FMI Cross Check:

How to Improve FMI Compliance

Revisions:

31.07.14	V3 of the Cross-Check Rules approved by Steering Committee
26.07.14	Draft V3, prepared for voting on July 31, 2014 (Andreas Junghanns)
26.06.13	Draft for Version 2 of the FMI Cross-Check Rules (for voting) (Andreas Junghanns) – approved by Steering Committee
12.01.13	Clarified Rule 12 after FMI design meeting 11.01.13
29.11.13	First consolidated version by Andreas Junghanns, Jakob Mauss, Torsten Blochwitz, Kilian Link, Martin Otter, Hubertus Tummescheit (edited by Andreas Junghanns).
20.11.12	Initial version (by Andreas Junghanns, based on Jakob Mauss' first email and Hubertus Tummescheit's comments)

Contents

1. Motivation and General Ideas

1.2 The State of FMI Compliance Q4 2012

We hear FMI users complaining, and our own experience confirms, that exchange of models via FMI does not work as stable as expected. For example, import of FMU generated by tool A into tool B fails, due to unexpected signature of functions found in DLL, unexpected directory structure of the zip file, or syntactically incorrect XML. We see access violations and other exceptions during simulation, FMU code that does not even compile during export, etc. Such failures damage the reputation of the FMI.

1.3 Goals of the Initial Proposal

This following proposal aims to:

- improve and document the quality of FMI-based tool chain
- help vendors to detect problems with their FMI support
- detect and fix problems with FMI specification
- improve reputation of the FMI
- clarify the rules for how tool-related information is published on the fmi-standard.org pages

1.4 The State of FMI Compliance and Cross-Check Q2 2013

The Cross-Check (XC) infrastructure is in place, multiple tools have submitted test FMUs and another set of tools has submitted test results. While working on the implementation of the XC infrastructure, exporting the test FMUs and reference signal traces and especially when running import checks multiple inconsistencies, omissions and ambiguities have been detected.

This update of the XC Rules aims to remove all such problems.

So far, FMI compliance has improved significantly, but much is left to improve. Furthermore, for FMI standard 2.0, this infrastructure can be used to detect problems earlier than for version 1.0 as the XC infrastructure is already in place.

If you want to submit Test FMUs and/or Cross-Check Results then please read the Implementation Notes for FMI Cross-Check.

1.5 The State of FMI Compliance and Cross-Check Q2 2014

During the last months the FMI community has been busy finalizing Version 2.0 of the FMI Standard. In parallel, the Cross-Check infrastructure was updated to handle Version 2.0.

The increased number of test FMUs available make automated Cross-Check computations for each importing company highly desirable. Version 2 of the Cross-Check Rules left a number of submission details unclear leading to difficulties in building Cross-Check scripts. Version 3 is clarifying these unclear points.

Furthermore, a number of changes have been made to change the criteria for the “Available” status.

1.6 FMI Certification versus FMI Cross Check

We previously discussed the idea to establish an 'FMI certification' procedure, that a tool must pass to get listed on the FMI tools page. However, implementing this is a lot of work, and the FMI Project has no resources to test tools. Furthermore, the FMI Project consists of tool vendors and is not in a position to independently and objectively run certification tests.

The rules of the FMI Cross Check are defined such that evaluation work is mostly on the side of the participating tool vendors, not on the FMI Project side. Test documentation is required to allow tool customers to verify claims made on the web pages.

Work for FMI Project reduces to:

- elaborating and incrementally improving the rules of the FMI Cross Check
- inviting tool vendors to participate
- collecting and publishing results on the FMI web pages
- arbitrating in case of conflicting test result interpretations

2. Restructuring of the Tools Page at fmi-standard.org

The Tools page of the FMI web was restructured (www.fmi-standard.org/tools):

There are 4 entries in the cells of the tools page:

1. Empty: means this tool will not support this variant of the standard
2. "**Planned**" in gray: at some point in the future, this tool will support this variant of the FMI standard. The description field might be more explicit as to what release of the tool and when this can be expected.
3. "Available" in yellow: The tool vendor indicates that he supports this variant of the FMI standard, but has not (yet) submitted supporting documentation that shows the cross-check with other tools.
4. "Available" in dark green for an "**Export** Column". The meaning is that the tool vendor provides sufficient information to show that FMUs can be exported that comply with the standard. Details can be found in the rules section. When clicking on the button, the web page with the svn entry of the respective tool is shown where the exported FMUs are present.

