Documentation for the Module Editor of the FRAMEwork System (FRAMES)

PNWD-3506

With Contributions From

B. L. Hoopes
M. A. Pelton
K. J. Castleton

Documentation for the Module Editor of the FRAMEwork System (FRAMES)

LEGAL NOTICE

This report was prepared by Battelle Memorial Institute (Battelle) as an account of sponsored research activities. Neither Client nor Battelle nor any person acting on behalf of either:

MAKES ANY WARRANTY OR REPRESENTATION, EXPRESS OR IMPLIED, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, process, or composition disclosed in this report may not infringe privately owned rights; or

Assumes any liabilities with respect to the use of, or for damages resulting from the use of, any information, apparatus, process, or composition disclosed in this report.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Battelle. The views and opinions of authors expressed herein do not necessarily state or reflect those of Battelle.

Documentation for the Module Editor of the FRAMEwork System (FRAMES)

Title Page
Legal Notice
Table of Contents
Introduction
Requirements
Design
Quality Assurance and Testing

Documentation for the Module Editor of the FRAMEwork System (FRAMES)

Introduction

This documentation provides information on a component of Version 2.x of the FRAMEwork System (FRAMES), which is a software platform that allows for the linking of various modules into complete assessment systems (Whelan et al. 1997 PNNL-11748). Documentation includes requirements, design and specifications or formulations, and quality assurance and testing.

Portions of this documentation may have been previously issued in reports from the Pacific Northwest National Laboratory (PNNL), operated by Battelle for the U.S. Department of Energy. All PNNL reports are issued a tracking number. Numbers on the title page of this documentation indicate these previous reports.

This documentation can be used by software engineers and testers to ensure that each component functions properly. The information can also be used by analysts and managers to better understand the component's use within FRAMES.

Documentation for the Module Editor of the FRAMEwork System (FRAMES)

Requirements for the FRAMES Module Editor

Developing modules in the FRAMEwork System (FRAMES) 2.0 is a two-part process of creating a module's executables according to FRAMES guidelines and describing the capabilities of a module to FRAMES 2.0. The Framework Development Environment is focused on the latter part of developing modules. This section describes the requirements for the former part of the process.

The FRAMES Module Editor has the following requirements:

1. Provide a module (MOD) file interface for creating, editing, and deleting module files
2. Allow the user to enter in module description information categorized by executable, reference, company, developer, requirement, and connection schemes
3. Allow the user to specify a module executable or a URL where the module is located.

Documentation for the Module Editor of the FRAMEwork System (FRAMES)

Design of the FRAMES Module Editor

The FRAMES Module Editor is designed to function through the FRAMES user interface. Modules are recognized by FRAMES through a file that describes the module according to a certain format. All information about a module is stored in an ASCII file with a .MOD file extension. When the Module Editor changes information about a module, it is actually manipulating the module's .MOD file.

Note that in previous versions of FRAMES, a module's description file was saved with a .DES file extension. The .DES file had a special syntax to follow to create a module's description file. However, modules in FRAMES 2.0 use a different file format.

Like all other data in FRAMES 2.0, module data are stored in a dataset. Although the previous module description files (.DES files) contained much of the same information as the new description files (.MOD files), the new files have a completely different format from the old files. Therefore, .DES files and .MOD files are incompatible with each other.

This section describes the file format in more detail as well as how the design of the Module Editor accomodates connecting modules within the system, adding and deleting a module from FRAMES, and making modules available via an online server.

Formatting a Module Description File

A module's description file (.MOD file) describes a module's capability to work within FRAMES 2.0. The file contains a variety of information fields categorized by executable information, reference information, contact information, online information, requirement information, and connection-scheme information.

Executable information includes the following:

- Name - name of the module
- Class - class (system, database, viewer, or model) to which it belongs
- Version - version of the software
- Icon - module's icon to be displayed within FRAMES 2.0
- UI executable - path of the application for the user interface
- UI arguments - number of user interface executable command-line arguments
- Executable - path of the application of the model
- Executable arguments - number of executable command-line arguments.

If the module is an online module, then three additional fields are required:

- URL address - where the module is located online
- Login name - name to log in to the online module's site
- Password - password to log in to the online module's site.

Reference information includes the following:

- Description - short description of what the module does
- Supporting document references - citations for reference documents.

Company information includes the following:

- Company name - name of the company responsible for the module
- Company's point of contact information - name, phone number, and e-mail address of the person responsible for the module within the company
- Company address - physical location of the company
- URL address - address of the company's web site.

Developer information includes the following:

- Contact name - name of the developer
- Telephone and fax numbers - contact information of the developer
- E-mail - e-mail address of the developer.

Requirement information includes the following:

- Operating system - operating systems under which the module will run
- Processor - minimum processing speed required for the module
- RAM - minimum required memory
- Disk space - minimum required disk space to install and run the module.

Connecting a Module in the System

To connect a module into FRAMES requires the user to add a User Interface Dictionary and create a scheme of produced and consumed datasets.

User Interface Dictionary

The UI Dictionary is used solely by a module for consumption and production. It is categorized as a model input dictionary, private to a particular module. This is in contrast to scheme dictionaries (see below), which can only use boundary-condition dictionaries that are accessible to all modules for consumption and production.

To add a UI Dictionary to FRAMES, the user opens the Module Editor and selects a module's connection scheme from an available tree. Clicking on the "UI Dictionary" button opens a form with available module input dictionaries. The user selects a dictionary and clicks the "Add Dictionary" button. The UI Dictionary will appear in the text field near the "UI Dictionary" button.

