Memo

To

-

Confidential until \memoConfidentialUntil{...}

Date	Our reference		Number of pages
11 January, 2013	0.00.04		12

Contact person	Direct line	E-mail
Irv Elshoff	+31 (0)88 335 8553	Irv.Elshoff@Deltares.NL
Adri Mourits	+31 (0)88 335 8332	Adri.Mourits@Deltares.NL
Jan Mooiman	+31 (0)88 335 8568	Jan.Mooiman@Deltares.NL

Subject

Draft: The DeltaresHydro XML-Based Configuration File

Copy to

-

1 Introduction

This document describes the (proposed) main configuration file for the DeltaresHydro suite. The plan is to replace all of the individual input files — e.g., the Flow2D3D mdf-file (master definition Flow file) — with a uniform XML-based configuration file. This process can and will be gradual.

XML has been chosen because it is powerful enough and well established. XML is not the best choice for efficiently representing multidimensional numerical data. NetCDF is better. For geo-referenced points, shapes, layers, etc. a GIS representation is best. These other files will be referenced from within the configuration file.

In some sense XML is overkill for what's left over. The most suitable alternative is the Windows-inspired INI file. An INI file file consists of sections, each with name value pairs. There is no official INI standard, and many slightly varying interpretations (and implementations). Microsoft has abandoned INI files. This does not mean Deltares could not use INI files, or something similar, however.

One stated advantage of INI files is their readability. Users would like to be able to edit the input file manually. XML is perceived as being complex, and unreadable. The are certainly many XML files that fit into this category, complexity and unreadability are not inherent to XML.

The DeltaresHydro XML input file format has been designed to clear, concise and amenable to manual editing using a standard text editor.[1]

[1] Many common editors support XML syntax highlighting. This document is best viewed in color. Please consider

The next section describes general design principles and file structure. Section 3 documents specific XML elements for DeltaresHydro. The last section describes the DeltaresHydro XMLTree C++ library that can be used to parse the configuration file.

2 General Principles

2.1 XML Background

An XML file consists of a properly nested set of elements, contained within a single root element. Elements are delimited by a matching pair of start and end tags, or consist of a single empty tag. Start and empty tags can contain any number of attributes, which associate a string value to an attribute name. Between start and end tags other, nested elements or so-called character data may appear. For more details on XML itself see http://en.wikipedia.org/wiki/XML .

There are numerous ways to use XML to represent the same information. A few are illustrated here using an airport as an example. One end of the spectrum is to limit the number of types of tags and attributes to an absolute minimum. The description of an airport would look something like:

```
<airport>
   <property>
      <propertyName>CommonName</propertyName>
      <propertyValue>Rotterdam The Hague Airport</propertyValue>
   </property>
   <property>
      <propertyName>ICAO airport code</propertyName>
      <propertyValue>EHRD</propertyValue>
   </property>
</airport>
```

This approach is commonly used, e.g., in Mac OS/X properly lists, GigE Vision camera specifications, and elsewhere. It is very verbose and not easy find parts to edit manually. Just reading such a file is tedious, even with syntax highlighting.

Using element attributes and empty tags makes the file more readable:

```
<airport>
   <identification>
      <property name="Name" value="Rotterdam The Hague Airport" />
      <property name="ICAO code" value="EHRD" />
      <property name="ARP" value="N 51.95694 E 4.43733" />
   </identification>
   <frequencies>
      <property name="APP" value="127.025" />
      <property name="TWR" value="118.2" />
      <property name="ATIS" value="110.4" />
   </frequencies>
</airport>
```

This is very much like an INI-file: sections with keyword value pairs.

If this were for a real pilots' airport database we would want to ensure that certain basic information, such as the ICAO code, location and tower frequency are always present.

---

our Environment before hitting the Print button to generate a paper copy that will be quickly discarded.

Ensuring that an XML file conforms to a certain rules — beyond basic XML syntax — is done using XML schemas.

