OATC OpenMI Editor 2.0

Adrian Harper

OATC OpenMI Editor 2.0

Adrian Harper

Publication date January 2010

Chapter 1. Overview

1. Why it exists

The “OpenMI Association” is responsible for the OpenMI standard. However, it also commits to providing an open source user interface (UI) that is easily obtainable. It is not intended that this UI be “complex” or “advanced”, its primary goals are

- Provide a simple demonstration of OpenMI functionality
- Allow a basic means of connection and running OpenMI components interactively
- Minimize complexity of maintenance e.g. Make bug fixing as simple as possible by anyone

The association assumes, and indeed actively supports, the development of 3rd party editors that provide advanced or more specialized functionality. The development of this UI is managed by the "OpenMI Association Technical Committee" (OATC).

However, whenever a new version of the standard is released (or proposed) then this editor is the simplest way of demonstrating and providing examples of any new features.

2. What it cannot do

It can only run compositions where all models implement the pull approach

- i.e ILinkableComponent.CascadingUpdateCallsDisabled = false

3. Where to get it

- The OpenMI Association web site http://www.openmi.org will provide an link to where it can be downloaded.
- As it is open source, the installers and code can de downloaded from the Source Forge web sitehttp://sourceforge.net/projects/openmi/
- Additional up to date information can also be found on the OpenMI Association Technical Committee Wiki at http://wiki.openmi.org

Chapter 2. Tutorials

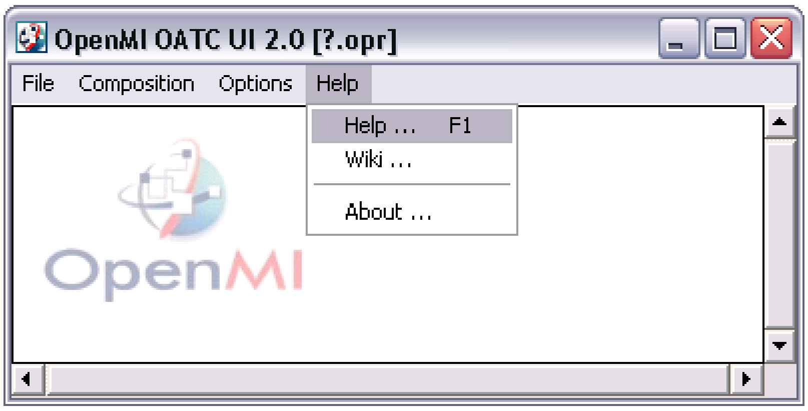
1. Quick Start

This section provides a quick example of building a simple composition and running it. It is intended for people

- Who have used and know version 1.0 and wish to quickly get acquainted with new features
- Who want to run an example as quickly as possible to check their implementation works as expected

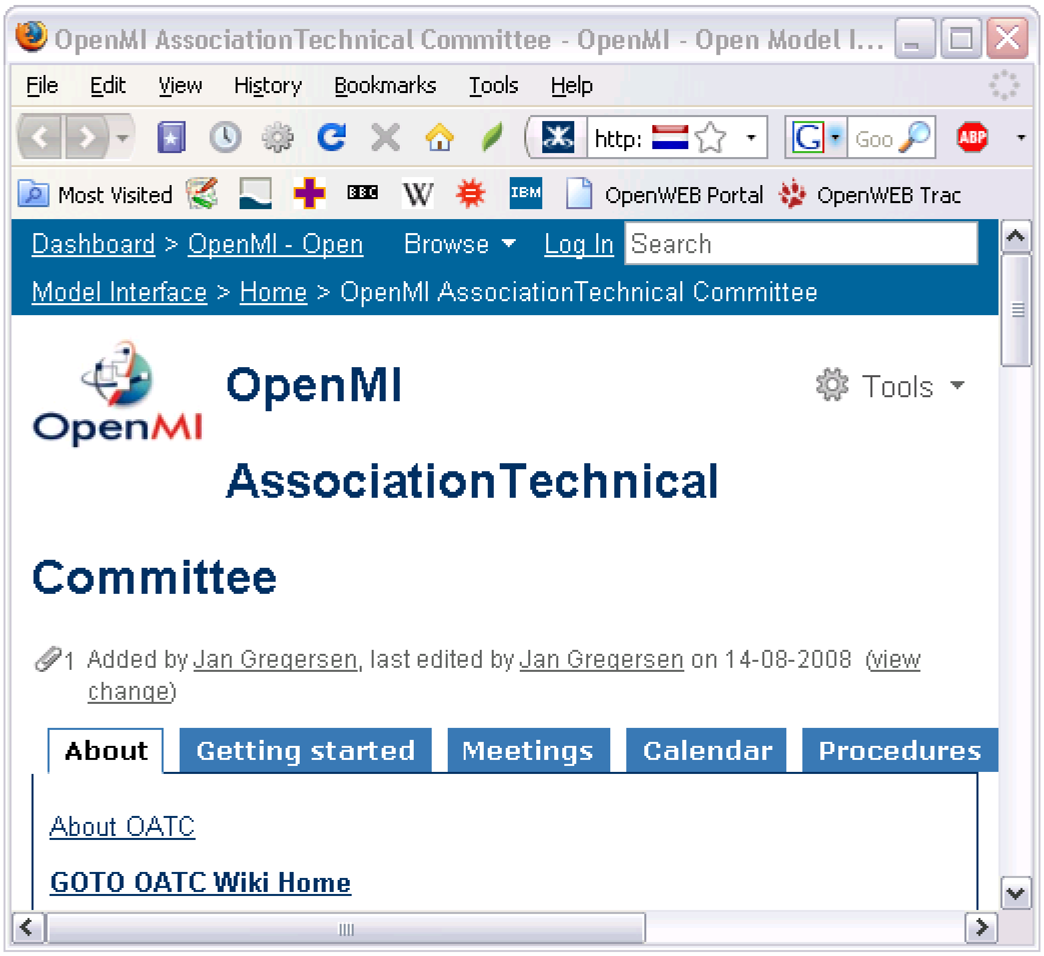
1.1. Basic Application and Help

Figure 2.1. Basic Application



The basic application should appear as above. The latest version of this document can be obtained from the help menu [F1]. The "Wiki ..." menu item will send a URI for the OpenMI OATC Wiki pages to the default Internet browser.

Figure 2.2. Wiki Pages


OpenMI AssociationTechnical Committee - OpenMI - Open Model I...
File Edit View History Bookmarks Tools Help
Most Visited
OpenWEB Portal
OpenWEB Trac
Dashboard > OpenMI - Open Model Interface > Home > OpenMI AssociationTechnical Committee
Browse
Log In
Search
Tools

OpenMI AssociationTechnical Committee

1 Added by Jan Gregersen, last edited by Jan Gregersen on 14-08-2008 (view change)

About
Getting started
Meetings
Calendar
Procedures
About OATC
GOTO OATC Wiki Home

"About ..." provides release information about this application e.g. EULA and current release version number.

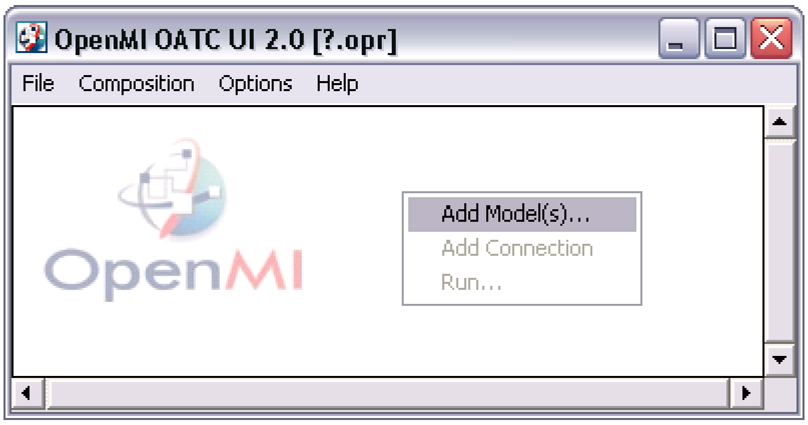
1.2. Adding Models

Either via main menu

- Composition
  - Add Model(s) ...

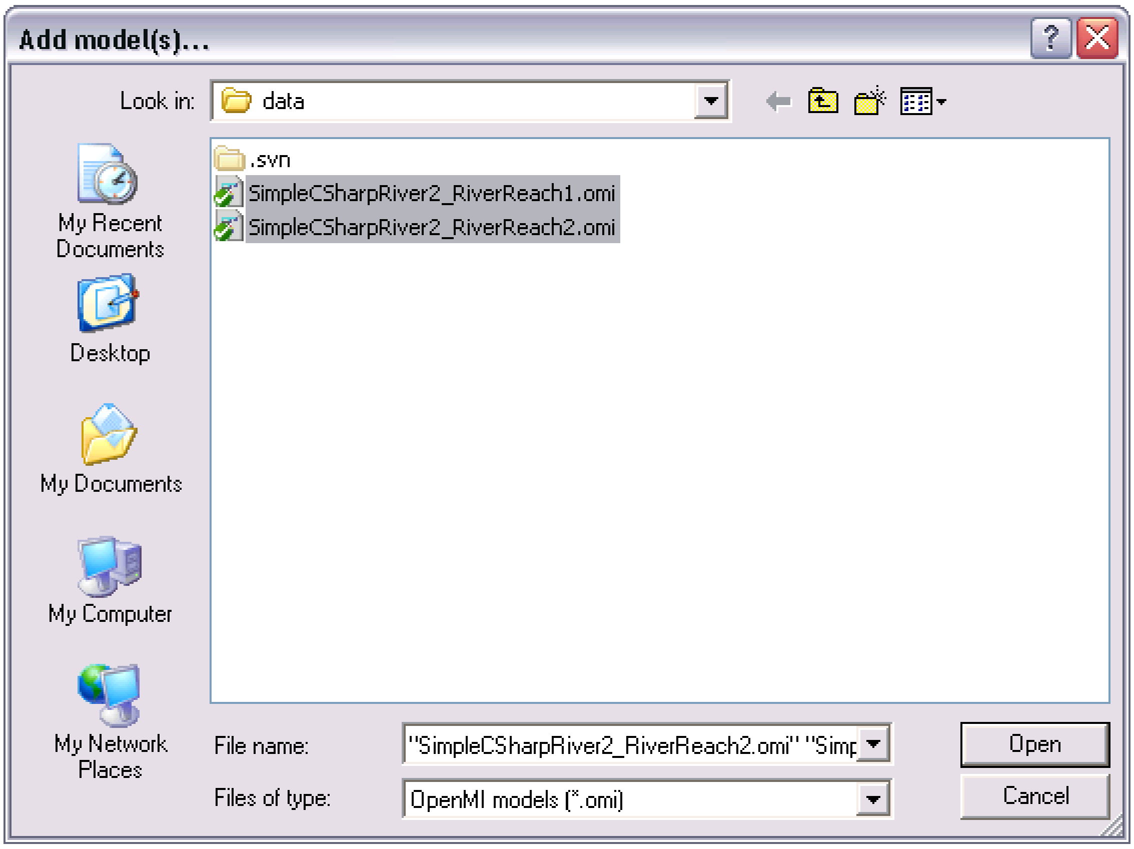
Or right click mouse within panel for context menu as follows

Figure 2.3. Context Menu



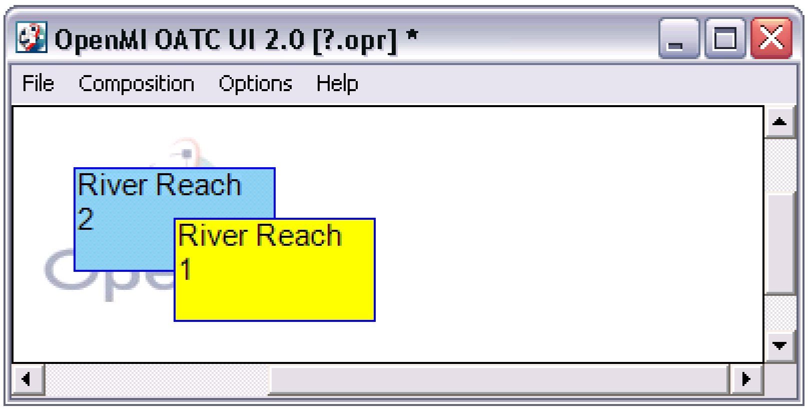
Which brings up a Open file dialog. Search for and select OpenMI models (typically files with 'omi' file extension)

Figure 2.4. Add Model(s) Dialog


Add model(s)...
Look in: data
My Recent Documents
Desktop
My Documents
My Computer
My Network Places
.svn
SimpleCSharpRiver2_RiverReach1.omi
SimpleCSharpRiver2_RiverReach2.omi
File name: "SimpleCSharpRiver2_RiverReach2.omi" "Simp
Files of type: OpenMI models (*.omi)
Open
Cancel

If the files contain valid OpenMI XML and all the underlying dll's are visible. Something like the following should appear

Figure 2.5. Loaded Models

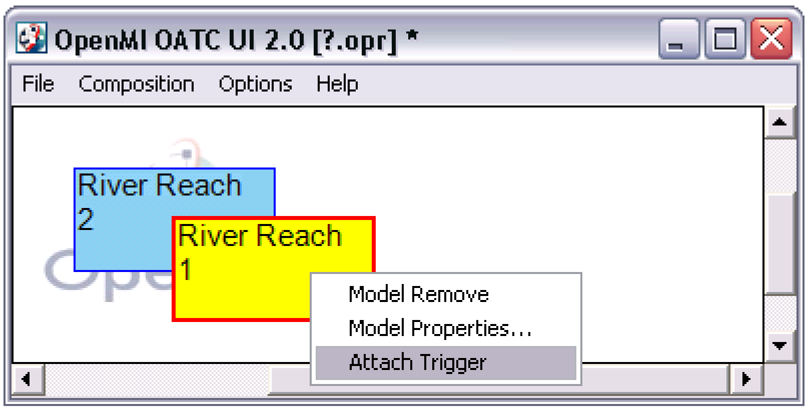


In this example, two OMI files were selected and loaded together. The last loaded model will appear blue, and all others appear yellow. Blue indicates the model within the composition that is connected to the trigger, i.e. the model that will drive the composition using the pull driven approach. Only one model within a composition can have a trigger, to change this use the main menu

- Composition
  - Attach Trigger ...

