Run-Time I/O Layer: Requirements and Design

MPAS Development Team

December 6, 2013

Contents

Chapter 1

Summary

This document describes the requirements and design for creating a run-time configurable I/O layer. This I/O layer will be used to add and remove fields from I/O streams at run-time rather than at build time.

Before beginning the discussion of requirements, a concept of I/O streams needs to be introduced. An I/O stream is fully described by the following information:

- A set of variables from the model
- A frequency describing how often the stream should be used
- A templated filename
- A direction (Input / Output / Off)
- An identified
- A number of frames per file

Chapter 2

Requirements

This chapter lays out the requirements the run-time I/O layer.

2.1 Requirement: Variables can be added/removed from I/O streams at run-time

Date last modified: 02/15/13
Contributors: (Doug Jacobsen, Michael Duda)

The run-time configurable I/O layer should provide the ability for modification of I/O streams at run time, as opposed to the current structure where it's at compile time.

This includes any persistent variable defined in registry.

2.2 Requirement: Ability to create new streams

Date last modified: 02/15/13
Contributors: (Doug Jacobsen, Michael Duda)

The new I/O layer should make the creation of a new stream available to a developer/user. Although it would be preferred to have stream creation possible at run-time, it is not specified when during the build/run time streams need to be defined. Simply that they need to be able to be defined at some point.

Presently there are only four streams allowed by a user, and these streams have to be modified at build time.

2.3 Requirement: Package Streams

Date last modified: 12/03/2013
Contributors: (Doug Jacobsen, Michael Duda)

A stream must be allowed to be attached to a package. Meaning, if the package is disabled at runtime, the stream will be completely disabled at runtime as well.

2.4 Requirement: Required Streams

Date last modified: 12/03/2013
Contributors: (Doug Jacobsen, Michael Duda)

There are three streams that will be defined by default. These are the mesh, initial condition, and the restart streams. Core developers should be able to fully describe these, as these are streams that are required to make a core run successfully.

These streams are not allowed to be modified at run-time.

2.5 Requirement: Default streams defined at build time

Date last modified: 12/03/2013
Contributors: (Doug Jacobsen, Michael Duda)

A core must be able to define a default set of streams. These will additionally specify the default configuration for the run-time I/O file.

Mesh, initial condition, and restart streams must also be defined at build time.

2.6 Requirement: Flexible stream definitions

Date last modified: 12/03/2013
Contributors: (Doug Jacobsen, Michael Duda)

Streams must be allowed to be defined by the union of other persistent constructs within a core's registry file.

These include:

- Variables (Individual variables, and var_array constituents)
- Variable arrays (Including all of their constituents)
- Variable structures (Including all of their variables and var_arrays)
- Packages (Including all variables attached to a given package)
- Streams (Including all variables included in another stream)

Some restrictions are that a stream will not include streams defined within another stream. This restriction is to prevent circular dependence problems.

2.7 Requirement: Ability to write output streams during initialization

Date last modified: 12/03/2013
Contributors: (Doug Jacobsen, Michael Duda)

The I/O layer must provide cores the ability to write any/all output streams at the initial time.

2.8 Requirement: Descriptive error checking

Date last modified: 12/04/2013
Contributors: (Doug Jacobsen, Michael Duda)

The I/O layer must provide users with descriptive error messages when configured inappropriately.

For example, the I/O layer must:

- Warn and continue on a variable in a stream not existing
- Error and fail when attempting to modify required streams (restart/mesh/initial condition)
- Error and fail when using output streams that share a filename template
- Warn when an output stream has no frequency, and write initially

2.9 Requirement: Well defined order for reading input streams

Date last modified: 12/04/2013
Contributors: (Doug Jacobsen, Michael Duda)

The I/O layer must define an order when reading input streams. For example, the first two streams read in a model run are the mesh stream, followed by the initial condition stream. After these have been read, the run-time defined input streams can be read in the order they are defined in the run-time configuration file.