The next step is to promote important items to full-fledged elements and use more attributes. One approach starts with:

```
<airport name="Rotterdam The Hague Airport" icaoCode="EHRD" towerFreq="118.2" ...>
   <runway
      designation="24" trueBRG="237.12"
      tora="2200",
      ilsFreq="110.9" ilsID="RVS"
      />
   <runway
      designation="06" trueBRG="057.10"
      tora="2200",
      ilsFreq="110.9" ilsID="ROS"
      />
</airport>
```

In fact, all essential information could be expressed as attributes, and formatting it this way might look nice. A fundamental problem with attributes is that most XML tools will reformat element start/empty tags to a single line, and it is not possible to comment out individual attributes.

2.2 The DeltaresHydro Way

2.2.1 Names

Element names should start with a lower case letter, consists of letters and digits only, use camel-Case for compound words, and be reasonably short.

Attribute names follow the same rules as element names.

2.2.2 Attributes

Start and empty tags should have a limited number of possible attributes (no more than about three or four).

For specifying simple values the attribute form is preferred over content data. The attribute name depends on the type of values possible. The following standard attribute names are to be used:

- `value` specifies a single scalar numerical value, either integer or floating point.
- `range` specifies multiple scalar numerical values, either integer or floating point, in a range between a starting and ending value. MATLAB syntax is used: $A : B$ means the range from A to B (inclusive) with an increment of $+1.0$. $A : I : B$ uses the non-zero increment I.
- `units` specifies the SI (or other) units of the data the element defines. E.g., pressure for aviation could be expressed in "hPa" or "mmHg". Use of the units attribute does not imply an automatic conversion by the base XML configuration file parser.
- `pathname` specifies the path to a file or directory. Windows and Unix (Linux, OS/X, BSD, ...) forms may be used interchangeably (see Section **??**). *All references to files and directories* **must** *be contained in a pathname attribute.* This is necessary to be able to collect the set of files needed for running the simulation remotely, e.g., in the cloud.
- `url` specifies a file or resource that is available using a wide variety of internet protocols.
- `name` is to be used for short character strings other than pathnames or URLs.
- `state` is used to specify one over several mutually exclusive conditions. Boolean values are `true` or `false`, `on` or `off`, `yes` or `no`, and `enabled` or `disabled`. True, on, yes

and enabled are logically equivalent, and will be converted to a Boolean true by the XMLTree library (see Section 4). Non-Boolean values can be arbitrary strings, but should be a member of a fixed element-specific set of values.

Other attribute names may be used, but should not be for purposes akin to the standard names.

2.2.3 Empty Elements or Start/End Tags

Another design choice is whether to use attributes or content data (CDATA) to specify values. For example:

```
<pressure>1013.25</pressure>
```

versus

```
<pressure value="1013.25" />
```

Empty element tags (the latter) should be used to specify simple, scalar values (or ranges). For multiple values start/end tags should be used. For example:

```
<sigmaLayers>
   0.100
   0.150
   0.250
   0.250
   0.150
   0.100
</sigmaLayers>
```

3 DeltaresHydro XML Elements

The DeltaresHydro configuration file consists of a collection of top-level elements related to primary DeltaresHydro modules within the XML root element. The order of the top-level elements is arbitrary.

3.1 The Root Element

The root elements identifies the XML file as a DeltaresHydro configuration file.

```
<deltaresHydro start="flow2D3D"
   xmlns="http://deltares.nl/DeltaresHydro"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://deltares.nl/DeltaresHydro
                       http://schemas.deltares.nl/DeltaresHydro/1.234.xsd">
   <!-- top-level elements -->
</deltaresHydro>
```

The XML name space identifier `http://deltares.nl/DeltaresHydro` looks like a URL, but it is just an arbitrary string with no special significance other than to uniquely identify our element tags. ("foobar" is probably already taken.)

The location of the XSD file is presently fictitious.

3.1.1 Versioning

The base name of the XSD file contains a major and minor version number; in the example above 1.234. This refers to the XML file, not to DeltaresHydro itself.

The current version of DeltaresHydro shall support the current and previous major versions of the configuration file, and may support earlier versions. Major version increments will be rare events. Conversion scripts will be provided to upgrade configuration files when absolutely necessary.

The minor version number is used to indicate additions, e.g., support for a new module or feature. Minor versions increments will be common events. Older versions of DeltaresHydro will always be able to read all newer minor versions of the configuration file; in most cases newer functionality will not be available.

3.2 Top-level Elements

DeltaresHydro recognizes the following top-level elements.

