FMI Cross Check Implementation Notes:

How to Submit Test FMUs and Cross-Check Results

Revisions:

31.07.14	V3 of the Cross-Check Rules and Implementation Nodes accepted by the FMI Steering Committee
16.07.14	V3 with changes suggested to Steering Committee 17.07.2014 (AJ)
17.06.13	V2 with suggested changes (AJ)
25.04.13	Initial version (by Andreas Junghanns)

Contents

1. Intent of this Document

This document describes how the back end of the CrossCheck table generation works and where to place the required data elements, etc. If you plan to submit Test FMUs or CrossCheck Results then you should read this document.

2. Motivation and General Ideas

As approved by the FMI Steering Committee, the CrossCheck rules require the FMI community to produce example FMUs by tools exporting FMUs and test results by tools importing FMUs.

On the Modelica Association Server is an infrastructure in place now that implements the table generation from submissions into the appropriate directory structure.

The following document explains the mechanisms involved relevant to the community and what to submit and where and how, such that the result-compilation scripts will successfully pick them up.

2.2 3 pieces of information – in 2 branches of the repository

There are three branches containing different stages of the Cross-Check information:

1. **sandbox**: a private repository branch used only by FMI members for development version Cross-Check. This information is non-permanent: it can be changed and removed.
2. **public**: This is a place where permanent Cross-Check information is stored by the tool vendors. Results, once stored here, are not to be changed, or removed.

There are three web-sites to read those results:

1. https://sandbox.fmi-standard.org/: shows the results of the XC-tests and FMUs in the sandbox branch of the repository
2. https://stage.fmi-standard.org/: shows the results of the XC-tests and the FMUs in the public branch of the repository, but unfiltered!! This is not public but is intended to check immediately after submitting new data what tables are generated from this new data
3. https://fmi-standard.org/: shows the results of the XC-tests and the FMUs **after** inspection by the web-master

There are three main directories that contain information relevant to the scripts described here. Note, they exist for each branch:

1. https://svn.fmi-standard.org/fmi/branches/FMISite/[sandbox | stage | live]/tools:

   [[This will be changed on Jan 1, 2015: the tools info will be moved both branches public/ and sandbox/]]

   This directory contains tool description files to be submitted by tool vendors. They contain, among other information, the current name of the tool supporting FMI as well as which variant and links to more information. Example file `TotalFMIMaster.info`:

   ```
   [Tool]
   name=Total FMI Master
   href=http://www.company.do/Products/TotalFMIMaster.aspx
   slave_cs=
   master_cs=A
   import_me=A
   export_me=P
   import_me_20=A
   export_me_20=
   ```

```
slave_cs_20=A
master_cs_20=A
note=FMI tool of doom <a href="http://www.company.do/">TotalFMIMaster</a>
```

This file specifies the capabilities currently available "`...=A`", planned "`...=P`" or not supported. A number of URLs can be specified to be linked in the tool overview table.
IMPORTANT: The name of the file shall not contain spaces or special characters, but may only contain any of: 0-9, a-z, A-Z, _ and '.'. This base of the file name (here `ToolFMIMaster`) will be used to match exported FMUs and test results.

2. https://svn.fmi-standard.org/fmi/branches/[sandbox | public]/Test_FMUs:
   Here test FMUs are submitted, sorted by (in hierarchical order, top down as directories):
   1. FMI variant (CoSimulation or ModelExchange),
   2. platform (Note this has been moved up to be consistent with CrossCheck Results)
   3. tool name (match the base of the info file name!),
   4. tool version (versions may only contain the same characters as tool names),
   5. specific FMU (one FMU per directory).
3. https://svn.fmi-standard.org/fmi/branches/[sandbox | public]/CrossCheck_Results:
   Here FMU test results are submitted, sorted by (in hierarchical order, top down as directories):
   1. FMI variant (CoSimulation or ModelExchange),
   2. platform,
   3. importing tool (match the base of the info file name!),
   4. importing tool version (versions may only contain the same characters as tool names),
   5. exporting tool (match the base of the info file name!),
   6. exporting tool version (versions may only contain the same characters as tool names),
   7. specific FMU (one FMU test per directory).