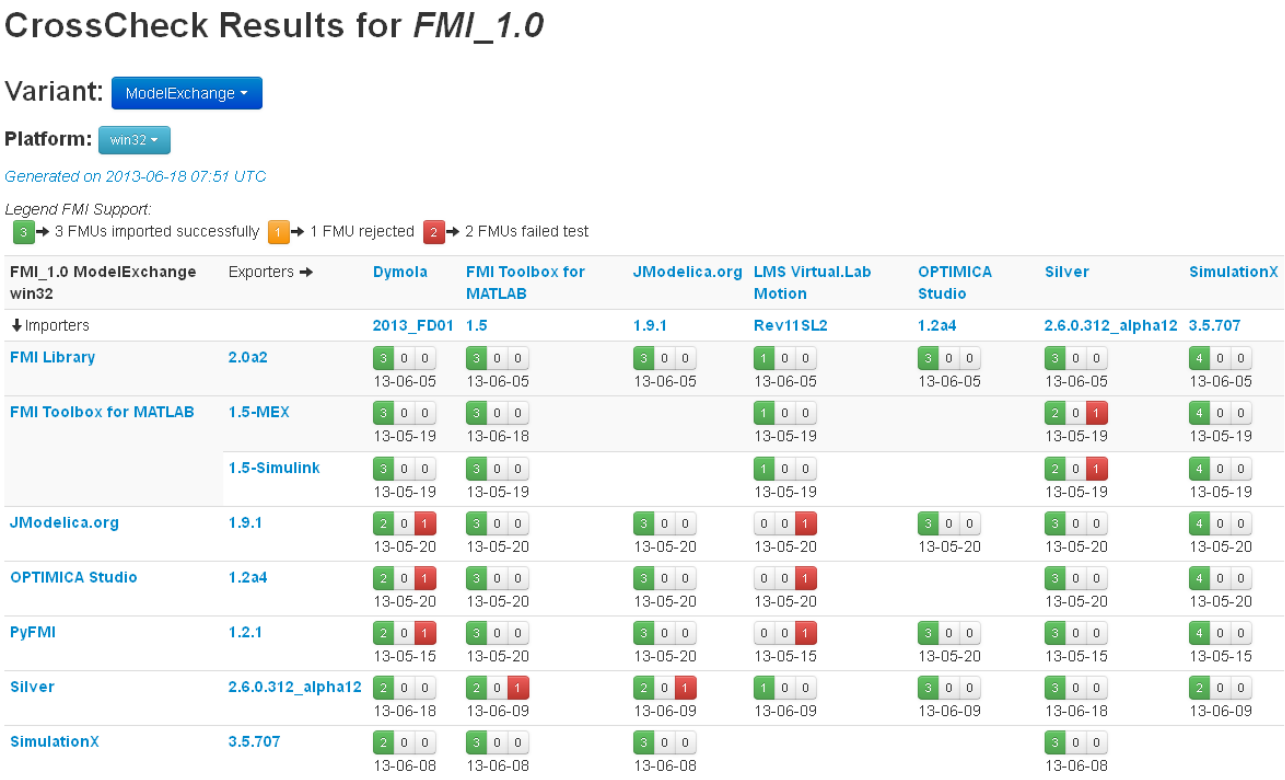
   The drop-down menu shall display only those platforms with FMUs exported for ("Available") and showing them "Available" when validated by Cross-Check Results (see above).
   "Available" in dark green for an "**Import** Column". The meaning is that the tool vendor has sufficiently checked the importing of FMUs and has submitted the required documentation to support his claim of Cross-Checking. When clicking on the button, the Cross-Check table of section 3 is shown.

   The drop-down menu shall display only those platforms with Cross-Check Results available ("Available") and showing them "Available" when sufficient Cross-Check Results are available (see above).

3. The Cross-Check Tables

A Cross-Check table offers a way to quickly learn which tool works well together with which other tool for which platform. Combinations not in the table might also work, but have not been tested yet.

