LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Note	**LIGO-T1100340-v1**	2011/06/23

RCG post_build_script

Joseph Betzwieser and Jameson Rollins

Distribution of this document:

LIGO ALL

California Institute of Technology
LIGO Project, MS 18-34
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project, Room NW22-295
Cambridge, MA 02139
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

LIGO Hanford Observatory
Route 10, Mile Marker 2
Richland, WA 99352
Phone (509) 372-8106
Fax (509) 372-8137
E-mail: info@ligo.caltech.edu

LIGO Livingston Observatory
19100 LIGO Lane
Livingston, LA 70754
Phone (225) 686-3100
Fax (225) 686-7189
E-mail: info@ligo.caltech.edu

http://www.ligo.caltech.edu/

1 Overview

This document is intended to explain how to use the post_build_script.py script to generate screens and call scripts after building a front end code.

2 Installation

The script requires python to be installed and as of this time has been tested with python 2.4.3 and python 2.6.5, although it should work most versions.

The post_build_script.py file can currently be found in the CDS cds_user_apps SVN repository. The script is located in /trunk/cds/common/scripts/.

The websvn for the userapps is located at https://redoubt.ligo-wa.caltech.edu/websvn/

3 Environment setup

The code requires some environmental variables be defined to run. A useful reference is Keith's DCC document on environment setup, T1000379, and on directory structures,T1000248.

3.1 SITE

`SITE` which should be set to the site name, such as llo, lho, etc.

3.2 IFO

`IFO` which should be set to the ifo designation, such as h1, h2, l1.

3.3 RCG_LIB_PATH

`RCG_LIB_PATH` is a set of paths to all directories where the simulink model files and libraries exist. This path includes the RCG simulink directories at

`/opt/rtcds/<site>/<ifo>/core/release/src/epics/simLink/`

and

`/opt/rtcds/<site>/<ifo>/core/release/src/epics/simLink/lib/`

It also includes all the userapps directories which contain models,

`/opt/rtcds/<site>/<ifo>/userapps/release/<system>/<ifo,common>/models/`

and

```
/opt/rtcds/<site>/<ifo>/userapps/release/<system>/<ifo,common>/models/templates/
```

3.4 CDS_MEDM_PATH

The `CDS_MEDM_PATH` is a set of paths to all directories where MEDM templates exist. This would include

```
/opt/rtcds/<site>/<ifo>/userapps/release/<system>/<ifo,common>/medm/templates/
```

3.5 CDS_SCRIPTS_PATH

The `CDS_SCRIPTS_PATH` is a set of paths to all directories where scripts exist. This would include

```
/opt/rtcds/<site>/<ifo>/userapps/release/<system>/<ifo,common>/scripts/
```

4 Use in .mdl files

To use this script, it requires command be placed within the model .mdl file. To add a command, go to a subsystem block, right click and select `block properties`. Commands are then added to the description field.

Lines must be seperated by newlines. Commands must begin at the beginning of the line, with no spaces and no new lines until the end of the command.

The 3 command options are:

```
ADL=
SCRIPT=
NO_DEFAULT
```

All lines which do not begin with the above commands are ignored, so normal descriptions are unaffected. If the piece is a library part, all descriptions in the library references (even if its a library part in another library) will be parsed and any commands executed. The order of execution is farthest library reference first, then ending with the current model's description block.

4.1 NO_DEFAULT

If this subsystem is a library part `NO_DEFAULT` prevents the library part's description from being parsed and executed as well. It has no effect on library parts inside the subsystem, just if the subsystem itself is a library part.

4.2 ADL=

ADL= is used to specify a medm .adl file template to be substituted. The CDS_MEDM_PATH will be searched to find the file. The resulting MEDM screen is sent to MEDM_TARGET, which defaults to

```
/opt/rtcds/<site>/<ifo>/medm/<model_name>/<site><part_name>
```

when not set.

This output will have a standard set of substitutions performed on it before being written.