   The table generation scripts will search for one of three files: passed, rejected and failed (see CrossCheck Rules for their meaning). If none of these files can be found, the experiment is considered "failed".

2.3 Table generation

We generate the page tools.html in https://svn.fmi-standard.org/fmi/branches/FMISite/[sandbox | stage | live]/templates which contains an overview over all tools and their status as submitted by tool vendors.

For each FMI variant and platform combination that has results submitted, we generate an html file (into the same directory as above) listing all exporting tools and all importing tools in a table (almost) as specified by the CrossChecking specification.

Table generation is triggered whenever new FMUs or CrossCheck results are submitted to the repository.

2.4 Submitting Results

Diagnostic

Currently, the table generation will check a number of properties, like empty directories, missing directories, matching names for tools and search for illegal characters in tool names.

More will be added later.

Errors will stop table generation.

If diagnostic fails (errors found), the old tables will not be removed and no new tables will be generated. An indication of this is that the generation date, displayed at the top of each generated page, does not change. A list of errors and warnings can be found in https://[sandbox | stage | live].fmi-standard.org/diagnostic.

What to submit

The CrossCheck Rules state what to submit (that specification has to be clarified significantly).

We also require the commit of one of the following files “passed”, “rejected” xor “failed”. These files indicate the test result. If no such file can be found, the result is considered “failed”.

2.5 Submitting FMUs

Diagnostic

Currently, when you submit an FMU and supporting documentation, we check, among other things properties of the CSV files, existence of proper documentation. If this check fails, your submission will be rejected and diagnostic can be found here: https://impact.modelica.org/crosscheckscript.log

What to submit

Formally, see the CrossCheck Rules on what to submit. Informally, we ask you to consider how to ease automated testing for importers: please supply all necessary information, preferably in machine readable form in the _ref.opt file, or in the readme files.

2.6 Sandbox area for Cross-Check

There is a sandbox area where test FMUs and Cross-Check results of unreleased tool versions can be submitted and Cross-Check tested. Here it is possible to update and delete those submits.

Read and write-access is restricted to FMI Steering Committee Members and FMI Advisory Committee Members.

2.7 FAQ

The following contains answers to questions we get asked a lot.

1. Why are importer/exporter rules more strict then the other?
   Note: I do get both complains, which is the sign of a good compromise – both sides are unhappy. Note furthermore, that both tasks are completely asymmetrical by nature: While exporter produce the FMU once will importers import many different FMUs. While FMU exporter will never be “out of date”, importer results will be once new or updated FMUs are available. While FMU exporter submit first versions almost blindly (exposing their export bugs), importers and work on their algorithms and submit their results after fixing their import bugs.
2. Why do we have to quote (all) variable names in the CSV files?
   The fundamental problem here is that the FMI Standard allows variable names to contain all the usual separators as legal characters. E.g. it is not sufficient to allow commas ',' in the variable names and use semi-colon ';' as separator – because semi-colon are also allowed as characters. As to why ALL variables must be quoted: We (programmers) simply believe this to be the simplest way for both writing CSV and reading CSV files.

3. The “passed” condition for CrossCheck Results are too vague!
   We agree on “vague”. We disagree on “too vague”. We failed so far to find a mathematically computable criterion that would always makes sense. Tolerances – relative and/or absolute – are insufficient because we have too many exceptions where different solvers and event detection algorithms create slight shifts in X-direction that lead to arbitrarily large differences to the references. Human judgment is required to overrule any failed automatic detection that is deployed for almost all FMUs.
4. What happens if a new version of an old FMU is submitted?
   We add a small indication for the case when CrossCheck-Results are older than the corresponding FMU. (not implemented yet)
5. How should values in the CSV files be interpreted between communication time points?
   There was a proposal to use left and right limits to indicate discrete changes for discrete variables. Such a change would have allowed to linearly interpolate all variables between communication time points.
   However, since the information about a variable being continuous or discrete is also available in the modelDesription.xml, the environment knows where to apply interpolation and where not.
6. How should I encode the CSV files?
   CSV files should be UTF-8 encoded. That is required for backward compatibility.