

# What's new in OpenMI 2.0

OpenMI Association, Technical Committee  
December 2009

Introduction .....	1
Linking components .....	1
Requesting values and control flow .....	2
Pull driven and loop approach.....	3
Calibration and data assimilation support.....	3
Component status .....	3
Less time stepping model focused.....	4
Events.....	4
Data definitions .....	4
Quantity and Quality.....	4
Element set.....	4
Time set.....	5
ValueSet.....	5

## Introduction

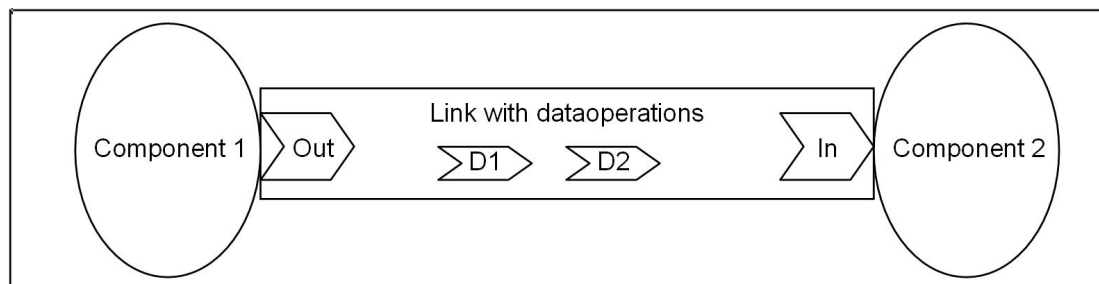
This document describes the new concepts in OpenMI 2.0, compared to OpenMI 1.4, and their elaboration in the class design.

The new features of OpenMI 2.0. can be summarized as:

- a more flexible way of linking
- more flexibility in the overall control flow
- less difference between spatial and temporal models
- support in categorized data values

## Linking components

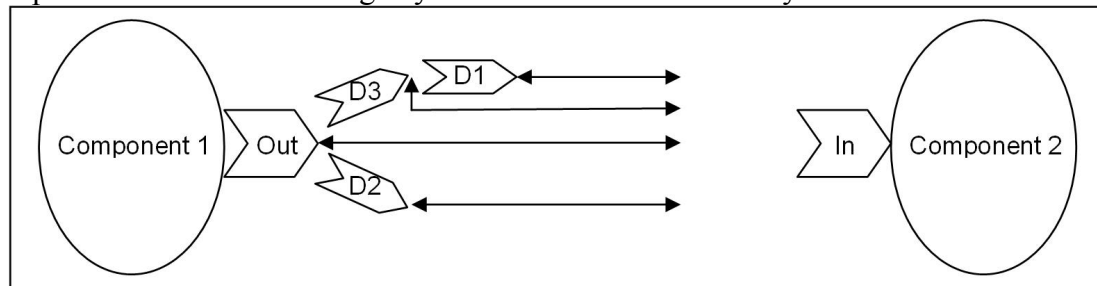
In OpenMI 1.4 the link object was used to connect components, containing a reference to the source component and the target component, and to the source output (exchange) item and the target input (exchange) item. Any operations in between the output item and the input item, to make them “compatible”, was handled by a series of DataOperations, also specified by the link object.



In OpenMI 2.0 a link between a source component and a target component is realized by a direct reference between an output item and an input item. The link, as a separate object, has been removed from the specification. If the values of an output do not exactly meet the way they are needed, you can “adapt” an output by means of a so called adapted output. This opens the possibility to add additional data-operations in between the original output and input item, realizing a truly piping and filtering pattern.

The adapted outputs take over the role of data operations in OpenMI 1.4. Typically, such intermediate adapted outputs are spatial and time interpolations and unit conversions. In the sequence of adapted outputs it is defined explicitly in which order such adapting operations are carried out. Also the possible data flows are extended, because each output and adapted output may be reused and connected to several requesting input items.