Which then allows the user to select a model to move the trigger too. Alternatively, select a model first and use the right click context menu item "Attach Trigger ..."

Figure 2.6. Attach Trigger



Note

In version 1 the trigger was a separate blue rectangle on the display. In version 2 we have combined the trigger with the model to which it is attached and turned that model blue.

Models can be moved around the screen by holding down the left mouse button over a model and dragging the mouse to a new location.

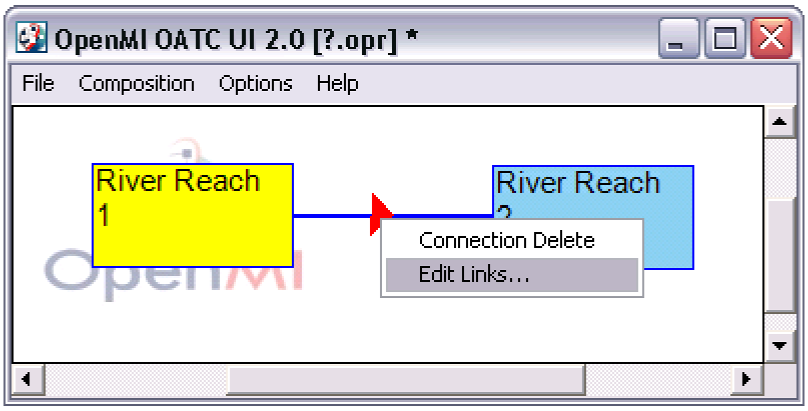
1.3. Adding a Connection

Either use main menu

- Composition
  - Add Connection ...

Which will require the user to select the source model followed by the target model. Or use the corresponding men item on the context menu for the source model. To escape from cursor 'selection mode' without completing the operation use the 'Esc' key. The connection should appear as follows.

Figure 2.7. Connection



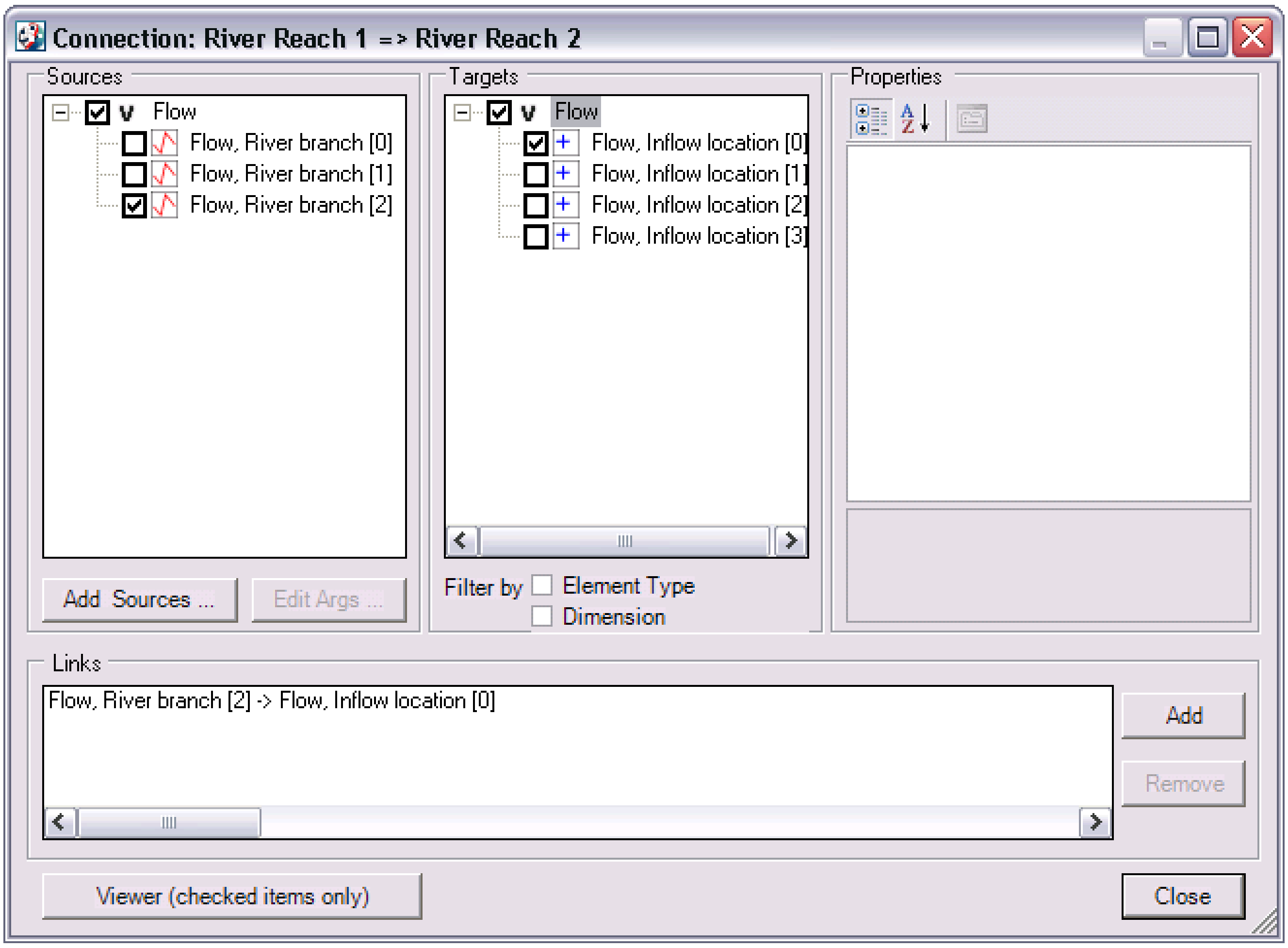
Also shown above is the right click context menu off of the connection with the "Edit Links ..." option selected.

1.4. Editing Links

The previous figure showed the context route to the edit dialog. This can also be displayed via the main menu

- Composition
  - Edit Links ...

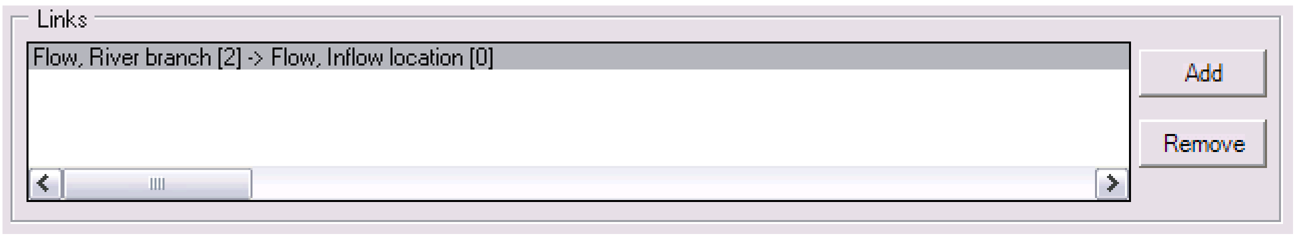
Figure 2.8. Edit Links dialog



Flow, River branch [2] -> Flow, Inflow location [0]

To create a link check both a source and target item. This should enable the "Add" button below which, when also clicked, adds a link. Currently added links are shown in the Links view pane as follows.

Figure 2.9. Links added to Connection



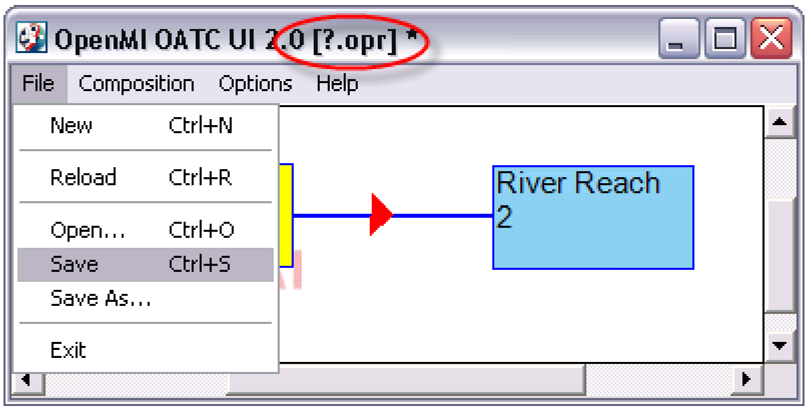
Flow, River branch [2] -> Flow, Inflow location [0]

Selecting a link enables the "Remove" button which can be used to remove the link from the connection.

1.5. Saving Compositions

If a composition is unsaved the title bar will appear as follows

Figure 2.10. Unsaved Composition



Compositions are saved to files with the '.opr' extension. The highlighted text '[?.opr]' shows the model is unsaved and if the application is closed any work will be lost. Use the File menu to save the file. Whenever a '*' appears in the title bar then the composition contains unsaved changes.

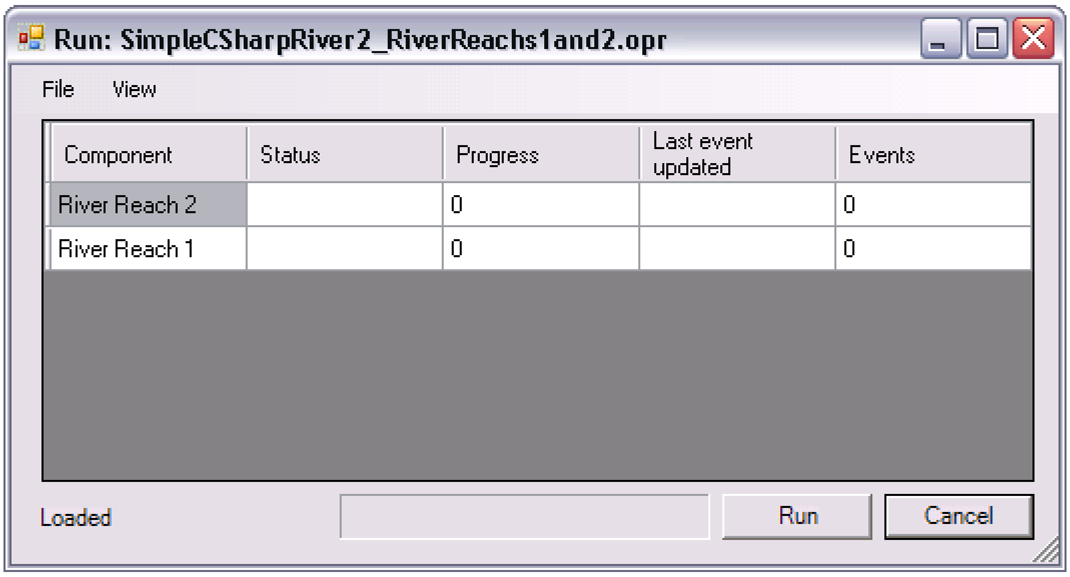
1.6. Running Compositions

Use the main menu item

- Composition
  - Run ... F5

To open the run dialog

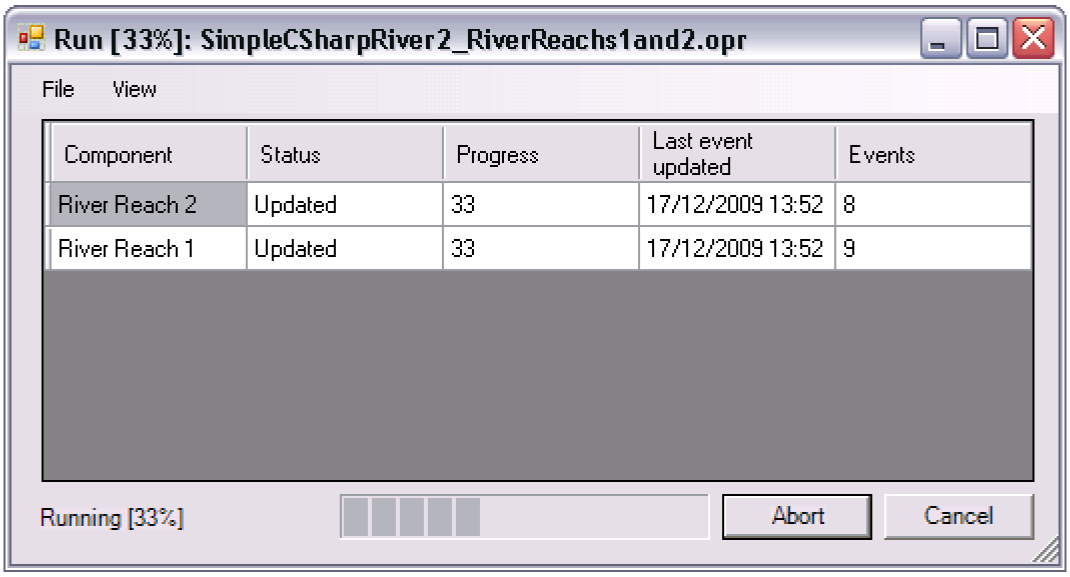
Figure 2.11. Run Dialog



Component	Status	Progress	Last event updated	Events
River Reach 2		0		0
River Reach 1		0		0

The main table has a row for each model in the composition. Pressing run button results in following.

