Developing a FEWS (Compliant) Adapter

This is an overview of some of the requirements for developing a Delft-FEWS compliant adapter. This list is by no means exhaustive so before starting to develop an adapter please contact fews.support@deltares.nl

Since 2014.01 it is possible to write a NetCDF-CF based adapter, see Developing a FEWS Model Adapter (NetCDF-CF)

Development of FEWS Adapter

The model adapter as described in this manual provides the interface between a so-called model and the Delft-FEWS system. It enables FEWS to run such a model, thus providing the essential forecasting functionality. For each particular FEWS application applying a model, however, some aspects of this system have to be configured by the user in order for the system to work correctly. To achieve this, it is useful that the user has at least some basic understanding of the relation between Delft-FEWS, the model adapter and the forecasting model. In this section, a brief overview of this relation is provided. For more information, the user is referred to the FEWS manual: General Adapter.

Delft-FEWS and the general adapter

A key feature of DELFT-FEWS is its ability to run external modules to provide essential forecasting functionality. The General Adapter is the part of the DELFT-FEWS system that implements this feature. It is responsible for the data exchange with these modules and for executing the modules and their adapters. The Delft3D model adapter is an example of such a module run from the General Adapter (see also FEWS manual: General Adapter).

In order to develop a model adpater for a FEWS application, it is important to have a clear understanding of the relation between the General Adapter and a model adapter. This section summarizes some of the functionalities included in the FEWS general adapter module, and their relation to a model adapter.

The schematic interaction between the general adapter and an external moduleis shown in the below figure.

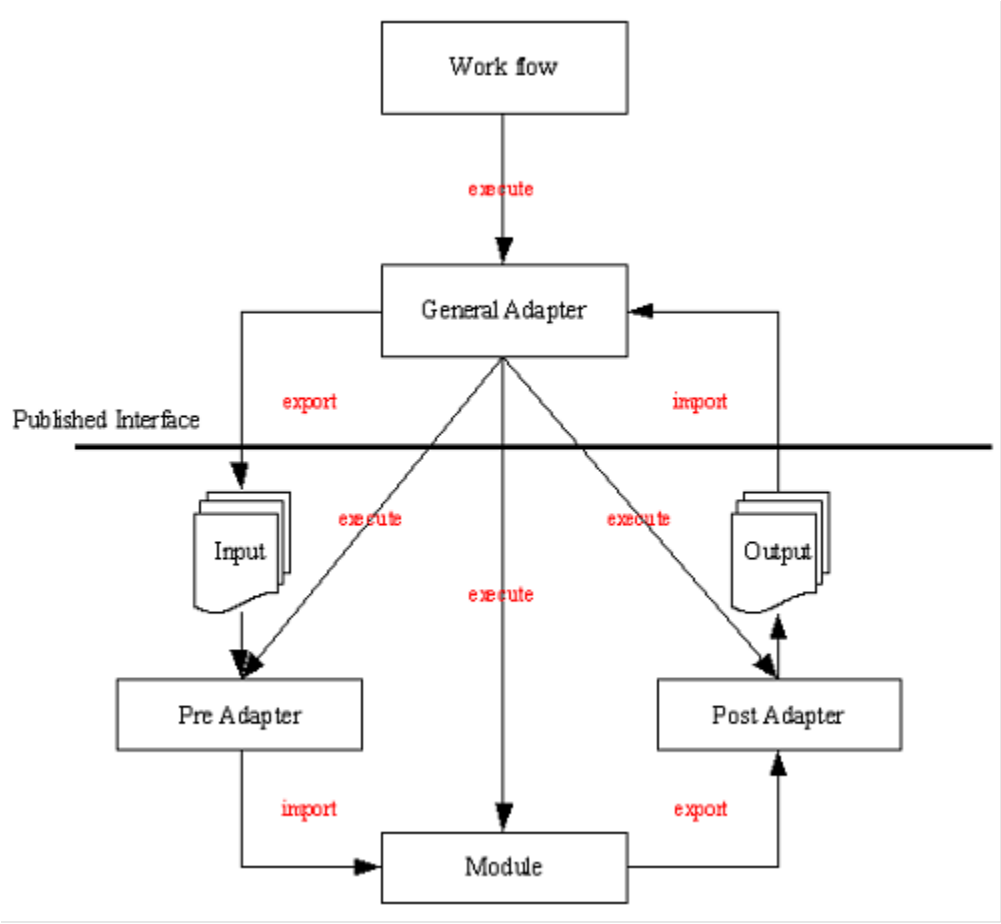

Figure 1: schematic interaction between the FEWS and an external module

This figure is understood in the following way:

1. The general adapter is that part of DELFT-FEWS which exports model input, imports model output data, and executes the pre-adapter, module and post-adapter.
2. Export and import of model data is done in datafiles following the Published Interface (PI) XML format.
3. The preAdapter, Module and postAdapter together form the model adapter. The model adapter is initiated from the general adapter.
4. The preAdapter is that part of the model adapter which converts input data in PI XML format to native model input data.
5. The postAdapter is that part of the model adapter which converts native model output data to PI XML format to be imported by FEWS.
6. The Module is that part of the model adapter which starts a model simulation.

