

CPACSGen

CPACSGen (short for CPACS generator) creates source code for classes and enums from the types defined in a CPACS XML schema file for the TiGL library.

GitHub

CPACSGen is hosted on Github:

https://github.com/RlanderRISCSW/cpacs_tigl_gen

Dependencies

The following components are required to build and run CPACSGen

- A working copy of CPACSGen
- CMake
- C++11/C++14 compliant compiler (VS2015 has been tested)
- TIXI library (<https://github.com/DLR-SC/tixi>)
- Boost libraries (v1.55 or higher)
- A working copy of TiGL with CPACSGen input files (<https://github.com/DLR-SC/tigl>)

Input files

CPACSGen requires the following input files:

- cpacs_schema.xml
- CustomTypes.txt
- FundamentalTypes.txt
- ParentPointer.txt
- PrefixedEnums.txt
- PruneList.txt
- ReservedNames.txt
- TypeSubstitution.txt

These files are typically part of the TiGL distribution and do not have to be created. Altering these files influences CPACSGen's output and may require further adaption of the TiGL code depending on the generated code.

Workflow

1. Clone the CPACSGen repository from Github
2. Clone the TiGL repository Github
3. Use CMake to configure CPACSGen
 - a. Set BOOST_ROOT
 - b. Set TIXI_PATH
 - c. Set TIGL_PATH to the directory of your TiGL clone
4. Build CPACSGen
5. Build target generate
executes CPACSGen \${TIGL_DIR}/cpacs_gen_input \${CPACSGEN_DIR}/src
\${TIGL_DIR}/src/generated
6. Use CMake to configure TiGL (which picks up generated files)
7. Build TiGL

Command line options

The CPACSGen executable takes the following positional command line options:

1. Input directory
The directory containing the CPACS schema and the table files
2. CPACSGen src directory
The src directory of the CPACSGen source code, the CPACSGen output runtime files are taken from here
3. Output directory
The directory to which the CPACSGen output files are written
4. (optional) output Graphviz file
If a filename is specified, writes a Graphviz dot file containing a directed graph of the created CPACS types. This is especially useful when optimizing PruneList.txt

Table details

- CustomTypes.txt
Types which will be sub-classed in TIGL, e.g. for implementing additional behavior or fixing issues with the generated code. Types for which a custom type has been specified will never be instantiated by the generated code. Instead, their customized derived classes are instantiated. Customized types are still outputted.
e.g. `tigl::generated::CPACSFuselage` -> `tigl::CCPACSFuselage`
- FundamentalTypes.txt
`std::string`, `double`, `bool`, `int`, `time_t`
Fundamental data types for which XML element and attribute IO functions exist in the TIXI wrapper.
- ParentPointer.txt
e.g. `CPACSWing`, `CPACSFuselageSegment`
List of types which should hold a pointer to their parent element in CPACS. Causes all parent classes to pass themselves to their child classes on child instantiation.
- PrefixedEnums.txt
e.g. `CPACSInterpolation`
Enums, which's values should be prefixed by the enum name (e.g. `CPACSInterpolation_1` instead of `_1`). This may be necessary, if two different enums have a value of the same name. This issue resolves itself when C++11 is available and scoped enums are used (enum class).
- PruneList.txt
List of CPACS types which (including their child types in the CPACS schema tree) should be removed from the type system built by CPACSGen. Pruned types are not outputted. A graphical representation of the type system after pruning can be obtained using the optional Graphviz file command line argument.
- ReservedNames.txt
e.g. `int`, `inline`, `break`, `false`
List of identifiers which are reserved in C++. Members of generated enums are checked against this table and altered.
- TypeSubstitution.txt
e.g. `stringUIDBaseType` -> `std::string`, `integerBaseType` -> `int`, `xsd:unsignedLong` -> `uint64_t`
Types of the CPACS schema which should simply be replaced without further logic. Replaced types are not outputted.

Generated code dependencies

The classes generated by CPACSGen depend upon the following helper files, which are part of CPACSGen and will be written to the output directory when CPACSGen is invoked:

- `TixiHelper.h/.cpp`
Wrappers over TIXI functions providing a C++ friendly interface (references, `std::string`, exceptions, ...). Is used by the `ReadCPACS` and `WriteCPACS` implementation.
- `UniquePtr.hpp`
Contains `tigl::unique_ptr<T>` which defaults to `std::unique_ptr<T>` if C++11 is available, otherwise provides a custom type based on `std::auto_ptr<T>`.

Furthermore, the generated classes may require `boost::optional<T>`, which is part of TiGL, and some internal TiGL files.