Figure 2.12. Run in progress



Component	Status	Progress	Last event updated	Events
River Reach 2	Updated	33	17/12/2009 13:52	8
River Reach 1	Updated	33	17/12/2009 13:52	9

Running [33%]

The "Run" button changes to "Abort" to allow the user to terminate a run before it has been completed. Progress is shown by progress bar, text percentage and values within the grid. The grid columns refer too

- Component

  The name of the Model

- Status

  The last status event change by the component

- Progress

  The percentage progress of the component with respect to its own time horizon

- Last event updated

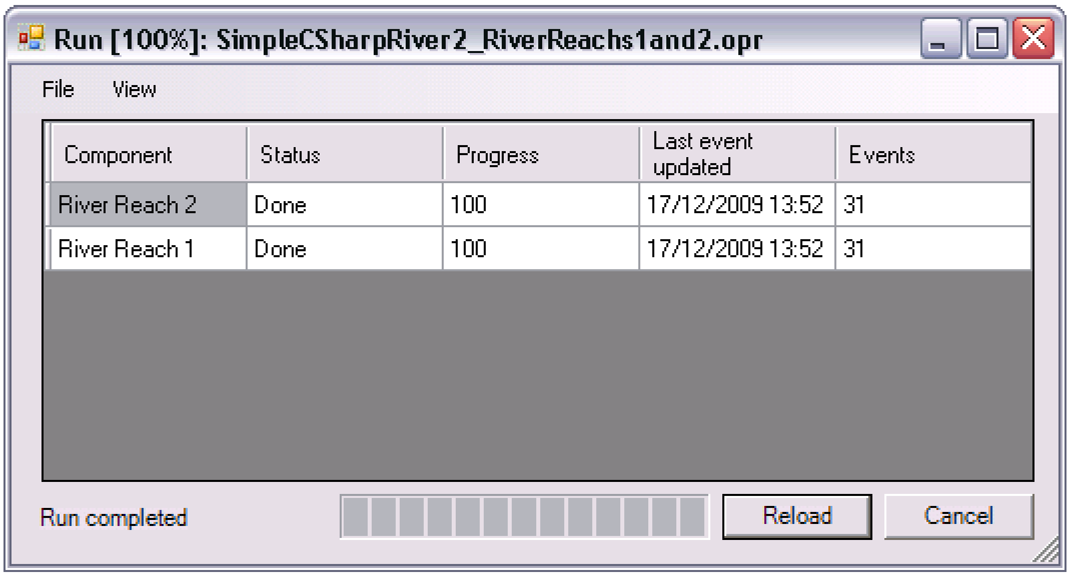
  The actual time the last event change occurred. Useful for indicating progress of large slow computations.

- Events

  The current total number of events monitored

On successful completion

Figure 2.13. Run completed



Component	Status	Progress	Last event updated	Events
River Reach 2	Done	100	17/12/2009 13:52	31
River Reach 1	Done	100	17/12/2009 13:52	31

Run completed

The "Run/Abort" button has now changed to "Reload" which allows a user to reload the composition so another run can immediately be executed.

For a successful run the model component with trigger attached must have reached 100%. However, it is not necessary for all the other models to have reached 100% as the pull driven approach only progress those models as far as necessary to satisfy the successful completion of the triggered component.

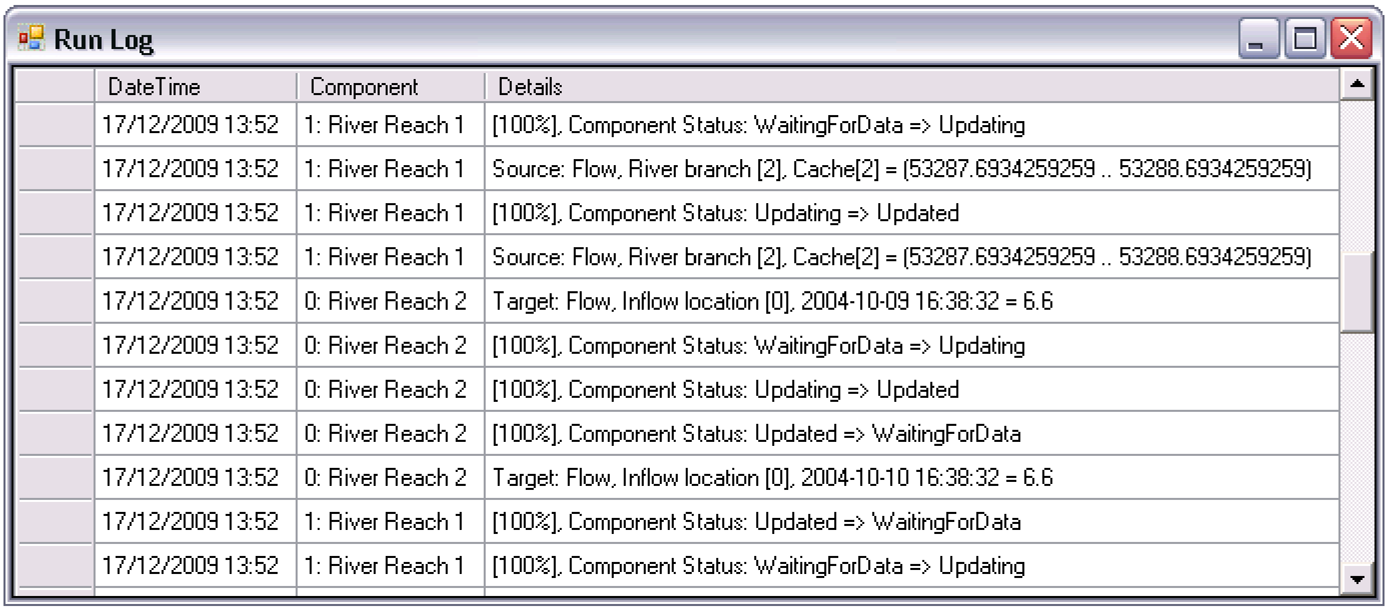
If the run fails; the text in the bottom left corner of the dialog gives information indicating this.

The status change information that appear in the table during the run are also written to a log. This log can be viewed from the main menu

- View
  - Log Ctrl+L

This opens

Figure 2.14. Run log



DateTime	Component	Details
17/12/2009 13:52	1: River Reach 1	[100%], Component Status: WaitingForData => Updating
17/12/2009 13:52	1: River Reach 1	Source: Flow, River branch [2], Cache[2] = (53287.6934259259 .. 53288.6934259259)
17/12/2009 13:52	1: River Reach 1	[100%], Component Status: Updating => Updated
17/12/2009 13:52	1: River Reach 1	Source: Flow, River branch [2], Cache[2] = (53287.6934259259 .. 53288.6934259259)
17/12/2009 13:52	0: River Reach 2	Target: Flow, Inflow location [0], 2004-10-09 16:38:32 = 6.6
17/12/2009 13:52	0: River Reach 2	[100%], Component Status: WaitingForData => Updating
17/12/2009 13:52	0: River Reach 2	[100%], Component Status: Updating => Updated
17/12/2009 13:52	0: River Reach 2	[100%], Component Status: Updated => WaitingForData
17/12/2009 13:52	0: River Reach 2	Target: Flow, Inflow location [0], 2004-10-10 16:38:32 = 6.6
17/12/2009 13:52	1: River Reach 1	[100%], Component Status: Updated => WaitingForData
17/12/2009 13:52	1: River Reach 1	[100%], Component Status: WaitingForData => Updating

The contents of this grid can be selected and pasted into the users clipboard.

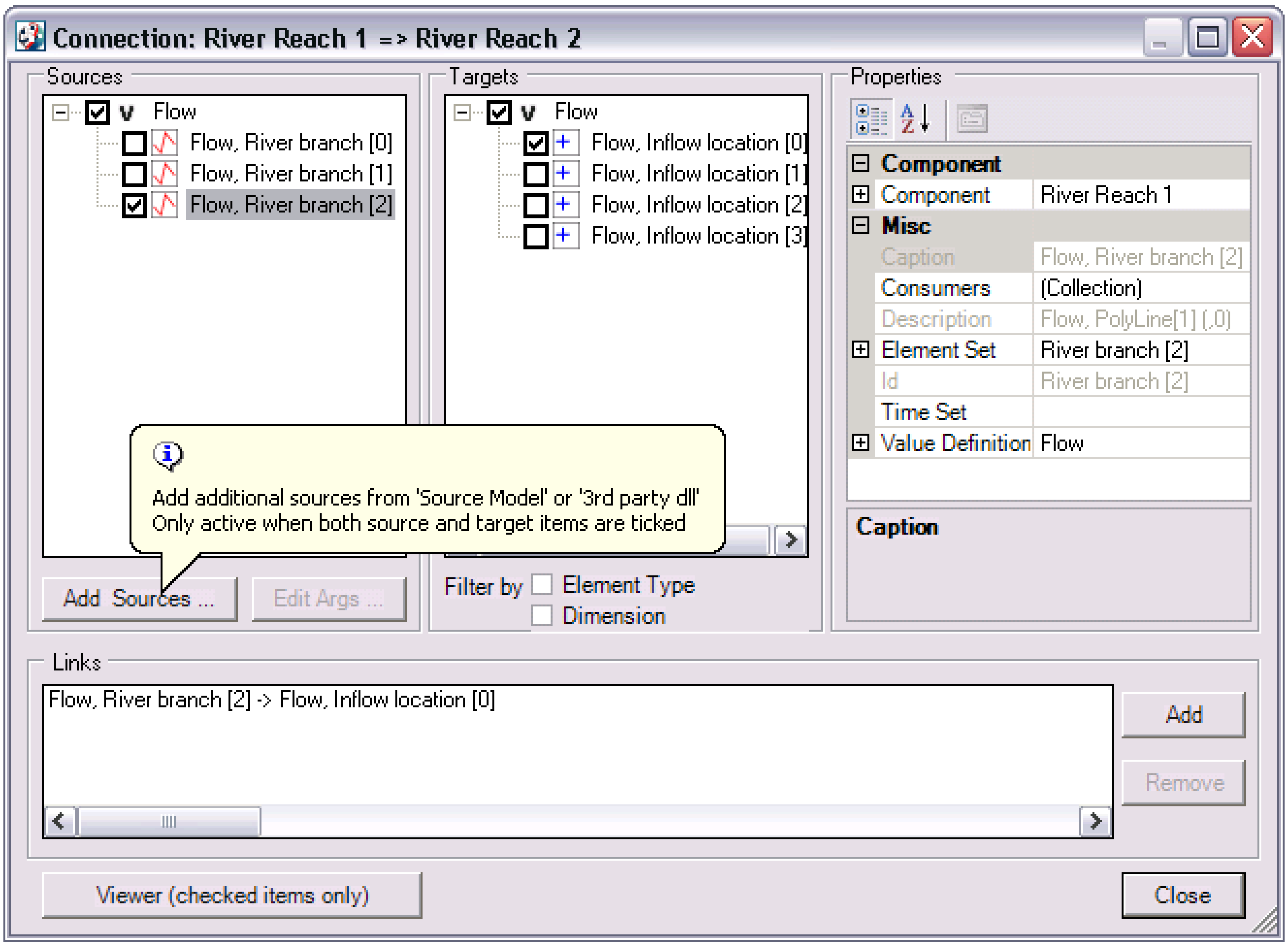
2. Adapted sources

"Adapted sources" are source exchange items that have been modified. They replace "Data Operations" which were used in version 1 of the OpenMI standard. They can be used to modify an existing source to modify either its spatial or temporal values before they are passed through the link to the target.