- `flow2D3D` for 2D and 3D hydrodynamics and morphology.
- `wave` for surface waves (SWAN).
- `waq` for water quality.
- `mesh` for specifying the computational grids or meshes used by the above elements, including subdomains for domain decomposition.
- `delftOnline` for online access to internal data structures.
- `control` provides a means for controlling complex, compound simulations.

The first three correspond to the major DeltaresHydro components; at least one of these must be present.

3.2.1 The `flow2D3D` Element

This element configures the Flow2D3D component of DeltaresHydro. The most simple form for a single-domain simulation is:

```
<flow2D3D>
   <library pathname="/opt/delft3d/lnx64/flow2d3d/bin/libflow2d3d.so" />
   <mdfFile pathname="f34.mdf" />
   <gridFile pathname="f34-coarse.grd" />
</flow2D3D>
```

The elements that can appear within the `flow2D3D` element are:

- `<library pathname="..."/>` specifies the path to the Flow2D3D shared library (Linux), dynamic library (Mac OS/X) or DLL (Windows). This element is mandatory.
- `<mdfFile pathname="..."/>` specifies the master definition file for single-domain simulations.
- `<gridFile pathname="..."/>` specifies the grid file for single-domain simulations.
- `<esmFsmTrace state="..."> />` is a flag that causes the ESM/FSM shared memory library to print verbose logging information for debugging purposes.
- `<partition name="...">...</partition>` is used to group other `flow2D3D` elements and give them separate names for use in the top-level `control` element (see Section 3.2.6). For example, there might be two Flow domains such as "water" and "fluid mud" or "overall" and "detailed". Both would use the same library, but have different mdf-files. See Appendix A.4 for a complete example.
- `<timeStep value="..."units="..."/>` overrides the simulation time step specified in the mdf-file (or files in the case of domain decomposition). The value is a floating-point number.

- `<waitFile pathname="..."/>` specifies the pathname of a file that must be present before Flow2D3D will start execution.

The order of the elements has no significance.

3.2.2 The `wave` Element *(To be added later)*

This element configures the surface wave (SWAN) component of DeltaresHydro.
Sub-elements are:

- `<library pathname="..."/>` specifies the path to the Wave shared library (Linux), dynamic library (Mac OS/X) or DLL (Windows).
- `<mdwFile pathname="..."/>` specifies the master definition wave file.
- `<partition name="...">...</partition>` is used to group other `wave` elements. See Section 3.2.6.

3.2.3 The `waq` Element *(To be added later)*

This element configures the water quality (WAQ) component of DeltaresHydro. Sub-elements are:

- `<library pathname="..."/>` specifies the path to the WAQ shared library (Linux), dynamic library (Mac OS/X) or DLL (Windows). This element is mandatory.
- `<mdqFile pathname="..."/>` specifies the master definition water quality file.
- `<partition name="...">...</partition>` is used to group other `waq` elements. See Section 3.2.6.

3.2.4 The `mesh` Element

This element defines one or more computational meshes for use by the flow, wave or waq components. It has two sub-elements:

- `<partition name="..."pathname="..."/>` defines a mesh and specifies the path to the mesh (or grid) file. This element is mandatory.
- `<couple name="...">...</couple>` defines a coupling edge between two meshes for domain decomposition. This element is mandatory.

The `mesh` element must have at least one `partition` element.

The `couple` element must have exactly two boundary sub-elements that specify the partitions and the indicies of grid cells that are coupled.:

```
<couple name="...">
   <boundary partition="...">
      m11 m12 n11 n12
   </boundary>
   <boundary partition="...">
      m21 m22 n21 n22
   </boundary>
</couple>
```

See Appendix A.2 for a real-world example.