Let us consider the example ULSEN filter module in the BS subsystem in the c1sus model, run with SITE equal to caltech and IFO equal to c1. It has DCU_ID equal to 25. MEDM_TARGET is as noted above. The standard substitions for this part would be:

```
#SYS# becomes SUS
#CHANNEL# becomes C1:SUS-BS_ULSEN
#FULL_PART_NAME# becomes C1SUS_BS_ULSEN
#PART_NAME# becomes ULSEN
#SITE# becomes CALTECH
#site# becomes caltech
#IFO# becomes C1
#ifo# becomes c1
#DCU_ID# becomes 25
#DATA_RATE# becomes 16384
#TARGET_DIR# becomes /opt/rtcds/caltech/c1/medm/c1sus
#MODEL_NAME# becomes c1sus
#SYS[0]# becomes SUS
#SYS[1]# becomes BS
#SYS[2]# becomes ULSEN
#SYS[-1]# becomes ULSEN
#SYS[-2]# becomes BS
#SYS[-3]# becomes SUS
```

Options can be added after the template, using comma separation. An example where all instances of ALTPOS is replaced with MCL is:

```
ADL=SUS_SINGLE.adl,ALTPOS=MCL
```

The name of the output file can be overriden with the --name= option. The name will also have the standard substitions run on it before being used. For example,

```
ADL=SUS_SINGLE.adl,--name=#SYS#_myname.adl,ALTPOS=MCL
```

would produce a file called SUS_myname.adl, that also had all instances of ALTPOS replaced by MCL.

4.3 SCRIPT=

SCRIPT= is used to specify a script to be run substituted. The CDS_SCRIPT_PATH will be searched to find the script. Text after the first space will be interpreted as arguments to the script. The standard set of substitions from the ADL= section will also be run on the script file name and its arguments.

Continuing our example from the ADL= section,

```
SCRIPT=generate_KisselButton.py 3 4 #CHANNEL# >
#TARGET_DIR#/#FULL_PART_NAME#_KB.adl
```

will run the generate_KisselButton.py script with the arguments

```
3 4 C1:SUS-BS_ULSEN >
/opt/rtcds/caltech/c1/medm/c1sus/C1SUS_BS_ULSEN_KB.adl
```

There is no new line in the middle of command, it is wrapping around to a second line due to formating needed to fit this PDF. Newlines are used as seperators between different commands.

5 Creating MEDM Templates

These instructions assume you are starting from an already working MEDM screen.

First save the template with an approriate name in the approriate userapps directory. For the moment we will assume the initial .adl was called S1ISI_CUST_ITMX_COMMANDS.adl. As an example, we save it as S1ISI_OPTIC_COMMANDS.adl in /userapps/release/isi/s1/medm/templates. As long as this directory is in the CDS_MEDM_PATH it will be found when the script is run.

In this file we have the line

```
command[0] {
        label="damp"
        name="$BTL_SCRIPTS/BSCISItool"
        args="damp s1isiitmx &"
}
```

which corresponds to a script execution button command line. We make this script generic, we need to change s1isiitmx to #MODEL_NAME#. This can be done with medm and editing the screen, a text editor, or with sed.

For example,

```
sed -i 's/s1isiitmx/#MODEL_NAME#/g' S1ISI_OPTIC_COMMANDS.adl
```

will change all instances of s1isiitmx into #MODEL_NAME# in the S1ISI_OPTIC_COMMANDS.adl file.

The new lines would look like

```
command[0] {
        label="damp"
        name="$BTL_SCRIPTS/BSCISItool"
        args="damp #MODEL_NAME# &"
}
```

To use this template file, we now need to modify the s1isiitmx.mdl file.

Since this is related to the ITMX optic, we decide to put the command in the description for the top level ITMX block which is in the s1isiitmx.mdl file.

First open the model in MATLAB. Then right click on the ITMX block and select Block Properties. Finally in the Description area write

```
ADL=S1ISI_OPTIC_COMMANDS.adl,--name=S1ISI_CUST_#PART_NAME#_COMMANDS.adl
```

When the command

```
./post_build_script.py c1isiitmx
```

is run it will generate a new screen called S1ISI_CUST_ITMX_COMMANDS.adl. Assuming the environment variable CDS_MEDM_TARGET was left unset and SITE was set to stn and IFO was set to s1, the file will be put in the /opt/rtcds/stn/s1/medm/s1isiitmx/ directory.