A very simple example of a adapted source is a Linear equation, i.e. an equation that modifies provided values according to the equation y = A*x + B where A and B are user adjustable coefficients.

Looking at the link dialog from the previous dialog

Figure 2.15. Connection Dialog



Add additional sources from 'Source Model' or '3rd party dll'
Only active when both source and target items are ticked

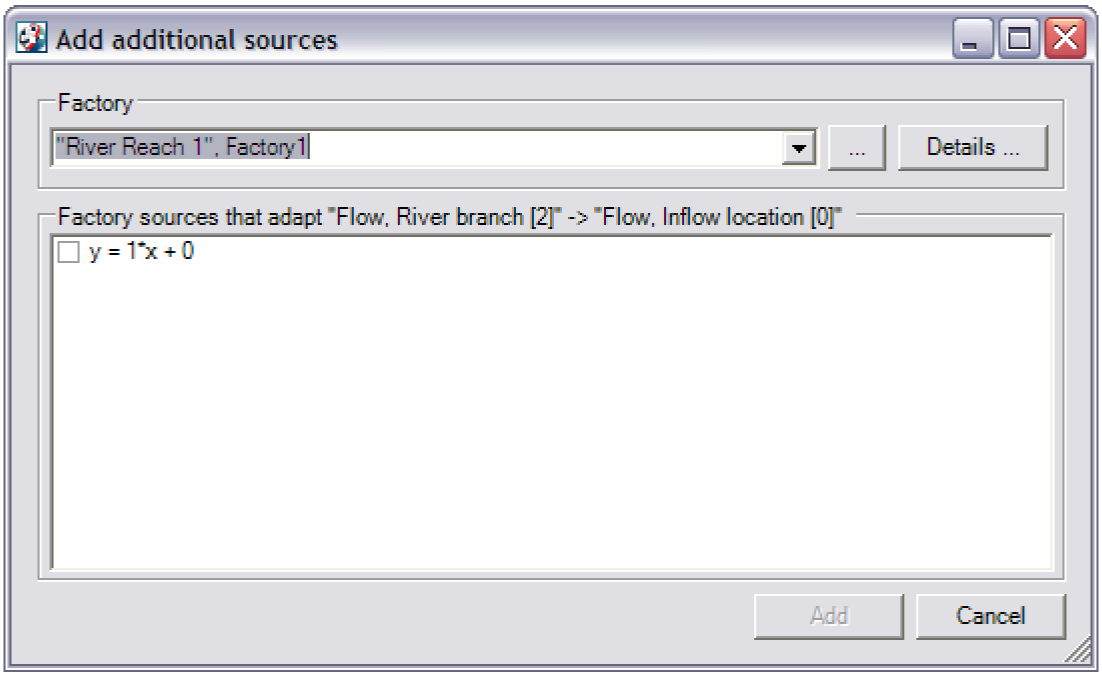
Component	
Component	River Reach 1
Misc	
Caption	Flow, River branch [2]
Consumers	(Collection)
Description	Flow, PolyLine[1] (,0)
Element Set	River branch [2]
Id	River branch [2]
Time Set	
Value Definition	Flow

Caption

Flow, River branch [2] -> Flow, Inflow location [0]

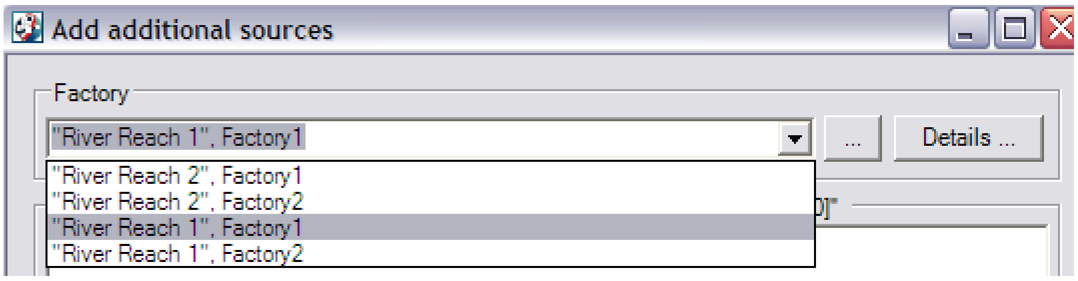
The dialog above shows the "Add Sources ..." button highlighted. As the tool-tip indicates, this will only be active if you have ticked both a source and target exchange item. Clicking this button will produce the following dialog.

Figure 2.16. Add Sources Dialog


Add additional sources
Factory
"River Reach 1", Factory1
...
Details ...
Factory sources that adapt "Flow, River branch [2]" -> "Flow, Inflow location [0]"
y = 1*x + 0
Add
Cancel

This dialog shows the available sources that can be chosen to adapt the current ticked source => target pair. Source adapters are accessed from dll's (assemblies) that implement the appropriate OpenMI standard interfaces. Collectively these different dll's of source adapters are referred to as factories since they create source adapters based on the source and target. The implementer of the source model can provide access to 'recommended' factories and by default the above dialog defaults to the first recommended factory of the source model; other composition model factories are also loaded into the drop down combo box. So for the example of a composition with two simple river models (same component but different models) the drop down combo could as follows.

Figure 2.17. Preloaded factories from composition

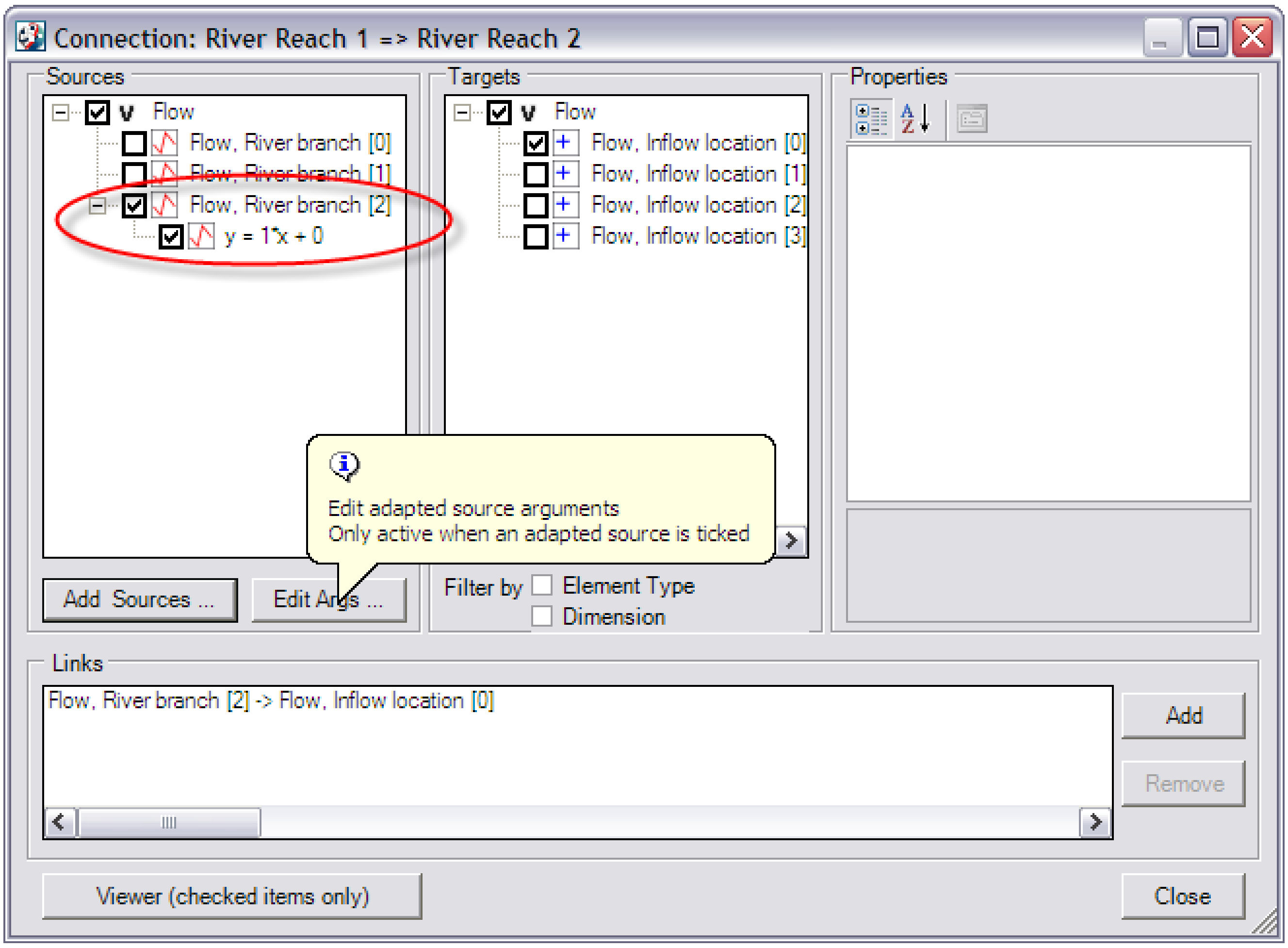

Add additional sources
Factory
"River Reach 1", Factory1
...
Details ...
"River Reach 2", Factory1
"River Reach 2", Factory2
"River Reach 1", Factory1
"River Reach 1", Factory2

Here both models in the composition have two factories with identical Ids Factory1 and Factory2, hence, the ids are also prefixed with the model id. It is possible that the model ids might also not be unique, so further information can be obtained by selecting the factory in question and clicking on the "Details ..." button.

The user is not restricted to using composition loaded factories, third party factories can also be added (see later).

In the example shown above; the first source recommended factory contains only one adapter. If the user ticks this item and then clicks the add button. The dialog is closed and the underlying connection dialog is modified as shown in the following figure (ringed area).

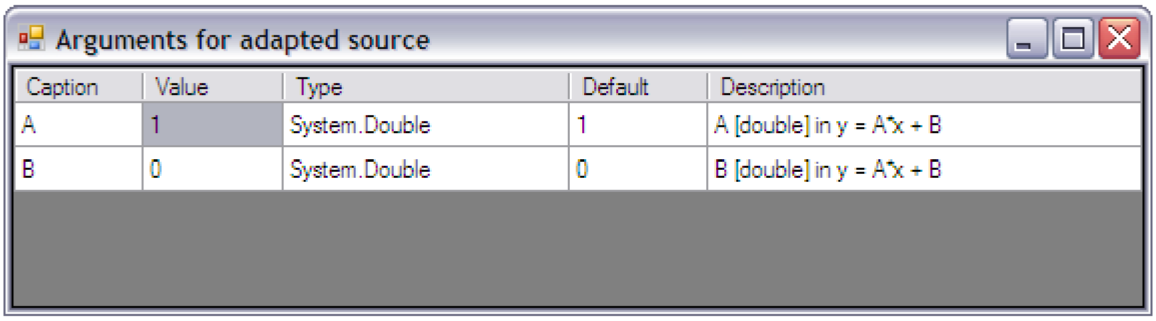
Figure 2.18. Added Source Adapter


Connection: River Reach 1 => River Reach 2
Sources
Flow
Flow, River branch [0]
Flow, River branch [1]
Flow, River branch [2]
y = 1*x + 0
Targets
Flow
Flow, Inflow location [0]
Flow, Inflow location [1]
Flow, Inflow location [2]
Flow, Inflow location [3]
Properties
Edit adapted source arguments
Only active when an adapted source is ticked
Add Sources ...
Edit Args ...
Filter by
Element Type
Dimension
Links
Flow, River branch [2] -> Flow, Inflow location [0]
Add
Remove
Viewer (checked items only)
Close

An additional source has been added, below the source item it adapts. This source can be used, and adapted, in the same way as any other source.

When an adapted source is ticked the "Edit Args ..." button becomes active; this is shown in the above figure where the buttons tool-tip is also shown. Clicking on this button will allow the user to edit any arguments that are associated with the adapted source. As shown in the next figure.

Figure 2.19. Edit Arguments dialog


Arguments for adapted source

Caption	Value	Type	Default	Description
A	1	System.Double	1	A [double] in y = A*x + B
B	0	System.Double	0	B [double] in y = A*x + B

This dialog allows the user to edit argument values. The columns in the table are

- Caption

  The name of the argument

- Value

  The value that the argument is currently set to. This column is the only editable column and a specific value might not be editable if the argument is specified as "read only".

- Type

  The type that the value should be convertible into

  i.e. if (as in this case) the type is "System.Double" the value must be in a valid format to convert into a double.

- Default

  The default value for this argument.

- Description

  Further information about the argument.