The following screen shot shows the Cross-Check table as of June 18, 2013:



CrossCheck Results for *FMI_1.0*

Variant: ModelExchange

Platform: win32

Generated on 2013-06-18 07:51 UTC

Legend FMI Support:
3 → 3 FMUs imported successfully 1 → 1 FMU rejected 2 → 2 FMUs failed test

FMI_1.0 ModelExchange win32	Exporters →	Dymola	FMI Toolbox for MATLAB	JModelica.org	LMS Virtual.Lab Motion	OPTIMICA Studio	Silver	SimulationX
↓ Importers		2013_FD01	1.5	1.9.1	Rev11SL2	1.2a4	2.6.0.312_alpha12	3.5.707
FMI Library	2.0a2	3 0 0 13-06-05	3 0 0 13-06-05	3 0 0 13-06-05	1 0 0 13-06-05	3 0 0 13-06-05	3 0 0 13-06-05	4 0 0 13-06-05
FMI Toolbox for MATLAB	1.5-MEX	3 0 0 13-05-19	3 0 0 13-06-18		1 0 0 13-05-19		2 0 1 13-05-19	4 0 0 13-05-19
	1.5-Simulink	3 0 0 13-05-19	3 0 0 13-05-19		1 0 0 13-05-19		2 0 1 13-05-19	4 0 0 13-05-19
JModelica.org	1.9.1	2 0 1 13-05-20	3 0 0 13-05-20	3 0 0 13-05-20	0 0 1 13-05-20	3 0 0 13-05-20	3 0 0 13-05-20	4 0 0 13-05-20
OPTIMICA Studio	1.2a4	2 0 1 13-05-20	3 0 0 13-05-20	3 0 0 13-05-20	0 0 1 13-05-20		3 0 0 13-05-20	4 0 0 13-05-20
PyFMI	1.2.1	2 0 1 13-05-15	3 0 0 13-05-20	3 0 0 13-05-20	0 0 1 13-05-15	3 0 0 13-05-20	3 0 0 13-05-15	4 0 0 13-05-15
Silver	2.6.0.312_alpha12	2 0 0 13-06-18	2 0 1 13-06-09	2 0 1 13-06-09	1 0 0 13-06-09	3 0 0 13-06-09	3 0 0 13-06-18	2 0 0 13-06-09
SimulationX	3.5.707	2 0 0 13-06-08	3 0 0 13-06-08	3 0 0 13-06-08			3 0 0 13-06-08	

Illustration 1: Example Cross-Check Results Table (generated 2013/06/18)

Column and line headings are links to the corresponding tool's SVN directory containing supporting documentation (as described below). All cells contain links to the corresponding subversion directory for that specific Cross-Check. An entry "3,1,1, 13-12-15" means that 3 (green) of the 5 provided FMUs from the tool in the column were successfully imported into the tool of the row and could successfully be simulated (only non-0 entries are colored). One FMU was rejected by the importer as it required a feature not provided by the importing tool (yellow). One FMU was accepted, but failed to simulate or failed to simulate correctly (red). This check was reported to the FMI web on Dec. 15, 2013.

There will be tables for the different supported architectures, such as win32, win64, linux32, c-code, ...

Tables for FMI Version 2.0 will generated automatically, when results are submitted – the table generation is already in place.

The tables are generated from information stored in the underlying subversion directories in a machine-readable format to be defined below.

4. Reference FMUs

In addition to FMUs exported by tools listed in the tools table, there will be a (growing) number of Reference FMUs. Mostly manually built, they test correct implementation of different FMU properties, such as call sequence compliance. In a sense, these FMUs are the reverse of the ComplianceChecker.

Reference FMUs log standard violations using error messages and may stop the simulation to indicate severe violations of the standard.

Experience with the first Reference FMUs will help us to refine their usage better in the future.

5. The Rules of Cross-Check

	#	Rules
General	1	These rules of the FMI Cross Check are set and communicated in advance. They will be revised at most every 6 months during the first year, at most once every year thereafter. Vendors are encouraged to report problems and improvement suggestions for the Cross-Check Rules and Implementation Notes here: https://trac.fmi-standard.org for component "cross-check document ".
	2	FMI Cross Check tables will be published showing which exporting tools are compatible with which importing tools for given variants of the standard and architectures, including a compatibility score. The score indicates how many of the exported FMUs run successfully in the importing tool or where causing problems (e.g. 8,0,0, 8,1,1, or 0,0,8). Missing or incomplete testing, usually reserved for slots opened by new exporting tools in the table where importing tools have yet to test those new FMUs, will be indicated with an empty score.
	3	All tests must be made using officially released tool versions, **with the exception that license checks may be removed from exported FMUs by the tool vendors**. Vendors are encouraged to participate with multiple versions of their tools, as a tool version is often crucial for success, and users do not always use latest tool versions.
	4	All test results will be published on the FMI web pages and public SVN repository. Tool vendors are required to submit detailed information to support their compatibility claim and to make repeating compatibility tests easy for other parties. This information shall not be changed or removed once committed.
	5	The Steering Committee has the right to change the status or even remove a tool from the fmi-standard.org web pages if test results are deemed insufficient or unreproducible.
	6	FMU exporting tools export at least three FMUs for each platform and FMI variant they claim to support, and provide a short textual description of the content of each FMU. This should motivate vendors to not supply completely trivial examples. Vendors are encouraged to pick examples that showcase a wide variety of tool and FMI features without making each example unnecessarily complex. Furthermore, it is required that at least one importing tool has to report successful import of at least 3 FMUs for at least one of the supported platforms in order to qualify for "Available" status of the given FMI Variant. If the exporting tool exports only FMUs with license checks or source-code only, the "Available" status will be granted after 3 importing tool vendors reported error free imports for at least 3 FMUs provided by this exporting tool for at least one of the supported platforms.
	7	All FMUs submitted to the SVN server must run without license check and contain all required files (DLLs, data files etc.) to allow running in any importing tool supporting the specified platform without additional requirements. Vendors that create only FMUs with license check or source-code only and would like to be listed in the Cross-Check Table have to organize one-to-one tests with importing tool vendors to solve license or compile/link issues.
	8	Vendors must run (and pass) all submitted FMUs through the ComplianceChecker for all

