

Core to SAL Translation

User guide

Gilad Hemi
Ayelet Bar-Niv Shiloh

TOC

Core to SAL Translation.....	1
TOC.....	2
1. Working with the program.....	2
2. Assumptions.....	3
3. Program files.....	3

1. Working with the program

1. Working with Core_To_SAL script (phases 1 and 2)
The script receives up to 4 arguments. The first argument is mandatory and contains the name of the input cdl file (without any suffix). The last 3 arguments are optional according to the following list:

- O:*filename* : output flattened xml full filename (optional).
- X:*value* : set the MAX_INT (optional. default is taken from settings.xml).
- N:*value* : set the MIN_INT (optional. default is taken from settings.xml).

The script sets the environment variables; runs phase 1 and then phase 2.

2. Working with CTS_Flat executable (phase 1)
The program gets one parameter which is the filename of the input program file cdl.

- 📄 Input file: *filename.cdl*
- 📄 Output file: *filename.cdl_cdl.xml*: The representations of the input file in xml.
- 📄 Output file: *filename.cdl_flat_cdl.xml*: The xml input file after necessary flattening for phase 2. In case that flattening was not necessary, this file exists and is similar to *filename.cdl_cdl.xml*.
- 📄 Output files: *filename.cdl_cdl.xml_#_log.xml*: There is a log file for each flattened module, to hold the additional information of the flattening process.

3. Working with coreToSal executable (phase 2)
When compiled in unix the program uses oracle xml library (can be found at http://otn.oracle.com/software/tech/xml/xdk_cpp/index.html). This requires two environment variables pointing to the msg and nlsdata directories in the project e.g.

```
setenv ORA_XML_MSG $HOME/proj/xml/xdk/msg
setenv ORA-NLS33 $HOME/proj/xml/nlsdata
```

coreToSal works with the following flags:

- I:*filename* : input flattened xml full filename (mandatory).
- O:*filename* : output flattened xml full filename (optional).

-X:*value* : set the MAX_INT (optional. default is taken from settings.xml).
-N:*value* : set the MIN_INT (optional. default is taken from settings.xml).

- 📄 Input file: *filename*.*.xml
- 📄 Output file: *filename*_sal.xml: The translated program in sal xml format (unless another filename was specified by the -O flag).
- 📄 Output file: *filename*_addInfo.xml: The changes log file.

4. Working with SAL tools by SRI.

The SRI programs work under Linux environment, and handle SAL xml program specifications.

They require an environment variable that points to the xml folder e.g.

```
setenv SALPATH /local/home/students-workers/gilad_sl/xml
```

5. NOTE : The `sal-pretty-printer` can probably work on input xml files of size < 42KB. The `card_game.cdl` example worked only when we reduced the card options from (2..joker) to (10..joker) reducing the `card_game_sal.xml` file size to 41KB.

2. Assumptions

1. The input file should be a correct core file, namely it should have a SYSTEM module, and not have recursive module combinations.
 2. If hold previous flag is off, all arrays are of one dimension (to simplify the array loop).
 3. When a nondeterministic assignment for integer is needed (split transition and hold previous stages), the integer type is replaced by the `ctsInt` type, and is bounded to MIN_INT..MAX_INT. These boundaries are given in a xml file called settings.xml .
 4. We assume there are no underscores in the modules and transitions names (actually, we can reduce this assumption, and assume that there is no name of a transition as a sub-string in another transition name, that also have an underscore, but we prefer to simplify the assumptions, to prevent the odds of mistakes caused by misunderstanding this assumption).
- All bad inputs generate friendly error messages except for errors in Core2XML stage (this part in the translation is done using Core2XML utility, and not by our code).

3. Program files

- 📁 coreToSal main directory
 - 📄 Core_To_SAL [unix]: is a script that runs phases 1 and 2 to translate a cdl file to a sal xml file.
 - 📄 run_examples[unix]: a script that runs all the examples under the examples folder.
 - 📄 env.txt [unix]: list of all the environment variables that need to be changed for the oracle library will work

- 📄 settings.xml: the cts setup file (xml file that holds the values of MAX_INT and MIN_INT).
- 📁 bin: Phase 2 binaries.
 - 📄 coreToSal: The executable that runs phase 2.
- 📁 Flat: Phase 1 files.
 - 📁 CTS_Flat
 - 📁 bin: Phase 1 binaries.
 - 📄 CTS_Flat: The executable that runs phase 1.
 - 📁 CTS_flat: visual studio 6 workspace for phase 1.
 - 📁 hdr: Phase 1 header files.
 - 📁 obj: Phase 1 object files.
 - 📁 src: Phase 1 source files.
 - 📁 CTS_Flat_examples: examples for phase 1 only.
- 📁 linux: utilities for sal.
 - 📄 env.txt [linux]: lists the environment variables needed by sal programs.
 - 📄 runEx [linux]: a script that runs sal-pretty-printer to convert *.sal.xml to *.sal.