2.1. Third party factories

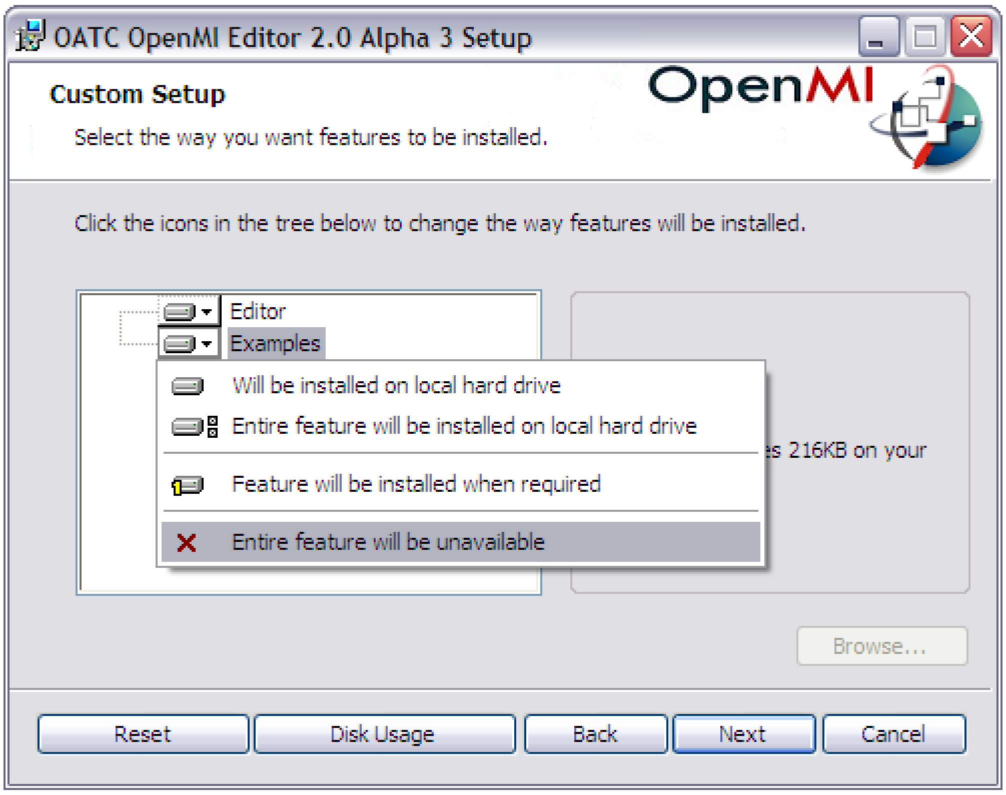
As mentioned earlier, the user is not restricted to using composition loaded factories, third party factories can also be added.

In the "Add Sources ..." dialog, click on the "..." button to open a "Open File" dialog. Use this to search for either a dll or exe file that implement adapter factories.

3. Examples

On installation, there is an option to install examples

Figure 2.20. Examples Installer


OATC OpenMI Editor 2.0 Alpha 3 Setup

Custom Setup

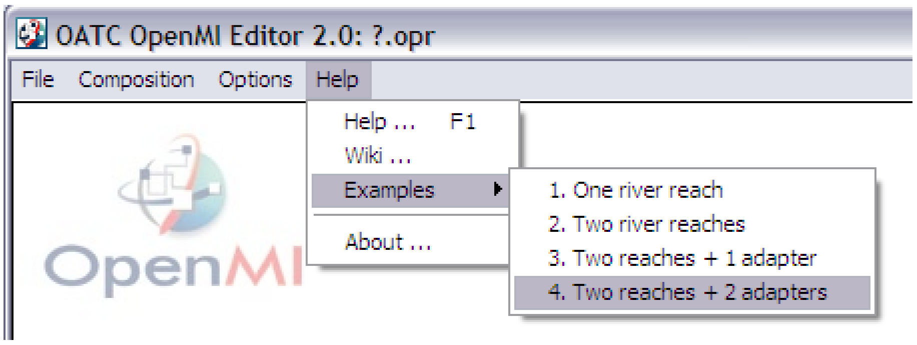
Select the way you want features to be installed.

Click the icons in the tree below to change the way features will be installed.

OpenMI
Editor
Examples
Will be installed on local hard drive
Entire feature will be installed on local hard drive
Feature will be installed when required
Entire feature will be unavailable
s 216KB on your
Browse...
Reset
Disk Usage
Back
Next
Cancel

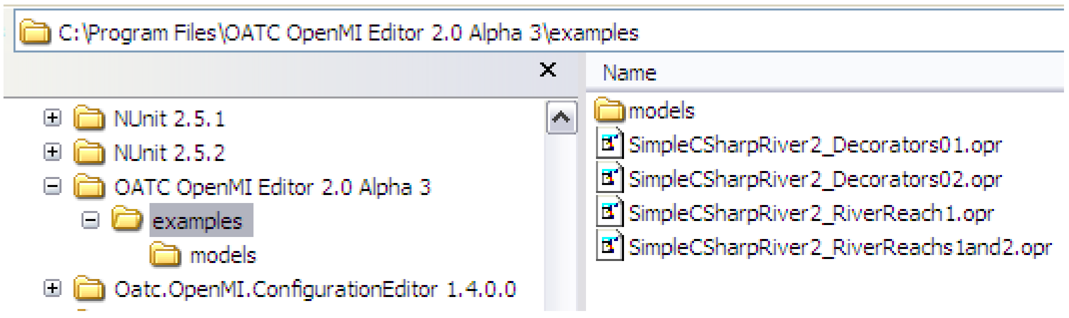
If you do not select the "Entire feature will be unavailable" option above, the examples will also be installed. They can be found under the Help menu.

Figure 2.21. Examples location



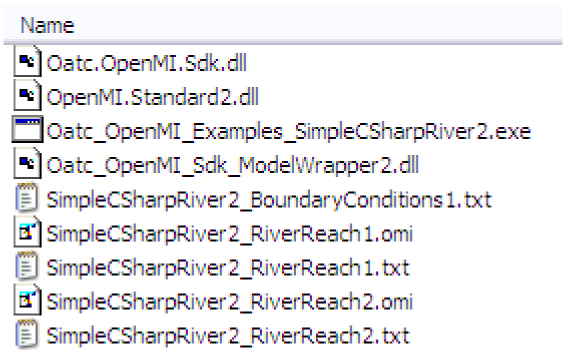
The examples are actually installed in the examples folder

Figure 2.22. Examples location


C:\Program Files\OATC OpenMI Editor 2.0 Alpha 3\examples
NUnit 2.5.1
NUnit 2.5.2
OATC OpenMI Editor 2.0 Alpha 3
examples
models
Oatc.OpenMI.ConfigurationEditor 1.4.0.0
Name
models
SimpleCSharpRiver2_Decorators01.opr
SimpleCSharpRiver2_Decorators02.opr
SimpleCSharpRiver2_RiverReach1.opr
SimpleCSharpRiver2_RiverReachs1and2.opr

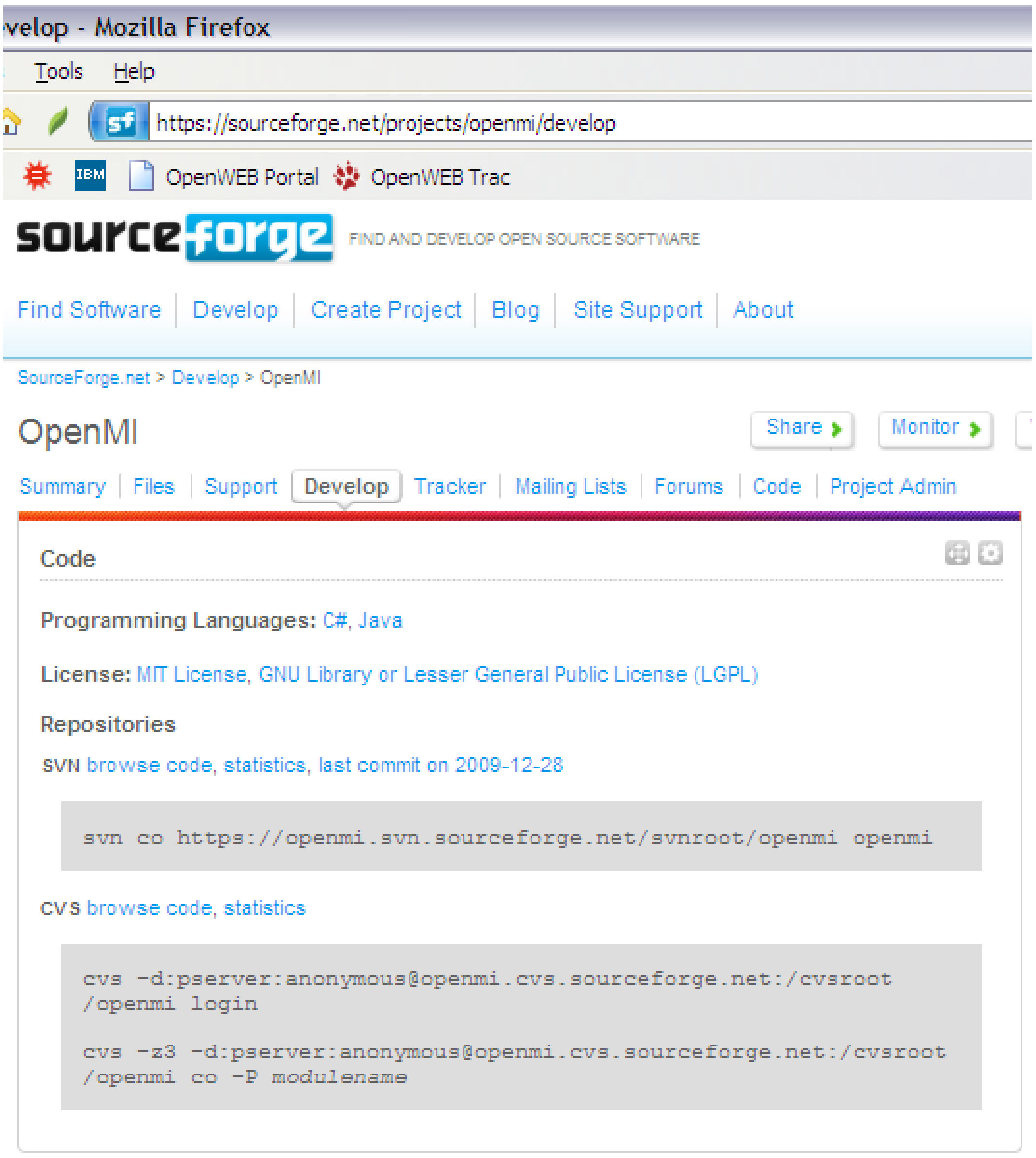
With the corresponding models and data in the models folder.

Figure 2.23. Models


Name
Oatc.OpenMI.Sdk.dll
OpenMI.Standard2.dll
Oatc_OpenMI_Examples_SimpleCSharpRiver2.exe
Oatc_OpenMI_Sdk_ModelWrapper2.dll
SimpleCSharpRiver2_BoundaryConditions1.txt
SimpleCSharpRiver2_RiverReach1.omi
SimpleCSharpRiver2_RiverReach1.txt
SimpleCSharpRiver2_RiverReach2.omi
SimpleCSharpRiver2_RiverReach2.txt

The Oatc_OpenMI_Examples_SimpleCSharp2.exe source code can be found on the OpenMI source forge site

Figure 2.24. Source forge site



OpenMI

Code

Programming Languages: C#, Java

License: MIT License, GNU Library or Lesser General Public License (LGPL)

Repositories

SVN browse code, statistics, last commit on 2009-12-28

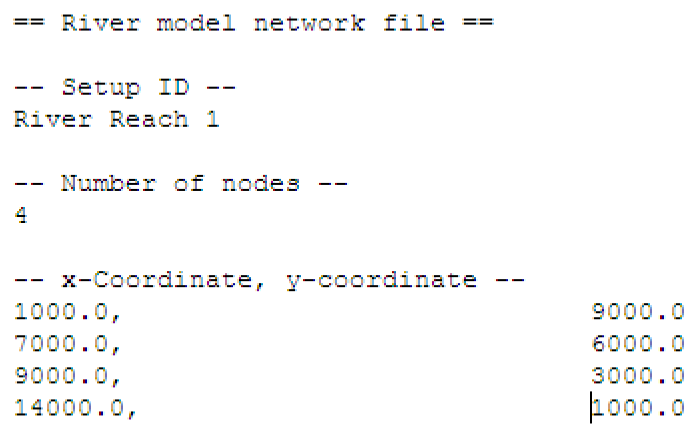
```
svn co https://openmi.svn.sourceforge.net/svnroot/openmi openmi
```

CVS browse code, statistics

```
cvs -d:pserver:anonymous@openmi.cvs.sourceforge.net:/cvsroot
/openmi login

cvs -z3 -d:pserver:anonymous@openmi.cvs.sourceforge.net:/cvsroot
/openmi co -P modulename
```

It models a simple river reach with a polyline. Inflow velocities are allowed on the polyline nodes (targets) and channel flows can be extracted from the polyline line segments (sources). The geometry (element sets) is defined in the model data file SimpleCSharpRiver2_RiverReach1.txt which contains

