User's guide for Waqua as a component in OpenMI

SIMONA report number 2009-01

Rijkswaterstaat
Ministerie van Infrastructuur en Waterstaat

User's guide for Waqua as a component in OpenMI

Version number : Version 1.1, March 2010
Maintenance : see www.helpdeskwater.nl/waqua

Log-sheet

Version	Author	Date	Description
1.0	V$\mathcal{O}$Rtech	09-06-2009	c88648: initial version
1.1	E.J. Spee	30-03-2010	c1379: exchange items for Waqua-Swan
File location:		bo_omgeving/simona/src/waqua/waqomi/doc/usedoc	

Contents

Chapter 1

Introduction

Deltares participed in the 3-years European project called "OpenMI-Life". This is a demonstration project for OpenMI which is a standard that allows time-dependent models to exchange data at run-time. Using this standard, existing models, even those of different institutions, can be run together and exchange information at each timestep.

Two flow models of a different structure, each describing a separate region, from two different organisations can now be coupled. The first model is called "Kust-Zuid", and is run using the Waqua software, owned by the Dutch Rijkswaterstaat. It covers the Dutch part of the Schelde basin. The other model describes the flow in the Flemish part of the Schelde basin and is run using Mike11, a software package by the Danish Hydrological Institute (DHI).

The purpose of this document is to describe the way in which Waqua can be used as a component in an OpenMI-system (just "component", for short). In Chapter 2 an overview is given of the way a Waqua-model could function as a component. (Note: where we mention Waqua, we actually mean Waqua/Triwaq, i.e. 3D Triwaq models are supported just as well.) This overview is given mostly by discussing a simple example. In Chapter 3 it is explained which aspects of a Waqua model can be used in a coupling using OpenMI. Next, Chapter 4 describes the way in which a user presents input to the OpenMI-system when starting a calculation in OpenMI.

Chapter 2

Overview of an OpenMI-system

In this chapter, a simple example of an OpenMI-system is presented in order to give a global idea of the way in which coupled calculations are processed by OpenMI. The word "OpenMI-system" is used to refer to a number of coupled OpenMI components. Figure 2.1 illustrates the example used in this chapter. The example consists of seven OpenMI components. Five of these components are very simple components: they have one (time-dependent) output, which is a sine-function. These five sine-function components are coupled via OpenMI to the sixth component, which is the Waqua model of the Continental Shelf with grid size approximately 8km: the so-called CSM8-model.

The CSM8-model uses the inputs from the five sine-function components as waterlevel boundary conditions. The boundaries consists of many more grid points than the five coupled points; the Waqua component uses linear interpolation to determine the boundary water level for the boundary grid points between the coupled points.

The seventh component is called the *trigger*-component. Its only task is to ask the Waqua-component for some information at a certain time, for example the water level at the point named "Esbjerg" at a chosen time. The simple question, posed by the trigger-component, triggers a whole series of events. The Waqua component, whom the question is asked, does not have the requested information at the given time. Its current state is valid for its initial time. In order to obtain the requested information, it must do a number of time steps. These time steps can only be taken with the necessary boundary values given. The Waqua component alternately requests the sine-function components for the boundary values and performs time steps, until finally the original request can be fulfilled. Note that nothing would happen without the trigger-component.

The OpenMI-system is created in the OpenMI-editor, the program `Oatc.OpenMI.ConfigurationEditor`. When creating an OpenMI system, the user needs a number of component definitions, stored in so-called `omi`-files (usually with the file extension `.omi`). A user can load components in this program, after which the couplings can be defined. The way in which the OpenMI-system of Figure 2.1 looks in the OpenMI-qeditor, is shown in Figure 2.2. The OpenMI-system can be saved in an `opr`-file (with the extension `.opr`). A saved OpenMI-system can be run from the command line using the program `Oatc.OpenMI.CommandLine`.

