

DRAFT

Unit-testing private C++ functions

Warning

There exists significant debate about whether or not private functions should ever be unit-tested. In most situations, it should be preferable to test the public functions of the class, and to use coverage analysis to make sure that private functions are tested as well.

However, there may exist circumstances in which testing and debugging is considerably simplified by testing a private class. For example, the private functions may be too complex to easily generate thorough coverage without calling them directly. In these cases, you may go ahead and directly test a private function.

The Problem with Boost Unit Testing Macros

Boost's unit testing macros create test classes at compile-time. The names of these classes are not known to the programmer, and thus it is **not** possible to simply declare them as "friend" to the tested class.

Some quick searching on Google will reveal some dubious ideas, such as include guards which re-declare the private functions as public if a "testing" flag is set. This will mean that code compiled for production will **not** pass tests. This is not an optimal solution.

How to do it, if you must

Thus, it is suggested to create an additional "bridge" class which is declared 'friend' to the tested class. This "bridge" class can then call the private functions of the tested class, and the unit tests should call an instance of the "bridge" class.

The "bridge" class should live in its own namespace ending in `::impl` e.g. to test a class in `lsst::mops`, put a bridge class in `lsst::mops::impl`. The code inside an `::impl` namespace should **never** be used outside of testing, and it should be generally understood that the contents of an `::impl` namespace are highly unstable.