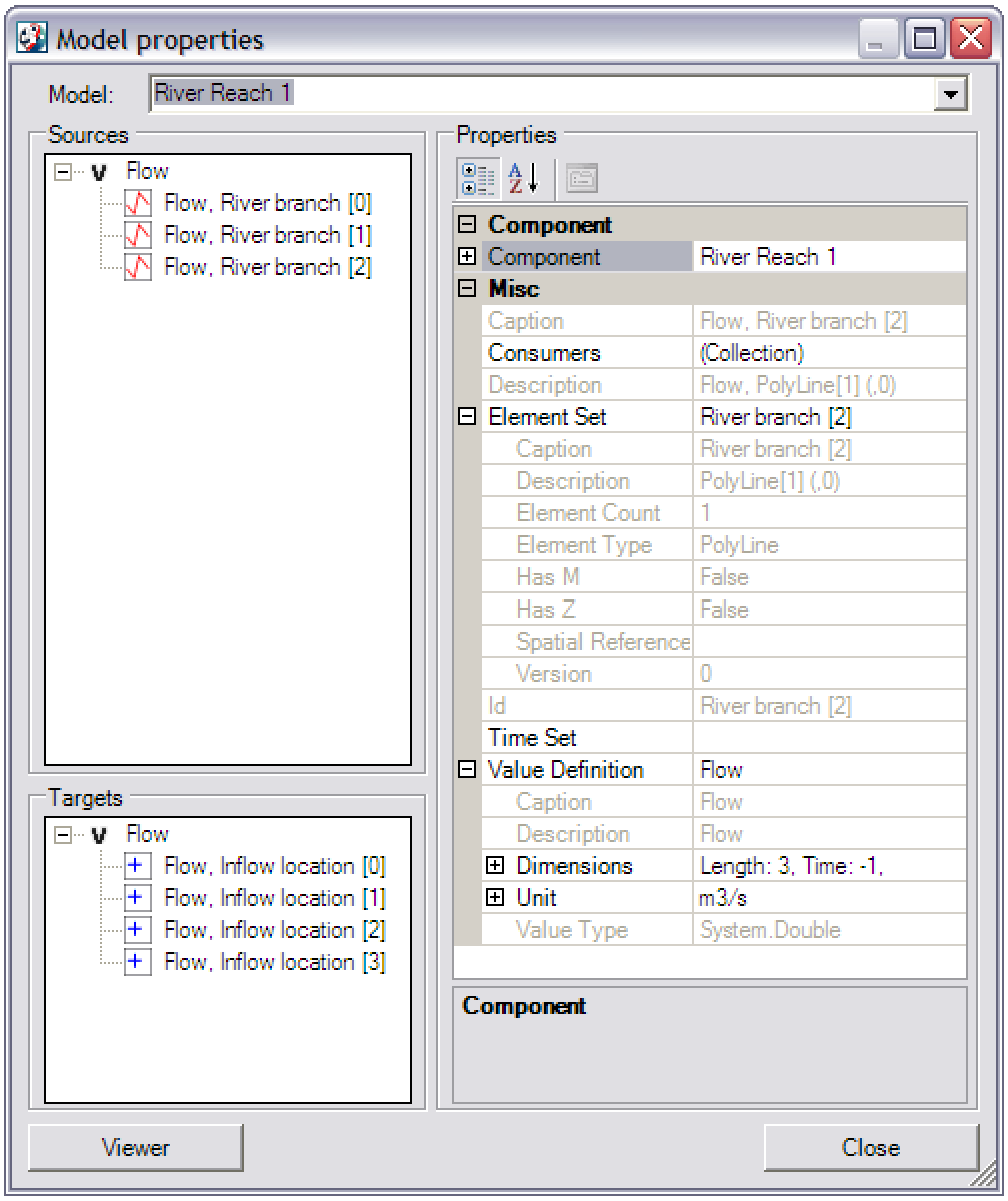
Figure 2.25. SimpleCSharpRiver2_RiverReach1.txt



```
== River model network file ==

-- Setup ID --
River Reach 1

-- Number of nodes --
4

-- x-Coordinate, y-coordinate --
1000.0,                              9000.0
7000.0,                              6000.0
9000.0,                              3000.0
14000.0,                             1000.0
```

Which will appear in the editor as

Figure 2.26. River Reach Model Properties



Model: River Reach 1

Sources
- Flow
  - Flow, River branch [0]
  - Flow, River branch [1]
  - Flow, River branch [2]

Targets
- Flow
  - Flow, Inflow location [0]
  - Flow, Inflow location [1]
  - Flow, Inflow location [2]
  - Flow, Inflow location [3]

Properties

Component	
Component	River Reach 1
Misc	
Caption	Flow, River branch [2]
Consumers	(Collection)
Description	Flow, PolyLine[1] (,0)
Element Set	River branch [2]
Caption	River branch [2]
Description	PolyLine[1] (,0)
Element Count	1
Element Type	PolyLine
Has M	False
Has Z	False
Spatial Reference	
Version	0
Id	River branch [2]
Time Set	
Value Definition	Flow
Caption	Flow
Description	Flow
Dimensions	Length: 3, Time: -1,
Unit	m3/s
Value Type	System.Double

Component

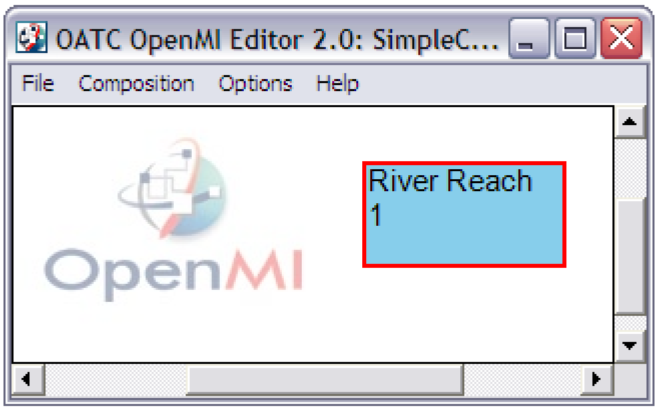
To run this model, one additional file is needed, which provides the boundary conditions SimpleCSharpRiver2_BoundaryConditions1.txt

Figure 2.27. SimpleCSharpRiver2_BoundaryConditions1.txt

```
== Simple River Boundary file ==

-- Leakage Coefficient ==
0

-- Simulation Start Time [YYYY-MM-DD HH:MM:SS] --
2004-10-07 16:38:32

-- Time Step Length [Sec] --
86400.0

-- Number of time steps --
6

-- Inflows (one row for each time step and one column for each node) --
1.1, 1.2, 1.3, 1.4
2.1, 2.2, 2.3, 2.4
3.1, 3.2, 3.3, 3.4
4.1, 4.2, 4.3, 4.4
5.1, 5.2, 5.3, 5.4
6.1, 6.2, 6.3, 6.4
```

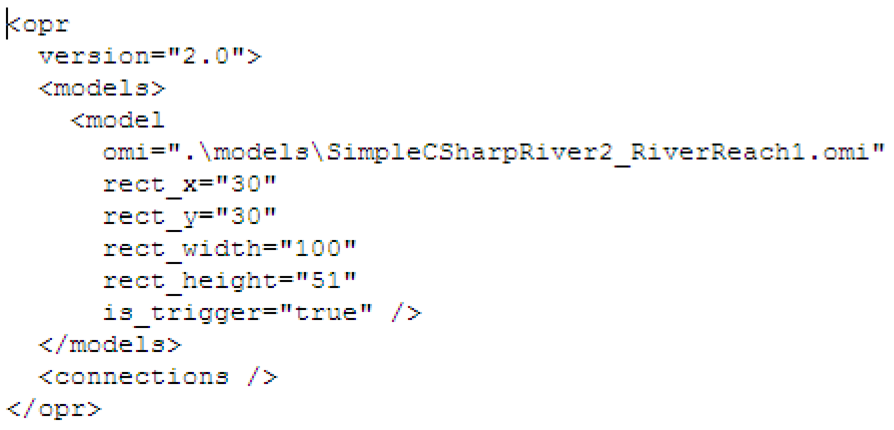
3.1. One river reach

Figure 2.28. One river reach



The loaded composition above is from SimpleCSharpRiver2_RiverReach1.opr

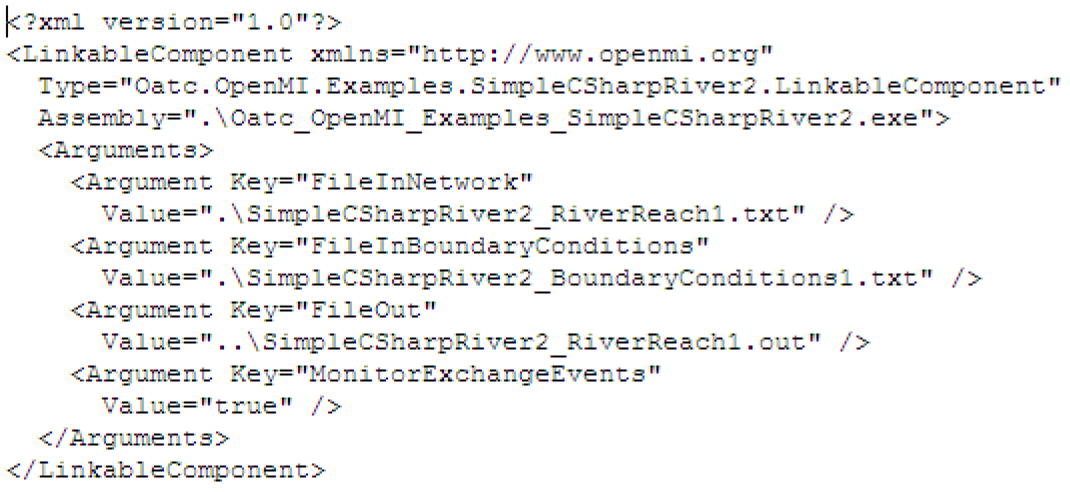
Figure 2.29. SimpleCSharpRiver2_RiverReach1.opr



```
<opr
  version="2.0">
  <models>
    <model
      omi=".\models\SimpleCSharpRiver2_RiverReach1.omi"
      rect_x="30"
      rect_y="30"
      rect_width="100"
      rect_height="51"
      is_trigger="true" />
  </models>
  <connections />
</opr>
```

Which references an OpenMI omi file for the model in question SimpleCSharpRiver2_RiverReach1.omi

Figure 2.30. SimpleCSharpRiver2_RiverReach1.omi

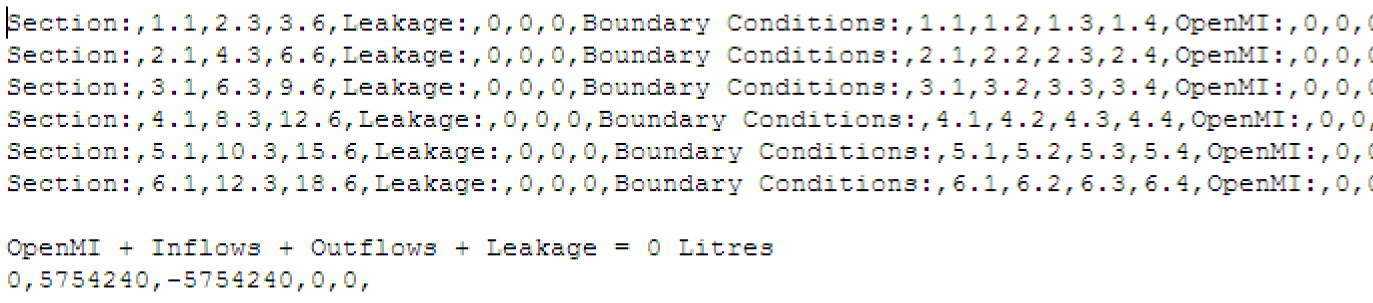


```
<?xml version="1.0"?>
<LinkableComponent xmlns="http://www.openmi.org"
  Type="Oatc.OpenMI.Examples.SimpleCSharpRiver2.LinkableComponent"
  Assembly=".\Oatc_OpenMI_Examples_SimpleCSharpRiver2.exe">
  <Arguments>
    <Argument Key="FileInNetwork"
      Value=".\SimpleCSharpRiver2_RiverReach1.txt" />
    <Argument Key="FileInBoundaryConditions"
      Value=".\SimpleCSharpRiver2_BoundaryConditions1.txt" />
    <Argument Key="FileOut"
      Value="..\SimpleCSharpRiver2_RiverReach1.out" />
    <Argument Key="MonitorExchangeEvents"
      Value="true" />
  </Arguments>
</LinkableComponent>
```

In this file are listed values specific to the model in question, this example defines the location of the input and output files used and generated during a run.