Essential in this division of tasks between model adapter and general adapter is that from the vantage point of the general adapter the model adapter is a black box, and visa-versa. Exchange of information between these components is done based on exchange of PI XML data files; DELFT-FEWS does not have any knowledge about the modelling system (Delft3D in this case), whereas the modelling system does not have any knowledge about DELFT-FEWS. Translation of data from one component to the other is done using the model adapter. Thus model adapter is not a part of the Delft-FEWS system itself, but is essentially an external program initiated by DELFT-FEWS through the general adapter. In other words, for Delft-FEWS system a model adapter(s) + model is a model itself which read PI XML files as input and writes PI XML files as output.

This external program (the model adapter) should provide all the functionalities to, i) convert specific PI XML data to specific native model input files and model state ii) initiate a model simulation and iii) convert model output data and end state to PI XML data readable by DELFT-FEWS. In addition, this model adapter has the following tasks, iv) logging of errors during all steps by the model adapter and by the model itself and v) administation of the model state. For both these tasks pre-defined PI XML file formats exists readable by DELFT-FEWS.

The model adapter should be as far as possible made configurable and must be done in a consistent way.

Log messages and diagnostics

External Modules and their adapters can be written in any programming language. Modules and module adapters can only communicate with the General Adapter via the Published Interface. The only 2 means to exchange information with the General Adapter are:

- A diagnostic file written in the Published Interface format. If such a diagnostic file is available the General Adapter will read it and write corresponding logs to the FEWS system.
- The return code indicating the result of an execution.

return code	meaning

non Zero	graceful failure - read message
0	successful execution

Modules and their adapters cannot use exception or event handling as a means to communicate with the General Adapter.

Return code error information
The GA distinguishes between two types of failures of a module or module adaptor, *graceful* and *non graceful*.

- On graceful failure the GA expects a non-zero return code. The type of error and the accompanied message are stored in the diagnostics file according to the GA.
- On non graceful failure the stack trace will be written for adapter (applies to adapters written within FEWS only) otherwise generic message will be given that adapter has crashed.
- On successful execution the GA expects a zero return code.

Actions taken by the GA on:

- a *non graceful* failure or on a *graceful failure that indicates a fatal error* (see the definition of the diagnostics file below)
  - the GA will make a binary dump of the entire module configuration (a list of directories supplied by the module adaptor constructor).
  - The GA will stop processing the remaining instructions.
- all other occasions
  - the messages in the diagnostics file are passed on to the event logging system
  - further operation is continued

Diagnostics file
The published interface documentation describes the (simple) xml file format. Per batch (pre processing - module run - post processing) one diagnostics file is always expected. Each line/message in this file contains the actual text and the warning level attached to this text. The warning levels in the diagnostics file will be interpreted by the general adapter. according to the following table:

Level	Name	Description	Example
4	debug	debug logging	version: 1.0
3	info	information, all is well	Module PreProcessor : program ended
2	warning	warning information	Module Processor : unresolved symbol
1	error	critical problems	Module Processor: module fails (returns 1)
0	fatal	fatal error, complete module crash	Module Processor: *division by zero*

All levels higher than 3 are regarded as non-essential (debug) information. The warnings are recorded in the system, but no actions will be taken.

In memory file transfer
The Java adapters should check if a file is transferred in memory to the adapter by FEWS.

```
public static BufferedInputStream
newInputStream(File file) throws
IOException {
    //noinspection
UseOfPropertiesAsHashtable
    var stream = System.getProperties().
get(file);
    if (stream instanceof
BufferedInputStream) return (
BufferedInputStream) stream;
    return new BufferedInputStream(new
FileInputStream(file));
}

public static BufferedOutputStream
newOutputStream(File file) throws
IOException {
    //noinspection
UseOfPropertiesAsHashtable
    var stream = System.getProperties().
get(file);
    if (stream instanceof
BufferedOutputStream) return (
BufferedOutputStream) stream;
    return new BufferedOutputStream(new
FileOutputStream(file));
}
```

Validating xml

When an adapter reads xml written by FEWS according to its own xsd there is no need for validation, it is an unnecessary step because in the creation process FEWS uses those xsd's and therefore the xml will be valid.

Since FEWS 2022.02 access to external xsd's from other xsd's is closed off for the Java xml package which is included in the jdk. This is done by setting the javax.xml.accessExternalSchema sytem property to an empty string for security reasons.

When existing adapters do use this and it is not an option to update the adapter, since FEWS stable2022.02 patch #118458 in clientConfig.xml the javax.xml.accessExternalSchema system property can be overruled via

<jvmOption>-Djavax.xml.accessExternalSchema=all</jvmOption>

- Updating PI State XML file within Pre and Post Model Adapters

Example model adapter (includes source code)

Download the example adapter as a quick start to develop your own adapter.

FEWS_Example_Model_Adapter..zip