		supported platforms and versions of exporting tool and variant. This test is deemed successful, if no errors are produced by the ComplianceChecker. Vendors are encouraged to produce FMUs that are also free of all ComplianceChecker warnings. FMUs for the "c-code" platform are exempt from this rule until the ComplianceChecker supports the "c-code" platform. If the FMU has inputs, the ComplianceChecker must be run with the input signals as specified in {FMUName}_in.csv. Vendors are encouraged to report problems and improvement suggestions for the ComplianceChecker here: https://trac.fmi-standard.org.
	9	To submit per exported FMU stored on the SVN server: • {FMUName}.fmu: The FMU • {FMUName}_ref.csv: Reference solution as computed by the exporting tool. It is recommended to limit the file to at most 10 of the important variables. • {FMUName}_in.csv: optional input signals in case the FMU has inputs. If intermediate values are required for continuous signals, linear interpolation is to be applied. • A ReadMe.txt or ReadMe.pdf with (see Appendix C for an example Reade.txt), e.g.: ◦ description of the FMU (features, intent, compile/link details,...) ◦ email address where to contact exporting company in case of import problems • {FMUName}_cc.bat: A batch file to run the experiment with the ComplianceChecker for Windows platforms, {FMUName}_cc.sh for *nix platforms • {FMUName}_cc.log: The log of the ComplianceChecker with minimal log level 3 (warning): -l 3. This allows smaller logs in case of excessively large files produced with the default -l 4. • {FMUName}_cc.csv: Result data for selected signals from the tool for that simulation as .csv file • {FMUName]_ref.opt: Options used to create reference output and to guide comparing against, CSV format, required elements: ◦ StartTime, 0.0 // in seconds ◦ StopTime, 0.0 // in seconds ◦ StepSize, 0.01 // in seconds, 0.0 means variable step solver ◦ RelTol, 0.0001 optional elements: ◦ AbsTol, 2 ◦ SolverType, FixedStep // see implementation notes for a list of predefined types Observe the naming conventions given here, including case. We recommend keeping {FMUName} short to avoid path length restriction problems on platforms like Windows.
FMU-Importer	10	FMU importing tools must report on importing for all Reference FMUs available for the supported FMI Variant and supported platforms provided on the SVN server (Note: not required to "pass" because of potentially missing capabilities of the FMU) and must successfully import at least 3 FMUs for at least one supported platform of at least 3 exporting tools, and run for as long as the {FMUName]_ref.opt states and supply a .csv file of the solution they computed to receive an "Available" status .
	11	To submit per imported FMU:

		• A ReadMe.txt or ReadMe.pdf with
		  ◦ a description of how to import and simulate each of the FMUs, if no test setup is provided
		  ◦ in case of failure to run: an analysis of the reasons.
		  This file is only needed if either test setup or test failure have to be described.
		• A test setup for the importing tool to simplify verification of the test run by anyone who licensed the importing tool. Ideally this uses some kind of automation provided by the importing tool.
		• {FMUName}_out.csv: Computed results as CSV file (CSV file format see Appendix B) for the same variables as given in the reference CSV file
		• In order to classify the result as "passed", the results should correspond to the reference solution.
		• Vendors are encouraged to produce a screen-shot of the results and the reference solution as displayed in the importing tool (for "important" signals) for simpler validation of their claim "passed".
		Vendors of tools that import FMUs for the "c-code" platform and would like to be listed in the Cross-Check Table have to organize one-to-one tests with exporting tool vendors in case they are not providing "c-code" FMUs publicly to produce testimonials for successful Cross-Check results.

