The C++ Shared and SimpleShared Classes

`IceUtil::Shared` and `IceUtil::SimpleShared` are base classes that implement the reference-counting mechanism for smart pointers. The two classes provide identical interfaces; the difference between `Shared` and `SimpleShared` is that `SimpleShared` is not thread-safe and, therefore, can only be used if the corresponding class instances are accessed only by a single thread. (`SimpleShared` is marginally faster than `Shared` because it avoids the locking overhead that is incurred by `Shared`.)

The interface of `Shared` looks as follows. (Because `SimpleShared` has the same interface, we do not show it separately here.)

C++

```
class Shared {
public:
    Shared();
    Shared(const Shared&);
    virtual ~Shared();

    Shared& operator=(const Shared&);

    virtual void __incRef();
    virtual void __decRef();
    virtual int __getRef() const;
    virtual void __setNoDelete(bool);
};
```

The class maintains a reference that is initialized to zero by the constructor. `__incRef` increments the reference count and `__decRef` decrements it. If, during a call to `__decRef`, after decrementing the reference count, the reference count drops to zero, `__decRef` calls `delete this`, which causes the corresponding class instance to delete itself. The copy constructor increments the reference count of the copied instance, and the assignment operator increments the reference count of the source and decrements the reference count of the target.

The `__getRef` member function returns the value of the reference count and is useful mainly for debugging.

The `__setNoDelete` member function can be used to temporarily disable self-deletion and re-enable it again. This provides exception safety when you initialize a smart pointer with the `this` pointer of a class instance during construction.

To create a class that is reference-counted, you simply derive the class from `Shared` and define a smart pointer type for the class, for example:

C++

```
class MyClass : public IceUtil::Shared {
    // ...
};

typedef IceUtil::Handle<MyClass> MyClassPtr;
```

See Also

- The C++ Handle Template
- Smart Pointers for Classes