This very simple composition consists of only one river reach model. Although we are running it using OpenMI there are no OpenMI connections. Running it produces a results file

Figure 2.31. SimpleCSharpRiver2_RiverReach1.out



```
Section:,1.1,2.3,3.6,Leakage:,0,0,0,Boundary Conditions:,1.1,1.2,1.3,1.4,OpenMI:,0,0,
Section:,2.1,4.3,6.6,Leakage:,0,0,0,Boundary Conditions:,2.1,2.2,2.3,2.4,OpenMI:,0,0,
Section:,3.1,6.3,9.6,Leakage:,0,0,0,Boundary Conditions:,3.1,3.2,3.3,3.4,OpenMI:,0,0,
Section:,4.1,8.3,12.6,Leakage:,0,0,0,Boundary Conditions:,4.1,4.2,4.3,4.4,OpenMI:,0,0
Section:,5.1,10.3,15.6,Leakage:,0,0,0,Boundary Conditions:,5.1,5.2,5.3,5.4,OpenMI:,0,
Section:,6.1,12.3,18.6,Leakage:,0,0,0,Boundary Conditions:,6.1,6.2,6.3,6.4,OpenMI:,0,

OpenMI + Inflows + Outflows + Leakage = 0 Litres
0,5754240,-5754240,0,0,
```

The lines starting with "Section:" show the flow rates after each time step. The time step is a constant 86400 s (1 day) and is specified in the SimpleCSharpRiver2_BoundaryConditions1.txt file. So we can see that this run for 6 days duration. The values are volume flow rates [litres/s]. Section refers to each line segment in the reaches polyline, so there are three values. Leakage is also along the section, again 3 values. The Boundary conditions indicate the flow in at each node (4 along the polyline), these values are also specified in the SimpleCSharpRiver2_BoundaryConditions1.txt file. The OpenMI values are also inflows

The last line in this file shows a volume balance. As the model is not connected to anything else there is no OpenMI contribution and the inflows should equal the outflows. The leakage is also zero as the leakage coefficient is set to zero in the boundary conditions input file.

The value 5754240 is obtained by summing the flow rates along the last segment and multiplying by the time step.

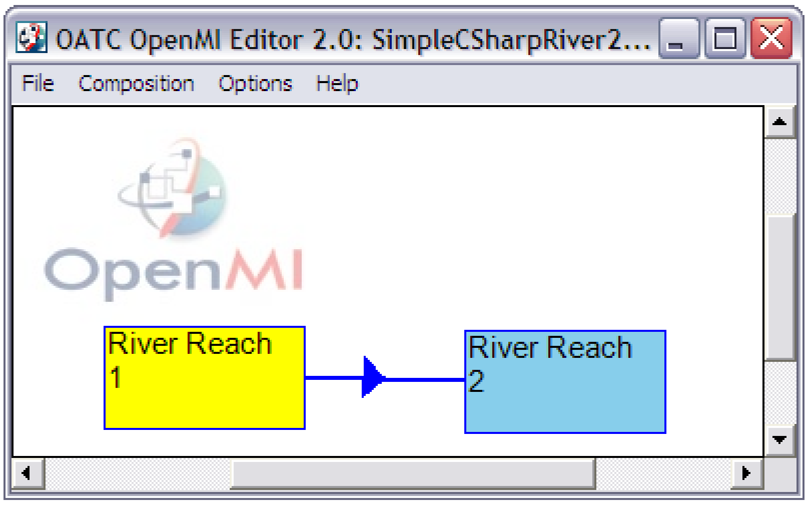
(3.6 + 6.6 + 9.6 + 12.6 + 15.6 + 18.6) * 86400 = 66.6 * 86400 = 575424

It is often worth testing a new model by running it in stand alone mode first and checking the results. This can often highlight any issues in the OpenMI implementation and also any issues with missing or incorrect model data. Clearly if model data is missing or incorrect; this will give clear indications as to what kind of composition and linkages will be required.

3.2. Two river reaches

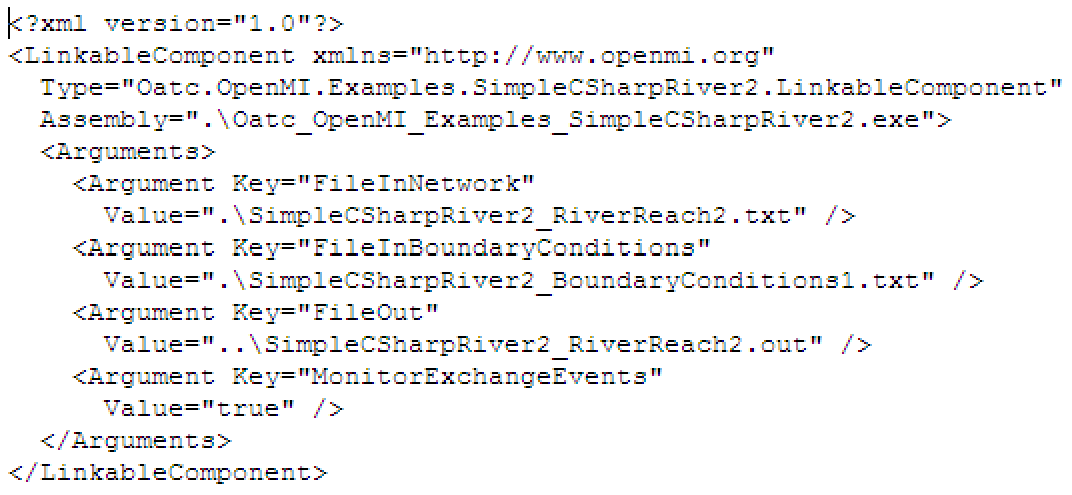
Two models of river reaches are added to the composition; both using the same linkable component.

Figure 2.32. Two river reaches



The corresponding model file (OMI) SimpleCSharpRiver2_RiverReach2.omi contains

Figure 2.33. SimpleCSharpRiver2_RiverReach2.omi

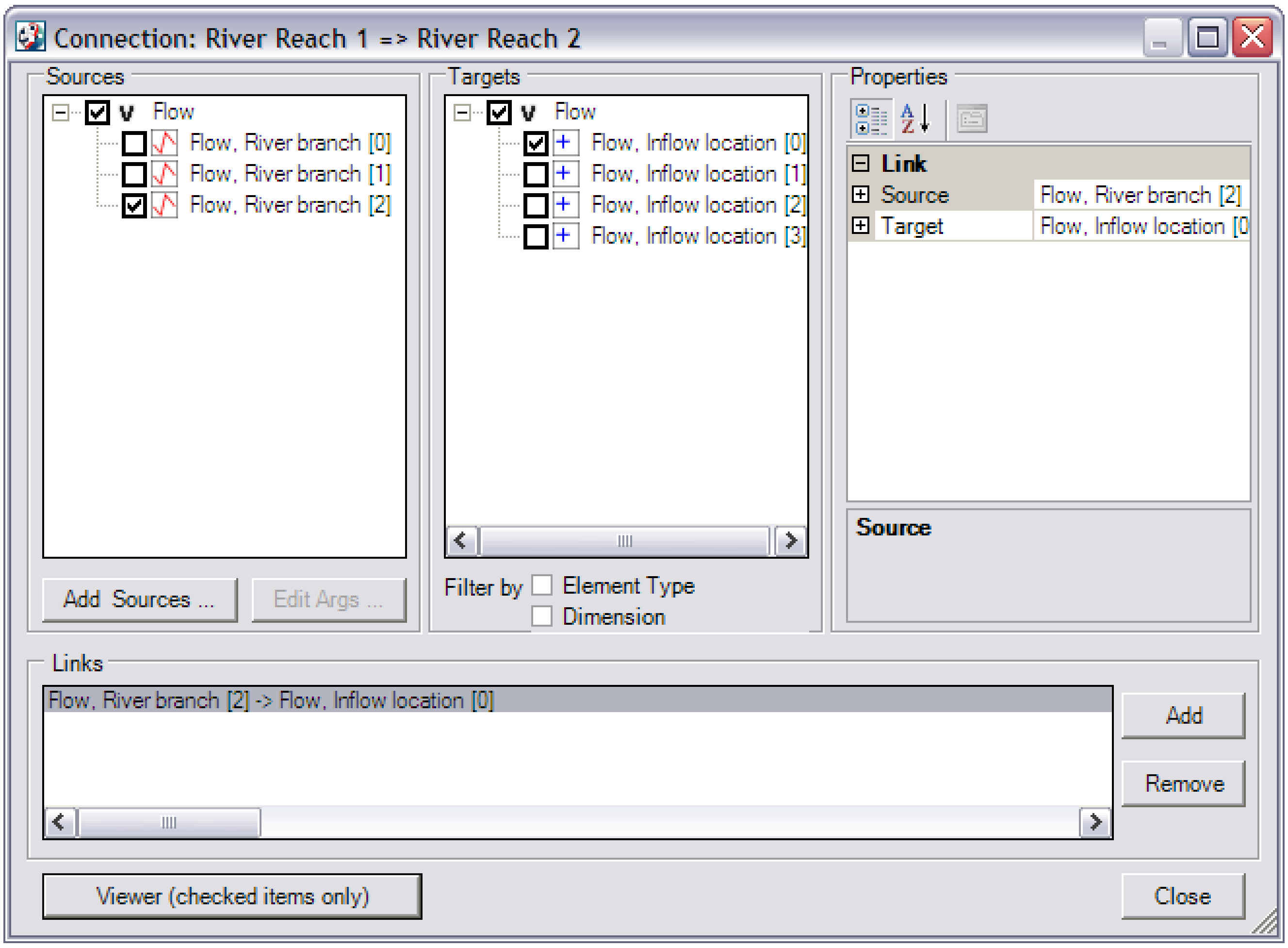


```
<?xml version="1.0"?>
<LinkableComponent xmlns="http://www.openmi.org"
  Type="Oatc.OpenMI.Examples.SimpleCSharpRiver2.LinkableComponent"
  Assembly=".\Oatc_OpenMI_Examples_SimpleCSharpRiver2.exe">
  <Arguments>
    <Argument Key="FileInNetwork"
      Value=".\SimpleCSharpRiver2_RiverReach2.txt" />
    <Argument Key="FileInBoundaryConditions"
      Value=".\SimpleCSharpRiver2_BoundaryConditions1.txt" />
    <Argument Key="FileOut"
      Value="..\SimpleCSharpRiver2_RiverReach2.out" />
    <Argument Key="MonitorExchangeEvents"
      Value="true" />
  </Arguments>
</LinkableComponent>
```

From which can be seen that the same boundary conditions file is used as in Reach 1 (same time step, time horizon and default inflows and leakage). The geometry is different, defined in SimpleCSharpRiver2_RiverReach2.txt and there is a different results file SimpleCSharpRiver2_RiverReach2.out.

The composition contains a single uni-directional link

Figure 2.34. Two river reaches linkage


Connection: River Reach 1 => River Reach 2

Sources
- [✓] v Flow
  - [ ] Flow, River branch [0]
  - [ ] Flow, River branch [1]
  - [✓] Flow, River branch [2]

Targets
- [✓] v Flow
  - [✓] Flow, Inflow location [0]
  - [ ] Flow, Inflow location [1]
  - [ ] Flow, Inflow location [2]
  - [ ] Flow, Inflow location [3]

Properties
- Link
  - Source: Flow, River branch [2]
  - Target: Flow, Inflow location [0

Source

Filter by [ ] Element Type [ ] Dimension

Links
Flow, River branch [2] -> Flow, Inflow location [0]

Which takes the flow out of the last channel of the first reach and adds it to the first node of the second reach. Effectively joining two river reaches together. Running this composition produces two output files SimpleCSharpRiver2_RiverReach1.out and SimpleCSharpRiver2_RiverReach2.out. SimpleCSharpRiver2_RiverReach1.out will contain the same values as for the stand alone example run (described in previous section) as it is acting as a OpenMI source in this composition i.e. links leave this model but none enter it. However, the downstream reach will be different, looking in SimpleCSharpRiver2_RiverReach2.out

Figure 2.35. SimpleCSharpRiver2_RiverReach2.out

```
Section:,1.1,2.3,3.6,Leakage:,0,0,0,Boundary Conditions:,1.1,1.2,1.3,1.4,OpenMI:,0,0,
Section:,5.7,7.9,10.2,Leakage:,0,0,0,Boundary Conditions:,2.1,2.2,2.3,2.4,OpenMI:,3.6,
Section:,9.7,12.9,16.2,Leakage:,0,0,0,Boundary Conditions:,3.1,3.2,3.3,3.4,OpenMI:,6.
Section:,13.7,17.9,22.2,Leakage:,0,0,0,Boundary Conditions:,4.1,4.2,4.3,4.4,OpenMI:,9
Section:,17.7,22.9,28.2,Leakage:,0,0,0,Boundary Conditions:,5.1,5.2,5.3,5.4,OpenMI:,1
Section:,21.7,27.9,34.2,Leakage:,0,0,0,Boundary Conditions:,6.1,6.2,6.3,6.4,OpenMI:,1

OpenMI + Inflows + Outflows + Leakage = 0 Litres
4147200,5754240,-9901440,0,0,
```

Reviewing the volume balance (last line). There is now an OpenMI contribution of 4147200 litres which should match the Outflow value in SimpleCSharpRiver2_RiverReach1.out