Schemes

In FRAMES 2.0, modules define their data dependencies and capabilities by creating schemes. A scheme describes which dictionaries the module's executable needs to run and which dictionaries the executable will then produce. Because dictionaries are just meta-data, schemes actually specify which type of dataset a module expects to consume or produce.

A module can have multiple schemes. Each scheme informs the system of a specific set of connections the executable is capable of handling. If Module A attempts to connect to Module B's produced dictionaries and does not have a connection scheme that includes Module B, Module A will not run.

To add a scheme, the user opens the Module Editor and selects connection-scheme information about the module in the display. Clicking on the "Add Scheme" button opens an input box, where the user can provide the name for the new scheme. After confirmation, the Module Editor adds the new scheme to the available list.

To delete a scheme, the user opens the Module Editor and selects connection-scheme information about the module in the display. The user then highlights the scheme to be deleted and clicks on the "Delete Scheme" button. After confirmation, the Module Editor removes the scheme from the available list.

Consuming Dictionaries

According to its scheme, a module can consume or produce boundary-condition dictionaries from or for other modules, respectively. Modules may also consume UI Dictionaries specific to that module. To add a consuming dictionary, the user opens the Module Editor and selects a scheme for the module of interest. The selection causes four lists to appear. The upper two are for consuming dictionaries and the lower two are for producing dictionaries. The upper and lower lists on the left contain all available dictionaries the scheme can add. The upper and lower lists on the right contain all the dictionaries already added to the scheme.

The user chooses a consuming dictionary from the available list and clicks on the ">>" button between the two upper lists. This adds a new consuming dictionary to the list in FRAMES.

To delete a consuming dictionary, the user opens the Module Editor and selects a scheme for the module of interest. The selection causes the same four lists to appear as noted above. The user chooses a consuming dictionary from the list of those already added to the scheme and clicks on the "<<" button between the two upper lists. This deletes the consuming dictionary.

Producing Dictionaries

A module may produce a boundary-condition dictionary for use by other modules. To add a producing dictionary, the user opens the Module Editor and selects a scheme for the module of interest. The selection causes the same four lists to appear as mentioned in the previous section. The user chooses a dictionary from the list of available producing dictionaries and clicks the ">>" button between the two lower lists. This adds a new producing dictionary to the list in FRAMES.

To delete a consuming dictionary, the user opens the Module Editor and selects a scheme for the module of interest. The selection causes the same four lists to appear as in the previous section. The user chooses a dictionary from the list of producing dictionaries already in the system and clicks the "<<" button between the two lower lists. This removes the producing dictionary from the list of those available.

Adding a Module

There are two methods for adding modules into the FDE, the main interface of FRAMES. The first method creates an entirely new module; the second method opens an existing module that has not yet been added to the list in FRAMES. In both cases, the FDE uses system files to store the module information.

To create a new module, the user selects the class of module, choosing from "System," "Database," "Viewer," or "Model," then clicks a button called "New Module." When the Open File form appears, the user selects a location and enters a filename that does not already exist. After confirmation, a message box notifies the user that the file does not exist and asks whether the file should be created. On confirmation, the Module Editor includes the new module in the selection list.

To open an existing module, the user opens the Dictionary Editor and selects a class of dictionary ("System," "Database," "Viewer," or "Model"), then clicks a button called "Open Module." When the Open File form appears, the user selects the module of choice and opens it. This action adds the module to the selection list across editors.

Deleting a Module

To delete a module from FRAMES, the user opens the Module Editor and selects a class of module ("System," "Database," "Viewer," or "Model"), then highlights a module from the available list. The user clicks on a button called "Delete Module." On confirmation, the module will be deleted, unless a domain uses the module in its palette. Domains are discussed in more detail in the Domain Editor documentation. If the module was deleted successfully, it will be removed from the selection list.

Publishing a Module on the Linkage Server

The Linkage Server is an online database for publishing and retrieving FRAMES modules and dictionaries. To make a module available in this manner, all files associated with the module, including the .MOD file and all dictionaries used in its connection scheme must be included on the server. If a newer version of a file exists on the server, the Linkage Server asks the user before overwriting the file.

To publish a module to the Linkage Server, the user opens the Module Editor and selects a module. The user then clicks the "Publish Online" button. FRAMES automatically publishes all related files to the Linkage Server.

To download a module from the Linkage Server, the user selects the Linkage Server from the Tools menu, which opens a form listing Linkage Server updates with modules and dictionaries. Dictionaries associated with modules are listed beneath the module. Icons associated with a dictionary or module in the Linkage Server Updates form define the relationship between the files located on the computer that houses FRAMES and the files in the Linkage Server database.

To retrieve a module from the Linkage Server, the user selects a module from the Linkage Server Updates form and clicks the "Update" button. This action updates the module file along with all the associated dictionary files.

Quality Assurance and Testing of the FRAMES Module Editor

The FRAMES Module Editor was developed under a quality assurance (QA) program that looked at the software life cycle: requirements analysis, design, programming, modification, testing, and implementation. Part of the QA program involves testing each component to ensure that it satisfies its requirements. The requirements section of this documentation provides a list of requirements for the FRAMES Module Editor. A test plan was developed with a test case that addressed these requirements. An overview of quality assurance and testing for FRAMES can be found in the system documentation.

An Approach to Ensuring Quality in Environmental Software (Gelston et al. 1998. PNNL-11880)
Test Plan for the Framework Development Environment
Status Report for the Framework Development Environment
- Test bed for the Framework Development Environment.