6. Appendix A – Modelica models suggested for export

Modelica tools should export at least 3 models from the list below. These models have been selected to test the support of particular features of the FMU, see column 2. All models simulate with an explicit Euler algorithm, that is they can be simulated with the FMU Compliance Checker. A recommended step size for the explicit Euler algorithm is given in column 3.

Modelica tools should export CoupledClutchesWithControl using all variants (me and cs) that they support. CoupledClutchesWithControl adds top-level inputs to command some clutches to the MSL CoupledClutches model.

#	Tested properties	Modelica model
1	Real variables, Real input variable, continuous-time states, state events, event iteration,	**Modelica.Mechanics.Rotational.Examples.CoupledClutches** (replace component "step2" by an input and provide a step from the outside so that the input jumps at time = 0.9 s from 0.0 to 1.0). Explicit Euler: suggested step size = 1e-2 s
2	Boolean variables, Boolean input variable, no continuous-time states, time events (sample operator)	**Modelica.Blocks.Examples.BooleanNetwork1** (replace component booleanStep by an input and provide a step from the outside so that the input jumps at time = 1.5 s from false to true). Explicit Euler: suggested step size = 1e-2 s
3	Integer variables, Integer input variables, no continuous-time states, time events (sample operator), state events	**Modelica.Blocks.Examples.IntegerNetwork1** (replace component integerStep by an input and provide a step from the outside so that the input jumps at time = 2.0 s from 0 to 3). Explicit Euler: suggested step size = 1e-2 s
4	Enumeration variables, no continuous-time states, state events	**Modelica.Electrical.Digital.Registers.DFFREG** Explicit Euler: suggested step size = 1 s
5	Real array variables, continuous-time states, linear algebraic equation system, non-linear algebraic equation system	**Modelica.Mechanics.MultiBody.Examples.Loops.Engine1b** Explicit Euler: suggested step size = 1e-5 s
6	Very simple fluid model, continuous-time states function calls	**Modelica.Media.Examples.MixtureGases** Explicit Euler: suggested step size = 1e-3 s
7	Electrical, Thermal, Blocks domains, continuous-time states, state events, ideal switch	**Modelica.Thermal.HeatTransfer.Examples.ControlledTemperatureExplicit** Euler: suggested step size = 1e-3 s
8	Mixed Real/Boolean algebraic equation systems, state events, event iteration	**Modelica.Electrical.Analog.Examples.Rectifier** Euler: suggested step size = 1e-6 s

9	Medium-sized multi-domain model (36 states, 744 algebraic variables), Real array variables, continuous-time states, linear algebraic equation system, time events, state events, event iteration	**Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.fullRobot** Explicit Euler: suggested step size = 1e-5 s
10	Large fluid model, (1200 states, 10400 algebraic variables), Real array variables, continuous-time states	**Modelica.Fluid.Examples.HeatExchanger.HeatExchangerSimulation** (change nNodes from 20 to 400). Explicit Euler: suggested step size = 1e-3 s For explicit Euler simulate only for 30 s, instead of 100 s. Output file size when storing all variables is then around 2 Gbyte.

7. Appendix B – CSV-Format Rules

CSV as input, output and reference file format is convenient as many tools support CSV import and export. However, CSV is not restrictive enough to allow easy exchange of time-based signals. Here we will declare a number of additional restrictions on top of the CSV format to ease handling:

- Separator: comma (','): separators may only be used between elements, not the end of a line
- Numbers must be unquoted and specified in the format used for floating point literals as in the C programming language (without the type suffix).
- All numeric cell entries contain numbers, labels for enumerations are not allowed (it would require readers to have access to the FMU information). Boolean values should be expressed as 0 and 1.
- The first column contains the time.
- First line contains variable names: Variable names are quoted with double quotes (“). Variable names are the same as defined in the FMU-XML file.
- Starting with line 2, data is supplied (no units in line 2, no comments allowed)

8. Appendix C – Minimal Content ReadMe.txt Example

```
Model Description:
        Modelica.Mechanics.Rotational.Examples.CoupledClutches from the
MSL with the component “step2” replaced by an equivalent input.

Compiler:
        Microsoft Visual Studio 10 - 32 bit

Available platforms:
        win32, win64, lin32, lin64, darwin

Notes:
        If any, for example known issues

Contact: (new in spec)
        fmiContact@sampleCompany.com
```

More information can be given, but simulation options from the _cc.opt file take precedence is in conflict.