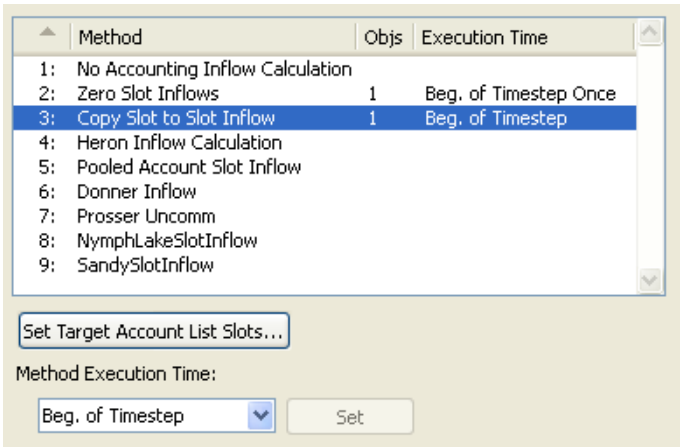
In this dialog, there are two ways of setting the execution time of OLAM methods:

1. By clicking the "Set" button next to the Method Execution Time combo box. This effects the methods on the objects currently having that method setting.
2. By clicking the "Apply New Method" button at the bottom of this dialog. *(See previous page)*. This simultaneously sets the selected method item on the selected objects AND sets the execution times of those methods.

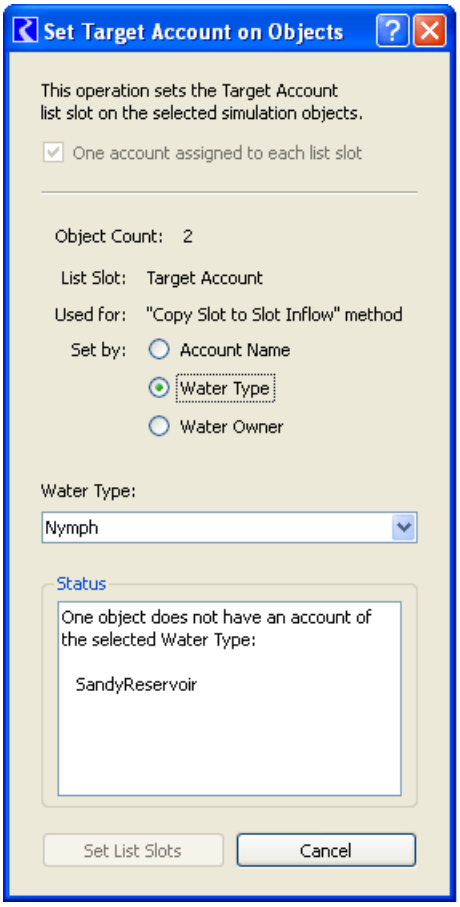
5.5.2 Special support for "Copy Slot to Slot Inflow": Set Target Account List Slot.

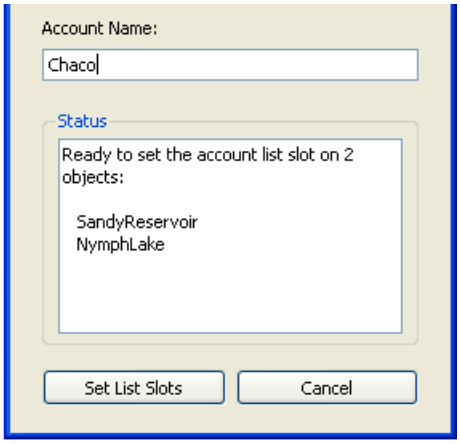
The Multi-Object Method Selector also has special support for the new "Copy Slot to Slot Info" OLAM.