This allows cores to anticipate which stream would overwrite another stream, assuming they share variables.

2.10 Requirement: Ability to read input streams only on initialization

Date last modified: 12/04/2013
Contributors: (Doug Jacobsen, Michael Duda)

Input streams must provide the ability to only be read upon initialization.

Chapter 3

Design and Implementation

3.1 Implementation: Configuration format

Date last modified: 02/15/13
Contributors: (Doug Jacobsen, Michael Duda)

The format for configuring run-time I/O options will be specified in XML format. This file will be merged with the namelist file to provide users a single file to configure all aspects of a model run.

```
<stream name="streamA" interval="00_00:00:00" frames="100" filename="infile.%Y.nc"
       direction="input">
       <var name="var1"/>
       <var name="var2"/>
       <var name="var3"/>
</stream>
```

Using this format, a stream encapsulates a list of variables that will be included as part of the stream when reading from or writing to the filename.

3.2 Implementation: Stream handler

Date last modified: 02/15/13
Contributors: (Doug Jacobsen, Michael Duda)

Although we already have the capability of creating I/O streams, an easy to use mpas_stream_handler.F module would be created to allow developers access to the created streams, and their associated alarms.

3.3 Implementation: Auto-documentation parser

Date last modified: 02/15/13
Contributors: (Doug Jacobsen, Michael Duda)

Developers will be provided with a parser (probably a python script) of the Registry input file, that will generate a default namelist, and documentation LaTeX files for use in a users guide.

3.4 Potential Implementation: Run-time addition of streams

Date last modified: 02/19/13
Contributors: (Doug Jacobsen, Michael Duda)

One possible idea for the run-time addition and removal of streams is to have a general IO namelist group in the I/O namelist file. This group would contain options to handle PIO parameters and general settings that apply to all streams. In addition to this, streams could be defined as follows:

```
&io
       config_io_streams = "input output restart newStream newStream2"
/
```

This config_io_streams string would be broken up, and each sub-stream (space delimited) would represent an independent stream. The namelist parser would then look for a group for each stream, and an option under each field group for each stream.

One potential idea to implement this is to store all streams in a linked list. When adding/creating streams, new streams are appended to this linked list. As fields are created, they are added to the appropriate streams.

By default, all streams are empty. Only fields that are explicitly part of a stream are added to a stream.

Chapter 4

Work list

Preparatory Work - Complete by Jan 1 2014

- Transition to universal mesh type
  - Required mesh definition and a mesh stream
- Distinct namelists for each core (Auto-generated at build time)
  - Auto generate default stream definitions
- Transition pre-defined streams to new stream definitions in Registry.xml
  - Using the new stream layout

Deal with namespace conflicts - Complete by April 1 2014
Overlaps heavily with run-time stream definitions. The implementation of run-time stream definitions might require back-tracking to cleanup for this purpose, so this is placed ahead of that.

- Collections
  - Groups
  - Vars
  - Namelist options
- Domain $->$ Core with additional information (like a core identifier)
- Rename mpas_core, mpas_core_run, mpas_core_init, mpas_core_finalize
- Driver update based on core renaming

- Build system needs to define core type (Function pointers and other related stuff)
- Module variables need to have a context (Might live in the core data structure)

Run-time configuration of default streams.

- Generate default run-time stream configuration for pre-defined streams at build time
- Run-time C-XML $->$ Fortran-DDT Translator

EITHER:

- Deal with "coupler" that handles I/O and communications for multiple cores
- Stream Manager module

Both would generate new I/O streams at run-time.
Revisit list of features, and prioritize. More intelligent intervals, start/stop times, logging, etc.

Continue with multiple cores.

Chapter 5

Testing

5.1 Testing and Validation: Run-time I/O Layer

Date last modified: 02/15/13
Contributors: (Doug Jacobsen, Michael Duda)