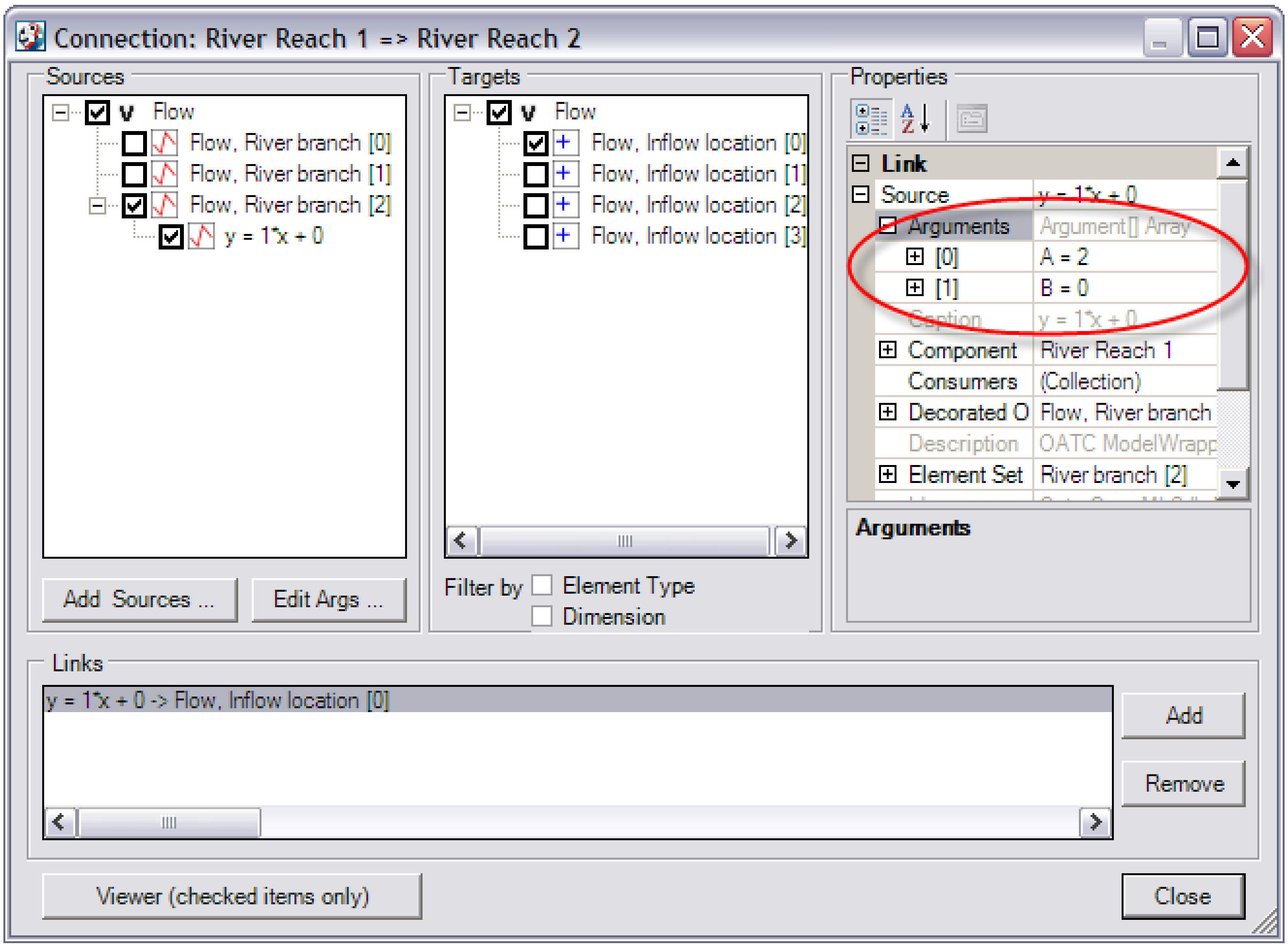
The value 147200 is obtained by summing the flow rates along the last segment of SimpleCSharpRiver2_RiverReach1.out and multiplying by the time step. Note that the last time step value is omitted as this value would be used by reach 2 on iteration 7 (if there had been one). These values are also in SimpleCSharpRiver2_RiverReach2.out; shown as OpenMI inflows.

(3.6 + 6.6 + 9.6 + 12.6 + 15.6) * 86400 = 48 * 86400 = 4147200

3.3. Two reaches + 1 adapter

This example is the same as the previous example except that the link is replaced with a link with an adapter applied to the source.

Figure 2.36. Two reaches + 1 adapter

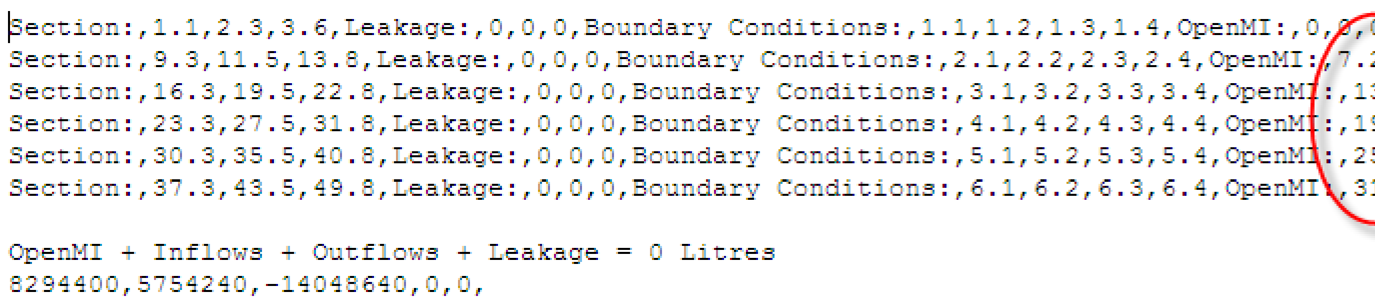


Link	
Source	y = 1*x + 0
Arguments	Argument[] Array
[0]	A = 2
[1]	B = 0
Caption	y = 1*x + 0
Component	River Reach 1
Consumers	(Collection)
Decorated O	Flow, River branch
Description	OATC ModelWrapp
Element Set	River branch [2]

Links: y = 1*x + 0 -> Flow, Inflow location [0]

The adapter is a linear equation y = Ax + B where A and B are arguments can be specified by the composition builder. In this example the values are A = 2 and B = 0 which results the reach 1 source values being doubled before being passed onto the reach 2 target. When run SimpleCSharpRiver2_RiverReach2.out contains

Figure 2.37. SimpleCSharpRiver2_RiverReach2.out (with adapter)



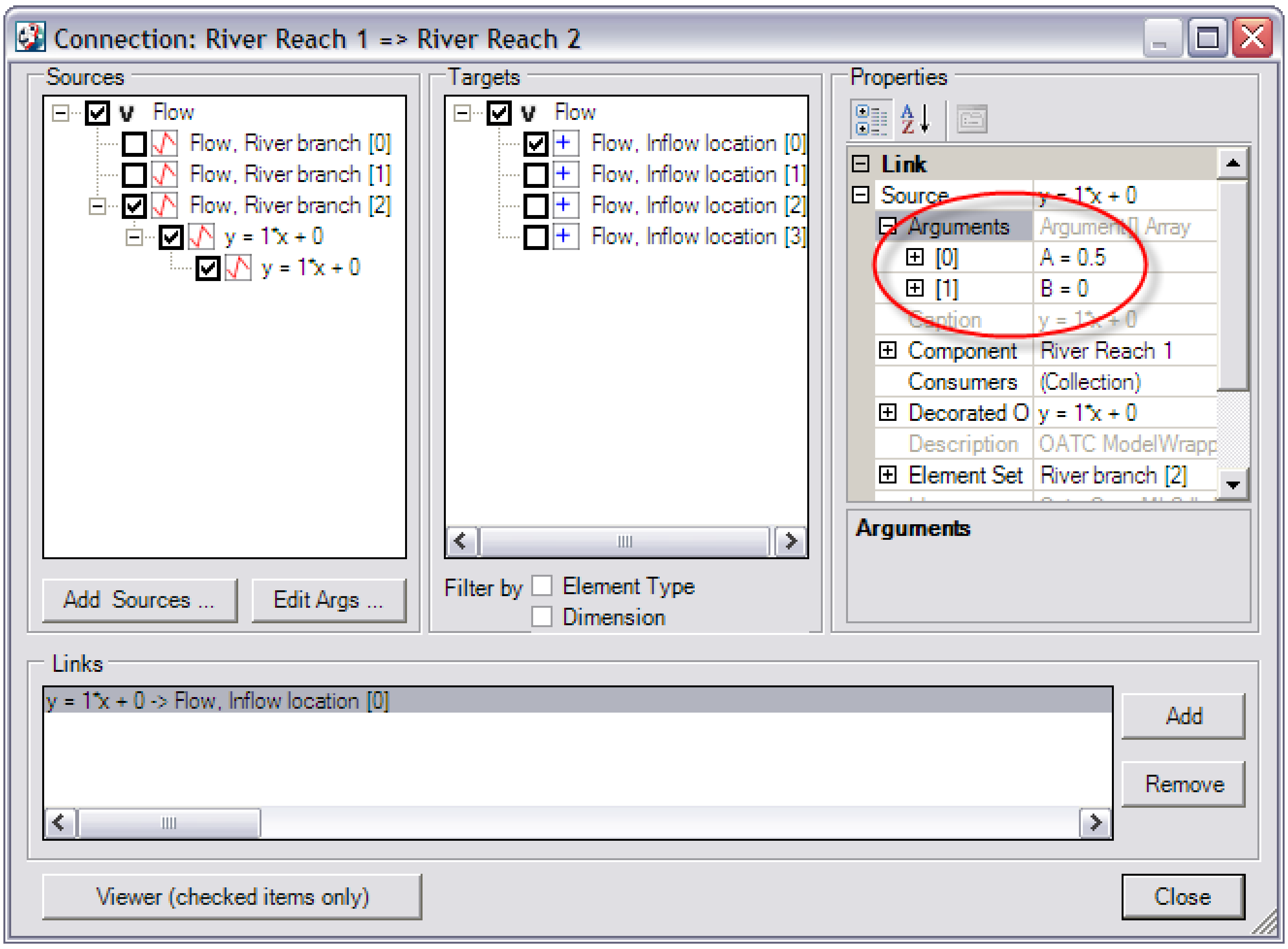
```
Section:,1.1,2.3,3.6,Leakage:,0,0,0,Boundary Conditions:,1.1,1.2,1.3,1.4,OpenMI:,0,0,
Section:,9.3,11.5,13.8,Leakage:,0,0,0,Boundary Conditions:,2.1,2.2,2.3,2.4,OpenMI:,7.2
Section:,16.3,19.5,22.8,Leakage:,0,0,0,Boundary Conditions:,3.1,3.2,3.3,3.4,OpenMI:,13
Section:,23.3,27.5,31.8,Leakage:,0,0,0,Boundary Conditions:,4.1,4.2,4.3,4.4,OpenMI:,19
Section:,30.3,35.5,40.8,Leakage:,0,0,0,Boundary Conditions:,5.1,5.2,5.3,5.4,OpenMI:,25
Section:,37.3,43.5,49.8,Leakage:,0,0,0,Boundary Conditions:,6.1,6.2,6.3,6.4,OpenMI:,31

OpenMI + Inflows + Outflows + Leakage = 0 Litres
8294400,5754240,-14048640,0,0,
```

These values are twice those in the previous example.

3.4. Two reaches + 2 adapters

This example is the same as the previous example except that the link is replaced with a link with two adapters applied to the source.

Figure 2.38. Two reaches + 2 adapters



Connection: River Reach 1 => River Reach 2

Sources
- ☑ v Flow
  - ☐ Flow, River branch [0]
  - ☐ Flow, River branch [1]
  - ☑ Flow, River branch [2]
    - ☑ y = 1*x + 0
      - ☑ y = 1*x + 0

Targets
- ☑ v Flow
  - ☑ Flow, Inflow location [0]
  - ☐ Flow, Inflow location [1]
  - ☐ Flow, Inflow location [2]
  - ☐ Flow, Inflow location [3]

Properties

Link	
Source	y = 1*x + 0
Arguments	Argument[] Array
[0]	A = 0.5
[1]	B = 0
Caption	y = 1*x + 0
Component	River Reach 1
Consumers	(Collection)
Decorated O	y = 1*x + 0
Description	OATC ModelWrapp
Element Set	River branch [2]

Arguments

Filter by ☐ Element Type ☐ Dimension

Links

y = 1*x + 0 -> Flow, Inflow location [0]

Both adapters are linear equations y = Ax + B where A and B are arguments can be specified by the composition builder. In this example the values are A = 2 and B = 0 for the first adapter and A = 0.5 and B = 0 for the second adapter. This results in the reach 1 source values being doubled and then halved before being passed onto the reach 2 target. As a consequence the results in SimpleCSharpRiver2_RiverReach2.out will be the same as for the example without any adapters at all.

The second adapter is different from the first in that the first is obtained from the source factory (reach 1) whilst the second is obtained from a third party factory, Oatc_OpenMI_Sdk_ModelWrapper2.dll