3.2.5 The `delftOnline` Element

This element configures the DelftOnline (DOL) module that provides access to internal data structures of other modules.[2]

```
<delftOnline state="...">
   <!-- sub-elements -->
```

[2] At present only Flow2D3D utilizes DelftOnline.

```
</delftOnline>
```

DOL can be turned on or off using the `state` attribute. The following sub-elements are supported:

- `<urlFile pathname="...">` specifies the pathname of a file to which the DOL URL will be written. DOL clients need this URL to connect to the simulation.
- `<tcpPort value="...">` specifies the TCP/IP port the DOL server will listen on for client connections. The port must be available on the local host. More practical is to specify a range of possible ports with `<tcpPort range="...">`. The first free port encountered will be used. If this element if not specified the default range 51001:51099 will be used.
- `<waitOnStart state="...">` controls whether or not the module(s) that use DOL will immediately start its outer timestep loop, or will wait for a client to connect and give and explicit command to start (or step). The default if this element is not specified is false.
- `<clientControl state="...">` controls whether or not clients start, stop or step the simulation. The default if this element is not specified is false.
- `<clientWrite state="...">` controls whether or not clients may modify internal data. When publishing a DataElement the server indicates when read and/or write access is allowed. Actual right are determined by the logical AND of the server-side settings and this global run-specific setting. The default if this element is not specified is true.
- `<logging state="...">` controls how information DOL write to the logging facility. Possible values are: `silent` (never), `error` (only error messages), `info` (also informational messages) or `trace` (everything necessary for debugging). The default is `error`.

3.2.6 The `control` Element

This top-level element is used to control the execution of complex simulations with interacting components. Unlike the rest of the DeltaresHydro configuration file, the order of sub-elements in this section is significant, and it is the only section in which order is significant.

The `control` element is optional, but if it is omitted exactly one of the major component top-level elements — `flow2D3D`, `wave` or `waq` — must be present and this element may not have more than one `partition` sub-element. In such a case, DeltaresHydro knows exactly what to do.

The most important element within `control` is `<start name="..."/>`. The name attribute refers to a major component. If that component has multiple partitions, the partition name is specified following a colon, e.g., `<start name="flow2D3D:mud"/>`.

An initial start means loading and running a major component and allowing it reach some coordination point (CP). A subsequent `start` element would trigger the component (partition) to advance to the next CP.

Things can be started simultaneously within a `<parallel>...</parallel>` element.

To be continued...

3.3 Miscellaneous Elements

A DeltaresHydro configuration file may contain elements not described here, or specified in an XSD file. This allow development and experimentation, without interfering with production versions.

The following miscellaneous elements are recognized by DeltaresHydro.

3.3.1 The `documentation` Element

This element can be used to describe something, and may be used within any element. Unlike XML comments, the contents of a `documentation` element is visible to DeltaresHydro.[3]

3.3.2 The `crashOnAbort` Element

The `<crashOnAbort state="..."/>` element controls whether a core dump is produced when an intentional abort is done.[4] As a top-level element it controls all of DeltaresHydro. If present within a top-level element (or domain) it only controls that module.

4 The XMLTree Library

Yet to be documented...

[3] At present it's not clear how the `documentation` element will or should be used.

[4] Core dumps — a snapshot of the memory of a process for debugging purposes — are not presently supported in the Microsoft Windows versions of DeltaresHydro.

A Example Configuration Files

A.1 Example: Simple

This example is to be defined on short term; it will be used immediately for Delft3D-FLOW. All elements included will be used immediately.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<deltaresHydro
   xmlns="http://deltares.nl/deltaresHydro"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://deltares.nl/deltaresHydro
                       http://schemas.deltares.nl/deltaresHydro/0.05.xsd">

   <crashOnAbort state="enabled"/>

   <delftOnline state="yes">
      <urlFile pathname="/p/myproject/myrundir/flow2d3d.url"/>
      <tcpPort range="51001:51099"/>
      <waitOnStart state="false"/>
      <clientControl state="false"/> <!-- start, step, stop by client -->
      <clientWrite state="true"/> <!-- allowed modify data -->
      <logging state="error"/> <!-- silent, error, info, trace -->
      <crashOnAbort state="disabled"/>
   </delftOnline>

