FMU Export of CYMDIST

Release 1.0.0

LBNL - Building Technology and Urban Systems Division

Nov 07, 2016

CONTENTS

CHAPTER

ONE

INTRODUCTION

This user manual explains how to install and use CYMDISTPy. CYMDISTPy is a software package written in Python which allows users to export the distribution simulation program CYMDIST version 7.2 or as a Functional Mock-up Unit (FMU) for model exchange using the Functional Mock-up Interface (FMI) standard version 2.0. This FMU can then be imported into a variety of simulation programs that support the import of the Functional Mock-up Interface for model exchange.

CHAPTER

TWO

DOWNLOAD

The CYMDISTPy release includes scripts and source code to export CYMDIST version 7.2 as an FMU for model exchange for Windows 32.

To install CYMDISTPy, follow the section *Installation and Configuration*.

Release 1.0.0 (December 01, 2016)

Download CYMDISTPy-1.0.0.zip.

Release notes

First release that uses FMI version 2.0 for model exchange.

CHAPTER

THREE

INSTALLATION AND CONFIGURATION

This chapter describes how to install, configure and uninstall CYMDISTPy.

Software requirements

To export CYMDIST as an FMU, CYMDISTPy needs:

1. Python 3.4.x. 32bit
2. jinja2
3. lxml
4. pandas
5. numpy
6. cython
7. Dymola

CYMDISTPy has been tested with:

- Windows 10 64 bit Professional.
- Dymola 2017.

Installation

To install CYMDISTPy, proceed as follows:

1. Download the installation file from the *Download* page.
2. Unzip the installation file into any subdirectory (hereafter referred to as the "installation directory").

The installation directory should contain the following subdirectories:

- `dev/`
  - `parser/` (Python scripts, Modelica templates and xsd validator files.)
  - `cymdist/` (Python scripts for communicating with CYMDIST.)
  - `fmuChecker/` (fmuChecker binaries for running unit test)
  - `bin/` (Scripts for running unit tests.)
  - `fmus/` (FMUs folder.)

- `models/` (Examples models)
  - `cymdist`

3. Add following folders to your system path:

- Python installation folder (e.g. `C:\Python34`)
- Python scripts folder (e.g. `C:\Python34\Scripts`),
- Dymola executable folder (e.g. `C:\Program Files(x86)\Dymola2017\bin`)

  You can add folders to your system path by performing following steps on Windows 8 or 10:

  - In Search, search for and then select: System (Control Panel)
  - Click the Advanced system settings link.
  - Click Environment Variables. In the section System Variables, find the PATH environment variable and select it. Click Edit.
  - In the Edit System Variable (or New System Variable) window, specify the value of the PATH environment variable (e.g. `C:\Python34`, `C:\Python34\Scripts`). Click OK. Close all remaining windows by clicking OK.
  - Reopen Command prompt window for your changes to be active.

  To check if the variables have been correctly added to the system path, type `python` into a command prompt to see if the right version of Python starts up.

4. Install Python dependencies by running

```
pip install -r bin/cymdistpy-dependencies.txt
```

Note: `lxml` cannot be installed using `pip`. Please download and install the executable (`lxml-3.4.4.win32-py3.4.exe`) from PyPyi.

Uninstallation

To uninstall CYMDISTPy, delete the *installation directory*.

CHAPTER

FOUR

CREATING AN FMU

This chapter describes how to create a Functional Mockup Unit, starting from a CYMDIST model description input file. It assumes you have followed the *Installation and Configuration* instructions, and that you have created the CYMDIST model description file following the *Best Practice* guidelines.

Command-line use

To create an FMU, open a command-line window (see *Notation*). The standard invocation of the CYMDISTPy tool is:

```
> python  <scriptDir>CYMDISTWritter.py  \
  -g <path-to-grid-model-file>  \
  -i <path-to-input-file> -b <path-to-buildings-lib-file> \
  -m <path-to-modelica-template-file> \
  -s <path-to-modelica-script-template-file> -x <path-to-xsd-file> \
  -r <flag-to-write-results>
```

where `scriptDir` is the path to the scripts directory of CYMDISTPy. This is the `parser/` subdirectory of the installation directory. See *Installation and Configuration* for details.

If `CYMDISTWritter.py` is run from within `scriptDir` then only the required arguments are needed as shown in the following example:

```
# Windows:
> python  CYMDISTWritter.py  -g C:\grid\test.sxst  -i test.xml  -b modelica-buildings
```

where `scriptDir` is the path to the scripts directory of CYMDISTPy. Typically this is the `parser/` subdirectory of the installation directory. See *Installation and Configuration* for details.

All file paths can be absolute or relative. For readability, the rest of these instructions omit the paths to the script and input files.

Note: If any file path contains spaces, then it must be surrounded with double quotes.

Script `CYMDISTWritter.py` supports the following command-line switches:

option <argument>	Purpose
-g <path-to-grid-model-file>	Path to the grid model (required).
-i <path-to-input-file>	Path to the input file (required).
-b <path-to-buildings-lib-file>	Path to the Buildings library (required).
-m <path-to-modelica-template-file>	Path to the Modelica template file This file is in the `scriptDir`. If the script is run for the `scriptDir` then this file is not required.
-s <path-to-modelica-script-template-file>	Path to the Modelica script template file This file is in the `scriptDir`. If the script is run for the `scriptDir` then this file is not required.
-x <path-to-xsd-file>	Path to the `XSD` validator file. This file is in the `scriptDir`. If the script is run for the `scriptDir` then this file is not required.
-r <flag-to-write-results>	Flag to write results. 0 if results should not be written, 1 else. Default is 0.