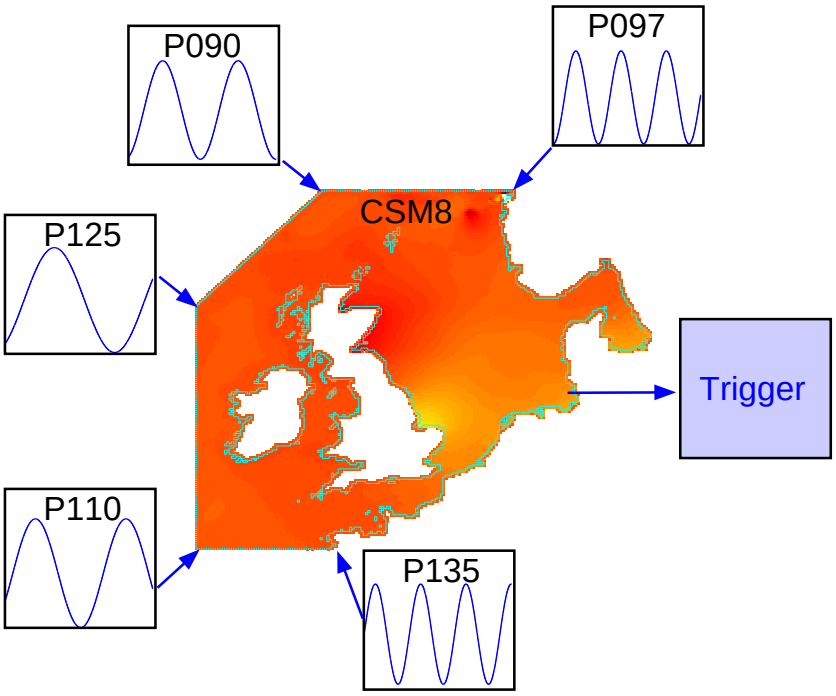

Figure 2.1: *An impression of a simple OpenMI-system, in which OpenMI Sine-function components provide the waterlevel boundary conditions for a CSM8-model.*

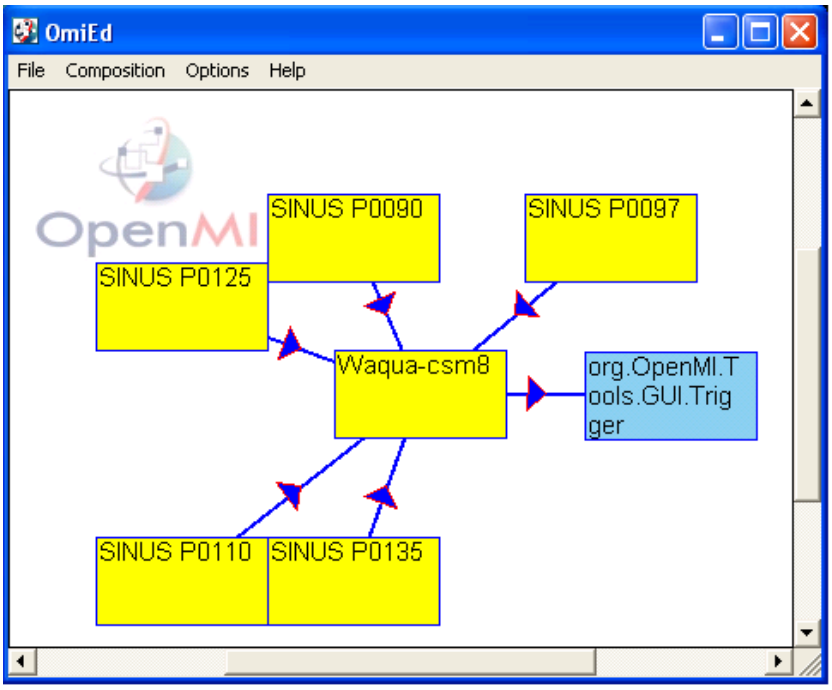

Figure 2.2: *The OpenMI-system shown in Figure 2.1, as it appears in the OpenMI-editor.*

Chapter 3

Exchange items for Waqua models

It has been determined in the software of the Waqua component what kind of information can be used as inputs by the Waqua component from other components, and what kind of information can be passed as output from a Waqua component to other components. Such possibly coupled pieces of information are called “exchange items”. In this chapter the exchange items for Waqua models are discussed.

3.1 Waqua inputs from other components

It was seen in the example of the previous section that the Waqua component can be coupled to another OpenMI component that provides the water levels which the Waqua component needs for its boundary conditions. In that case, the coupling concerns an input for the Waqua component and an output for the other component.

Apart from water level boundary conditions, all other boundary conditions, such as velocity, discharge and Riemann boundary conditions may be obtained as inputs from another OpenMI component. For boundaries with automatically distributed discharges, the coupled information is the value for the whole boundary. For all other boundaries, the coupled information is the boundary value at one of the end points of the open boundary as defined in the Waqua model. Also, the discharge at a discharge source point may be used as an input value by Waqua.

