# SharedPtr

std::tr1::shared_ptr is an almost-standard smart pointer template that provides unintrusive reference-counting semantics for any class. It almost makes memory management too easy for a C++ programmer.

It's available in g++ and some other compilers by default. There are several open source implementations if we ever port to a compiler that doesn't have it.

The golde rule: if class Foo has shared ownership then never *ever* write `Foo*`. Anywhere. Ever. Use shared_ptr in all function signatures and variables, use std::tr1::dynamic_pointer_cast and friends for casting.

Qpid will use it for all classes with shared ownership semantics, enforced by private constructors and static factory functions. We'll also adopt the convention to typedef shared_ptr within the class for convenience. E.g.

```
class Foo {
   Foo() { ... }

 public:
   typedef std::tr1::shared_ptr<Foo> shared_ptr;
   static shared_ptr create() { return new Foo() }
   // .. a create for each constructor.
}

Foo::shared_ptr p = Foo::create();  // etc...
```

There's a good article at http://www.boost.org/libs/smart_ptr/sp_techniques.html.