   <flow2D3D>
      <library pathname="/opt/delft3d/lnx64/flow2d3d/bin/libflow2d3d.so"/>
      <documentation>
         Basic tutorial testcase.
      </documentation>
      <mdfFile pathname="f34.mdf"/> <!-- single domain calculation -->
      <!-- ddbFile pathname="vlissingen.ddb"/--> <!-- domain decomposition calculation -->
      <waitFile pathname="debug.txt"/> <!-- needed for debugging parallel runs -->
      <esmFsm state="trace"/> <!-- silent, trace -->
   </flow2D3D>
</deltaresHydro>
```

A.2 Example: Integrated ddb

When using DomainDecomposition, the ddb-file is/was optionally in XML-format. Merging this into the config file is obvious. The user will have the advantage of controllong the distribution of the processess on a cluster.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<deltaresHydro
   xmlns="http://deltares.nl/deltaresHydro"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://deltares.nl/deltaresHydro
                       http://schemas.deltares.nl/deltaresHydro/0.05.xsd">

   <mesh>
      <partition name="overall" pathname="overall.bnd"/>
      <partition name="bay" pathname="bay.bnd"/>
      <partition name="harbor" pathname="harbor.bnd"/>

      <couple name="north">
         <boundary partition="overall">
            22 11 22 31
         </boundary>
         <boundary partition="bay">
            1 1 1 21
         </boundary>
      </couple>
      <couple name="west">
         <boundary partition="overall">
            20 1 20 21
         </boundary>
         <boundary partition="bay">
            1 31 1 51
         </boundary>
      </couple>
      <couple name="toHarbor">
         <boundary partition="bay">
            20 1 20 11
         </boundary>
         <boundary partition="harbor">
            1 1 1 11
         </boundary>
      </couple>
   </mesh>

   <flow2D3D>
      <library pathname="/opt/delft3d/lnx64/flow2d3d/bin/libflow2d3d.so"/>
      <partition name="overall">
         <mdfFile pathname="overall.mdf"/>
      </partition>
      <partition name="bay">
         <mdfFile pathname="bay"/>
      </partition>
      <partition name="harbor">
         <mdfFile pathname="harbor.mdf"/>
      </partition>
   </flow2D3D>
</deltaresHydro>
```

A.3 Example: Integrated parallel

When running parallel, the domain is automatically split. For optimization, overruling this automatic process may be needed.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<deltaresHydro
   xmlns="http://deltares.nl/deltaresHydro"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://deltares.nl/deltaresHydro
                       http://schemas.deltares.nl/deltaresHydro/0.05.xsd">

   <flow2D3D>
      <library pathname="/opt/delft3d/lnx64/flow2d3d/bin/libflow2d3d.so"/>
      <mdfFile pathname="f34.mdf"/>
      <split direction="n">
         23 31 41 69 101 <!-- six partitions -->
      </split>
   </flow2D3D>
</deltaresHydro>
```

A.4 Example: Control

This example shows how "d_hydro.exe" may handle complex run configurations. Main ideas:

- Each kernel is wrapped in a dll/so and reads the kernel specific information in the XML-config file
- "d_hydro.exe" reads the (unique) control block that defines what kernel(s) have to be started in what order.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<deltaresHydro
   xmlns="http://deltares.nl/deltaresHydro"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://deltares.nl/deltaresHydro
                       http://schemas.deltares.nl/deltaresHydro/0.05.xsd">

   <control>
      <documentation>
         Start a fluid mud calculation (FLOW for water phase, FLOW for mud phase)
         with WAVE online. When finished, continue with a WAQ calculation.
      </documentation>
      <sequential>
         <parallel>
            <start name="flow2D3D:water"/>
            <start name="flow2D3D:mud"/>
            <start name="wave"/>
         </parallel>
         <start name="waq"/>
      </sequential>
   </control>

   <flow2D3D>
      <library pathname="/opt/delft3d/lnx64/flow2d3d/bin/libflow2d3d.so"/>
      <timeStep value="5.0" units="minutes"/>
      <partition name="water">
         <mdfFile pathname="sed.mdf"/>
         <waitFile pathname="debug.txt"/>
      </partition>
      <partition name="mud">
         <mdfFile pathname="mud.mdf"/>
      </partition>
   </flow2D3D>

   <wave>
      <library pathname="/opt/delft3d/lnx64/wave/bin/libwave.so"/>
      <mdwFile pathname="01.mdw"/>
   </wave>

   <waq>
      <library pathname="/opt/delft3d/lnx64/waq/bin/libwaq.so"/>
      <mdqFile pathname="01.mdq"/>
   </waq>
</deltaresHydro>
```

Deltares, 2016. “BIBTEX key with no entry, needed if no citations are made in the document.”