


C++11 Mapping for Identifiers



A Slice [identifier](#) maps to an identical C++ identifier. For example, the Slice identifier `clock` becomes the C++ identifier `clock`. There is one exception to this rule: if a Slice identifier is the same as a C++ keyword, the corresponding C++ identifier is prefixed with `_cpp_`. For example, the Slice identifier `while` is mapped as `_cpp_while`.

A single Slice identifier often results in several C++ identifiers. For example, for a Slice interface named `Foo`, the generated C++ code uses the identifiers `Foo` and `FooPrx` (among others). If the interface has the name `while`, the generated identifiers are `_cpp_while` and `whilePrx` (*not* `_cpp_whilePrx`), that is, the prefix is applied only to those generated identifiers that actually require it.

 You should try to [avoid such identifiers](#) as much as possible.

[Back to Top](#) ^

See Also

- [Lexical Rules](#)
- [C++11 Mapping for Modules](#)
- [C++11 Mapping for Built-In Types](#)
- [C++11 Mapping for Enumerations](#)
- [C++11 Mapping for Structures](#)
- [C++11 Mapping for Sequences](#)
- [C++11 Mapping for Dictionaries](#)
- [C++11 Mapping for Constants](#)
- [C++11 Mapping for Exceptions](#)