In the figure below there are 3 decorators added using the same output item, and an input item can connect using any of the 4 relations marked by the arrow-lines.



To implement this, the `GetValues` call of the linkable component has moved to the output item. The data operation is gone and instead, a decorator output item is introduced. The decorator output item is a configurable output item, in order to control operations like unit conversion and interpolation. The adapted output has a `Refresh()` method, that is used to notify an adapted output that the values of its parent output may have been changed.

Whereas in OpenMI 1.4 the requested time stamp or span was the main argument in the `GetValues()`-call, in OpenMI 2.0 the argument has become a full 'query specification'. Of course the requested time stamp or span can still be provided, but now also the requested element set can be specified, so that OpenMI 2.0 is more fit for usage GIS-systems than 1.4 was.

### ***Requesting values and control flow***

In OpenMI 1.4, values were asked from a linkable (model) component for a certain time and for a certain link. The component had the responsibility to return values meeting the requirements specified through the data definition specified in the link, basically the target quantity and target element set. In case calculation required input from other components, those would be triggered as well (the pull-based chain computation approach).

In OpenMI 2.0, the link identifier is gone, because it is implicitly defined by the sequence of output items. The data definition now is passed in the `GetValues` call,

containing the requested time stamps or spans, element set and value type (note that multiple time stamps / spans are supported).

### **Pull driven and loop approach**

In OpenMI 1.4, the control flow was purely pull driven, since linkable components were triggered to perform an action when there was a request to produce data.

In OpenMI 2.0 this is still the same, but there is an additional control flow possible, with a controller from outside. This controller can invoke each linkable component separately, by calling its Update() method. This will let the linkable component progress to its next step, but only if all necessary data for that component is available. If the data is not available yet (because another component has not computed it yet), the component specifies its state as "waiting for data", and returns the control flow. When the required data from other components is available indeed, the computation is performed, and the component sets its state to "updated".

A typical application of this approach could be an operational system, where the component that gathers the observations returns "waiting for data" as long as there are new measurements, and returns "updated" as soon as new observations have come, in which case, and only then, the forecasting components Update() method will be invoked.

Note that the loop approach doesn't offer wider usage of linked components, it is only an alternative way of controlling the data flow.

Implementing the loop approach is optional. A linkable component that does not support it is still OpenMI compliant.

### **Calibration and data assimilation support**

Development of calibration, optimization and data assimilation tools for OpenMI compliant models is possible with version 1.x of the Standard. However, the pull-driven architecture makes this sometimes complicated.

Since in 2.0 the data exposed for exchange is a precise reflection of the data inside the component, the implementation of set values methods has become possible. Hence, version 2 will be able to support setting values before letting a component run, thus facilitating the usage of OpenMI in calibration, optimization, data assimilation and decision support systems.

### **Component status**

Linkable components implement a property status. This property describes status of the component in every moment in the time. The diagram of the changes between different states is shown below (Figure 1). This mechanism is used in both the loop approach ("waiting for data" versus "updated") and the pull-driving approach (a GetValues-call on an output whereas its component is in the "waiting for data" state means that the GetValues-call was "self imposed, i.e. is the result of a bi-directional link).

