
FMU Export of the Opal-RT Simulator

Release 1.0.0

LBL - Building Technology and Urban Systems Division

Sep 27, 2017

CONTENTS

1	Introduction	1
2	Requirements	3
3	OPAL-RT FMU	5
4	Help	7
5	Notation	9
6	Glossary	11

INTRODUCTION

This user manual explains how to export the Python RT-LAB API of OPAL-RT as a *Functional Mock-up Unit* (FMU) for model exchange or co-simulation using the *Functional Mock-up Interface* (FMI) standard version 1.0 or 2.0. The RT-LAB API FMU allows to interface with the OPAL-RT real-time simulator through the FMI API.

Note: The RT-LAB API exported FMU has been tested on Windows with OPAL-RT/RT-Lab version 11.1.4.59.

REQUIREMENTS

The Python RT-LAB API of OPAL-RT is exported using the [SimulatorToFMU](#) Python package. The next sections assume that SimulatorToFMU is installed and configured.

OPAL-RT FMU

This section explains how to export the RT-LAB API of OPAL-RT as an FMU.

To export the Python RT-LAB API,

1. edit the XML template `SimulatorModeldescription.xml` provided in the utilities of the distribution of `SimulatorToFMU`
to specify the inputs, and outputs of the FMU.
2. run `SimulatorToFMU.py` to create the RT-LAB API FMU with

```
# Windows:  
> python parser\\SimulatorToFMU.py -s opalrtfm\\utilities\\simulator_  
↪ wrapper.py
```

Note: To use the FMU, the path to the configuration file of OPAL-RT will be set in the FMU. This file has the extension `.llp` and specifies the location of the OPAL-RT simulink grid model. Examples of such files can be found in the xamples folders of RT-LAB. The path to the configuration file will be either set by the master algorithm which imports the RT-LAB API FMU, or provided as `-c` argument for `SimulatorToFMU.py`. See the user guide of `SimulatorToFMU` for a list of arguments of `SimulatorToFMU`.

To use the FMU, the content of the `.binaries.zip` and the `.scripts.zip` folders need to be added to the `PATH` and the `PYTHONPATH` as described in the `SimulatorToFMU` user guide.

HELP

The help section of SimulatorToFMU dicusses issues and solutions encountered when exporting or running the RT-LAB API FMU.

NOTATION

This chapter shows the formatting conventions used throughout the User Guide.

The command-line is an interactive session for issuing commands to the operating system. Examples include a DOS prompt on Windows, a command shell on Linux, and a Terminal window on MacOS.

The User Guide represents a command window like this:

```
# This is a comment.  
> (This is the command prompt, where you enter a command)  
(If shown, this is sample output in response to the command)
```

Note that your system may use a different symbol than “>” as the command prompt (for example, “\$”). Furthermore, the prompt may include information such as the name of your system, or the name of the current subdirectory.

GLOSSARY

Dymola Dymola, Dynamic Modeling Laboratory, is a modeling and simulation environment for the Modelica language.

Functional Mock-up Interface The Functional Mock-up Interface (FMI) is the result of the Information Technology for European Advancement (ITEA2) project *MODELISAR*. The FMI standard is a tool independent standard to support both model exchange and co-simulation of dynamic models using a combination of XML-files, C-header files, C-code or binaries.

Functional Mock-up Unit A simulation model or program which implements the FMI standard is called Functional Mock-up Unit (FMU). An FMU comes along with a small set of C-functions (FMI functions) whose input and return arguments are defined by the FMI standard. These C-functions can be provided in source and/or binary form. The FMI functions are called by a simulator to create one or more instances of the FMU. The functions are also used to run the FMUs, typically together with other models. An FMU may either require the importing tool to perform numerical integration (model-exchange) or be self-integrating (co-simulation). An FMU is distributed in the form of a zip-file that contains shared libraries, which contain the implementation of the FMI functions and/or source code of the FMI functions, an XML-file, also called the model description file, which contains the variable definitions as well as meta-information of the model, additional files such as tables, images or documentation that might be relevant for the model.

Modelica Modelica is a non-proprietary, object-oriented, equation-based language to conveniently model complex physical systems containing, e.g., mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents.

MODELISAR MODELISAR is an ITEA 2 (Information Technology for European Advancement) European project aiming to improve the design of systems and of embedded software in vehicles.

PyFMI PyFMI is a package for loading and interacting with Functional Mock-Up Units (FMUs), which are compiled dynamic models compliant with the Functional Mock-Up Interface (FMI).

Python Python is a dynamic programming language that is used in a wide variety of application domains.

INDEX

D

Dymola, [11](#)

F

Functional Mock-up Interface, [11](#)

Functional Mock-up Unit, [11](#)

M

Modelica, [11](#)

MODELISAR, [11](#)

P

PyFMI, [11](#)

Python, [11](#)