When creating a coupled system with a Waqua model, the user must take into account that the values obtained from the other component are not always applied directly and unchanged. The values given for the Waqua-inputs `REFL`, `WGHTHALFTIME` and `SMOOTHING` determine whether Waqua applies the coupled information unchanged, or whether changes are made to the inputs.

For the coupling with the wave model SWAN, Waqua accepts the quantities wave direction, wave force, wave height and wave period.

location	quantity	role	name
water level checkpoints	wl	providing	POINT name
current checkpoints	x_velocity or u_velocity	providing	POINT name
current checkpoints	y_velocity or v_velocity	providing	POINT name
cross sections	momentary_discharge	providing	CURVE name
cross sections	cumulative_discharge	providing	CURVE name
dicharge sources	discharge	providing	SOURCE name
dicharge sources	discharge	accepting	SOURCE name
disch-ad opening	discharge	accepting	OPENING name
opening points	u_velocity, u_discharge, v_velocity, v_discharge, or wl	accepting	"P<point number>"
whole grid	waterlevels	both	sgrid
whole grid	wave direction x, y	accepting	sgrid
whole grid	wave force x, y	accepting	sgrid
whole grid	wave height	accepting	sgrid
whole grid	wave period	accepting	sgrid
whole grid	u velocity	providing	ugrid
whole grid	v velocity	providing	vgrid

Table 3.1: *Exchange items created in the initialization phase of the Waqua component. The use of the keyword* NO_BACKTRANS *in the Waqua-input determines whether the current checkpoints get quantities* x_velocity *and* y_velocity *(east- and north-component of the velocity vector) or the quantities* u_velocity *and* v_velocity *(grid aligned components of the velocity vector).*

3.2 Waqua outputs to other components

In Section 3.1, a description was given of all the values which can be supplied to the Waqua component by other components. The Waqua component may also produce outputs to other components. Possible outputs are the time histories for flow variables: water levels at water level checkpoints, velocities at velocity checkpoints and discharges (momentary or cumulative) at cross-sections.

For the coupling with the wave model SWAN, Waqua can provide the quantities waterlevel and u, v velocities on the whole grid.

A complete list of the types of exchange items is given in Table 3.1.

Chapter 4

Installation and use of Waqua components

4.1 Installation

The use of Waqua components requires that the following steps be taken:

- Install the OpenMI editor, to be downloaded from `www.sourceforge.net`;
- Copy software from the directory `extern\OpenMI_1.4\bin`, which is used by the Waqua components, to a chosen directory. Important parts are:
  - `Deltares.OpenMI.Wrapper.dll`: the *Wrapper* for the Delft3DFlow, Sobek and Waqua components;
  - `Deltares.OpenMI.AccessServer.exe`, the *Access Server*.

  The roles of these files is of lesser importance for users. They are discussed in the programmer's manual of the OpenMI-component Waqua.
- Install the SIMONA software which contains the Waqua model. Some configuration settings for SIMONA are set in the file `%SIMONADIR%\etc\win32\Settings.inc`. Make sure that the full directory name of the location for the Wrapper and the Access Server are set in this file under the names `OPENMIWRAPPERDIR`.

4.2 Use of Waqua as a component in an OpenMI-system

To use the Waqua component, start up the OpenMI Configuration Editor. Now construct or load an OpenMI system with a Waqua component using the Configuration Editor.

When constructing OpenMI systems, you need an OpenMI input file (`.omi`-file) which describes the Waqua component. The script `make_omi.pl` is intended to help you make such input files. The script takes the following command line options:

`-runid` The run-identification of the component. The output SDS-file will be `./SDS-<runid>`.

-bufsize The (initial) buffer size to be used in the Waqua component (optional).

-sds The SDS-file with the input for the component. The input file must not be identical to the output file, i.e. the input file must not be `./SDS-<runid>`.

-siminp The simulation input-file from which the input SDS-file may be created. Make sure that either `-sds` or `-siminp` is specified, and not both.

-nmdbg Name of the input file with debug-settings (optional).

-ownproc Flag `yes/no`: create a separate process for this Waqua component. Use this setting if there is more than one Waqua component in the OpenMI system. Running a component in a separate process avoids conflicts between components with respect to the ownership of common blocks. The default setting is `-ownproc yes`, so you only need to supply this option when you want to run the Waqua component in the same process as OpenMI itself.

Once a correct `omi`-file has been obtained, the user can easily change it using her or his favorite (xml- or text-) editor.