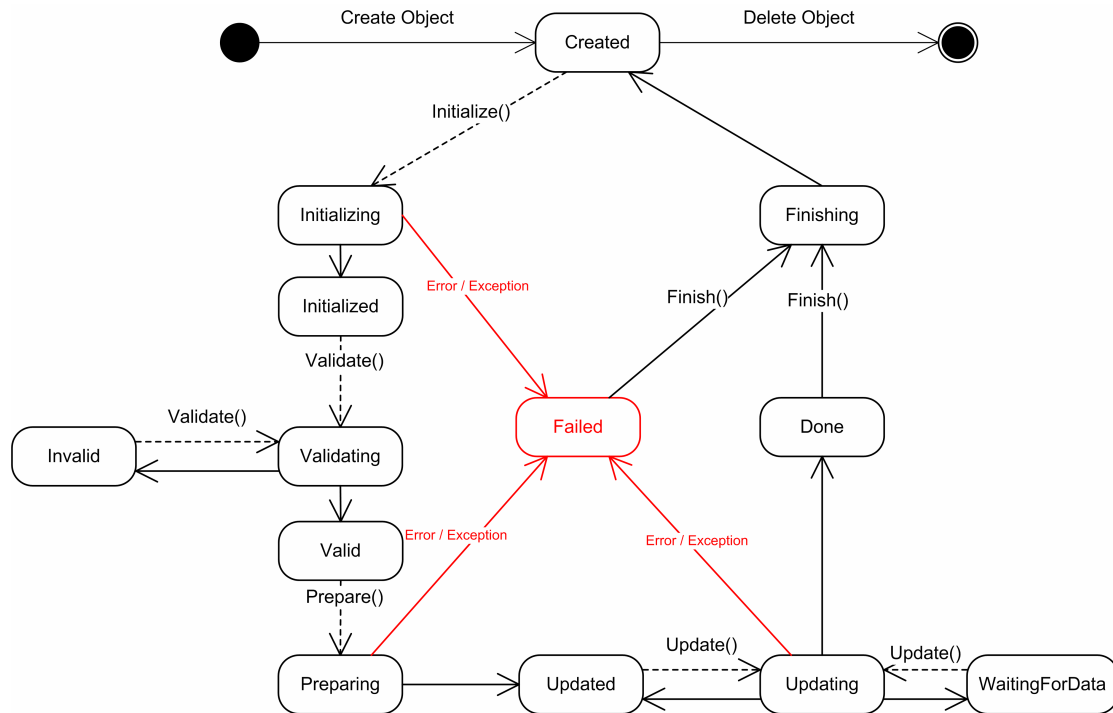


Figure 1 Linkable Component Status

### Less focused on time stepping models

The OpenMI 1.x Standard was developed specifically with the linkage between numerical time stepping models in mind. However, in systems for integrated modelling other components such as data providers, data bases, viewers, etc. are also used.

So OpenMI 2.0 accesses the various types of models as unified as possible:

- varying in time nor in space
- varying in time but not varying in space
- varying in time but not varying in space

### Events

Version 1.4 had its own event system. For version 2 the standard .Net mechanism is used. A similar system is available in Java and will be used.

### Data definitions

#### Quantity and Quality

Values in OpenMI 1.4 could only be quantitative, but OpenMI 2.0 adds the support for qualitative information (e.g. land use types, or indications like “hot” and “cold”, or “more sustainable” and “less sustainable”), and even custom types of values.

#### Element set

The element set definition was expanded to follow OpenGIS standard. The distinction between element sets in the horizontal plane and the 3-dimensional space (e.g. XYPoint / XYZPoint) has been removed, so georeferenced items now are positioned

on a Point, a Polyline or a Polygon, while a HasZ-property indicates whether the vertical dimension is involved or not. Also the M-coordinates (linear referencing) have been introduced.

The ISpatialReference interface has been removed. Instead, the element set now has a string that defines the spatial reference. This string follows the OGC standard WKT (well know text) for spatial reference.

### **Time set**

Handling of the time has been made similar to the handling of the element set. An exchange item now has a time set, specifying for which time stamps or spans the item can provide values (output), or wants to retrieve values (input).

### **ValueSet**

In OpenMI 1.4, the ValueSet could contain double precision scalars, or vectors consisting of double precision X/Y/Z-components, with one value for each element. A ValueSet instance held for one time stamp or time span.

In OpenMI 2.the 0 values stored in a value are organized in a two-dimensional array (times and elements), whereas the stored values can be objects of any type. Most comment type of course is the double precision real value, but there are many quantities that can be expressed by an integer value, by a boolean value, or by user defined types like a vector, or a list of possible categories, etc.