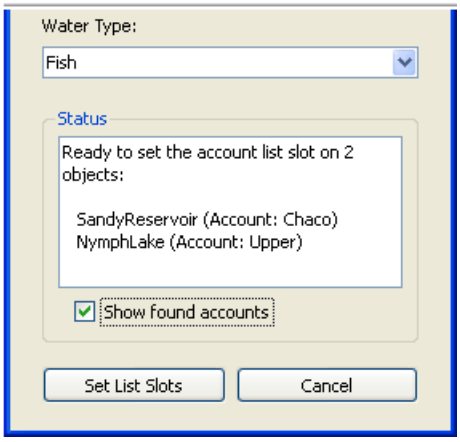
When that method item is selected, the "Set Target Account List Slots..." button is shown below the methods list. This button is enabled if any objects currently have the "Copy Slot to Slot Info" method selected. Clicking this button brings up the new "Set Target Account on Objects" dialog box for the purpose of setting the value of the selected objects' Target Account List Slot to an Account chosen by one of several alternative criteria.



The "Set Target Account on Objects" dialog operates on each simulation object being editing in the Multi-Object Method Selector which has the "Copy Slot to Slot Inflow" method selected.







The user selects one of three criteria for choosing an Account on each simulation object to be assigned to that object's Target Account List Slot:

- Account Name
- Water Type
- Water Owner

There must be exactly one Account on each object matching the criteria. The assessment of that condition is dynamically displayed in a Status panel.

In the case of the "Set List Slots" operation being "ready" (i.e. exactly one Account matching the entered criteria for each simulation object), the Status panel enumerates the simulation objects; *see the two right-side images.* Otherwise, the Status panel enumerates the simulation objects which either have no Accounts matching the criteria, or have more than one Account matching the criteria. Additionally, when the Water Type or Water Owner criteria are selected, the names of the matching Accounts can optionally be shown.

Clicking the "Set List Slots" button (when enabled) assigns the objects' Target Account List Slots and dismisses the dialog.

5.6 Phased Implementation

5.6.1 Phase 1

- Add an enumerated type for execution time to the Method class.
- Add public methods to get and set the execution time on the Method class.
- Add new ObjectMethod constructor that allows specification of execution time.
- Code all of the controller modifications discussed above.
- Create the "Zero Slot Inflows" OLAM and update its documentation.

At the end of this phase we will be able to execute precompiled OLAMs at different times, based on their compile-time-determined execution times, and will have the "Zero Slot Inflow" OLAM as an example one executing at a different time (Beginning of Run) than was formerly allowed.

5.6.2 Phase 2

- Coding for each object to save and load the execution time attributes of its OLAMs with the model file.
- Code the open object dialog changes to allow user selection of OLAM execution times.
- Update the user documentation for the open object dialog and for OLAMs.

At the end of this phase the user will be able to specify the desired execution time for any OLAM (including User Defined Accounting Methods) on any object.

5.6.3 Phase 3

- Code the "Copy Slot to Slot Inflow" OLAM including development of the new type of ListSlot for Account.
- Write user documentation for the "Copy Slot to Slot Inflow" method.
- Extend the Multi-Object Method Selector to allow selection of OLAM execution time across multiple objects and to allow specification of the input account by either account name or by water type for the "Copy Slot to Slot Inflow" method across multiple objects.

At the end of this phase, all work will be complete including the new OLAMS for accounting-driven models.

6.0 Time Estimates by Phase

Phase 1:

- Almost all of the work in Phase 1 was completed by Nancy in January of 2009 (with review by Patrick).
- The only remaining item is a new execution time that we have decided to add (Beginning of Timestep and as Dependencies Change). This will require adding an enumeration member for the new time and modifying the Account controller's processQueue() and preBlock() methods as well as the p_executeOLAMs method. (4 hours).

Total: 4 hours

Phase 2:

- Save and load OLAM execution times with model file. (5 hours)

- Code the open object dialog changes to allow user selection of OLAM execution times. (16 hours)
- Update the user documentation for the open object dialog and for OLAMs. (4 hours)

Total: 25 hours

Phase 3:

- Code new type of ListSlot for Account. (4 hours)
- Code the “Copy Slot to Slot Inflow” OLAMs for three simulation object classes. (16 hours)
- Extend Multi-Object Method Selector. (32 hours)
- Testing. (4 hours)
- Write user documentation for “Copy Slot to Slot Inflow”. (2 hours)

Total: 58 hours

--- (end) ---