The main functions of CYMDISTPy are

- reading, validating, andparsing the CYMDIST XML input file,
- writing Modelica code with input and outputs,
- invoking Dymola to compile the Modelica code as an FMU for model exchange 2.0.

Note: In the process of creating the FMU, CYMDISTPy will rewrite the model description file generated by Dymola to include `needsExecutionTool=true` attribute in the FMU capabilities of the CYMDIST FMU. This is currently not supported by default in Dymola.

Output

The main output from running `CYMDISTWritter.py` consists of an FMU, named after the modelName specified in the input file. The FMU is written to the current working directory, that is, in the directory from which you entered the command.

The FMU is complete and self-contained. Any secondary output from running the CYMDISTPy tools can be deleted safely.

Note that the FMU is a zip file. This means you can open and inspect its contents. To do so, it may help to change the "`.fmu`" extension to "`.zip`".

CHAPTER

FIVE

USAGE OF CYMDIST AS AN FMU

The following items must be observed to import an FMU that contains CYMDIST:

1. `Python 3.4` must be installed.
2. `CYME` version 7.2 must be installed. CYME can be downloaded from www.cyme.com.
3. The CYMDIST Python API scripts directory must be added to the `PYTHONPATH`.

Note: The CYMDIST Python API scripts are in the installation folder of CYME. It can typically be found in `pathCYME\CYME\cympy`, where `pathCYME` is the path to the installation folder of CYME 7.2.

To add the CYMDIST Python API scripts folder to the `PYTHONPATH`:

- In Search, search for and then select: System (Control Panel).
- Click the Advanced system settings link.
- Click Environment Variables. In the section System Variables, find a variable named `PYTHONPATH` environment variable and select it. If the variable does not exist, create it. Click Edit.
- In the Edit System Variable (or New System Variable) window, specify the value of the PYTHONPATH environment variable which should in our case be `pathCYME\CYME`. Note that `cympy` is not included in the name of the variable.

4. The `cymdist` utility folder must be added to the `PYTHONPATH`. The `cymdist` utility folder can be found in the distribution of CYMDISTPy. It is in `dev/cymdist`. To complete your set-up, add the folder `dev\cymdist` to the `PYTHONPATH`.

CHAPTER

SIX

BEST PRACTICE

This section explains to users the best practice in configuring a CYMDIST XML input file for an FMU.

To export CYMDIST as an FMU, the user needs to write an XML file which contains the list of inputs, outputs and parameters of the FMU. The XML snippet below shows how a user has to write such an input file. A template named `CYMDISTModeldescritpion.xml` which shows such a file is provided in the `parser` installation folder of CYMDISTPy. This template should be used and modify to create new XML input file.

The following snippet shows an input file where the user defines one input and one output variable.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <CYMDISTModelDescription
3    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4    fmiVersion="2.0"
5    modelName="HL0004"
6    description="Input data for a CYMDIST FMU"
7    generationTool="CYMDISTPy">
8    <ModelVariables>
9      <ScalarVariable
10       name="VMAG_A"
11       description="VMAG_A"
12       causality="input">
13       <Real
14         unit="kV"
15         start="0.0"/>
16     </ScalarVariable>
17     <ScalarVariable
18       name="voltage_A"
19       description="voltage_A"
20       causality="output">
21       <Real
22         unit="V"/>
23       <Device
24         name="HOLLISTER_2104"/>
25     </ScalarVariable>
26   </ModelVariables>
27 </CYMDISTModelDescription>
```

To create such an input file, the user needs to specify the name of the FMU (Line 5). This is the `modelName` which should be unique. The user then needs to define the inputs and outputs of the FMUs. This is done by adding `ScalarVariable` into the list of `ModelVariables`.

To parametrize the `ScalarVariable` as an input variable, the user needs to

- define the name of the variable (Line 10),
- give a brief description of the variable (Line 11)

- give the causality of the variable (`input` for inputs, `output` for outputs) (Line 12)
- define the type of variable (Currently only `Real` variables are supported) (Line 13)
- give the unit of the variable (Currently only valid Modelica units are supported) (Line 14)
- give a start value for the input variable (This is optional) (Line 15)

To parametrize the `ScalarVariable` as an output variable, the user needs to

- define the name of the variable (Line 18),
- give a brief description of the variable (Line 19)
- give the causality of the variable (`input` for inputs, `output` for outputs) (Line 20)
- define the type of variable (Currently only `Real` variables are supported) (Line 21)
- give the unit of the variable (Currently only valid Modelica units are supported) (Line 22)
- give the name of the output device (Line 24)

Note: To avoid name-clash, CYMDISTPy concatenates the name of the output with the name of the device to make it unique. The new output name will have the form `outputName_DeviceName`.

CHAPTER

SEVEN

DEVELOPMENT

The development site of this software is at https://github.com/lbl-srg/cyder.

To clone the master branch, type

git clone https://github.com/lbl-srg/cyder.git

CHAPTER

EIGHT

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.

CHAPTER

NINE

ACKNOWLEDGMENTS

The development of this documentation was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technologies of the U.S. Department of Energy, under contract No. xxx.

The following people contributed to the development of this program:

- Thierry Stephane Nouidui, Lawrence Berkeley National Laboratory
- Jonathan Coignard, Lawrence Berkeley National Laboratory
- Michael Wetter, Lawrence Berkeley National Laboratory

CHAPTER

TEN

DISCLAIMERS

CHAPTER

ELEVEN

COPYRIGHT AND LICENSE

xxxx fixme xxx still needed