Syntax and restrictions

A **ref-qualifier** is an optional part of a nonstatic **member function** declaration. If present, it must come after any **cv-qualifiers** and before any **exception specification**. A constructor or **destructor** may not have a **ref-qualifier**:

```
void f1() &;  // Error, ref-qualifier on a nonmember function

class Class1
{
    // ...
public:
    Class1() &&;          // Error, ref-qualifier on constructor
    ~Class1() &;          // Error, ref-qualifier on destructor
    void mf() & const;    // Error, ref-qualifier before cv-qualification
    void mf() noexcept&;  // Error, ref-qualifier after exception specification
    void mf() & &&;       // Error, two ref-qualifiers
    static void smf() &;  // Error, ref-qualifier on static member function

    void mf(int) const && noexcept;  // OK, ref-qualifier correctly placed
};
```

A **member function** that does not have a **ref-qualifier** can be called for *either* an *lvalue* or an *rvalue*. Thus, C++03 code continues to compile and work as before:

```
class Class2 {
    // ...
public:
    void mf();
    void mf() const;
};

const Class2 makeConstClass2();

void f2()
{
          Class2 uobj;
    const Class2 cobj;

    uobj.mf();               // calls mf()
    cobj.mf();               // calls mf() const
    Class2().mf();           // calls mf()
    makeConstClass2().mf();  // calls mf() const
}
```