Document Number: J16/05-0051 = WG21 N1791
Reference Document: J16/04-0162 = WG21 N1722
Date: 2005-05-01
Reply to: William M. Miller
Edison Design Group, Inc.
wmm@edg.com

# Extended `friend` Declarations (Rev. 3)

## I. Introduction

The first two versions of this document (J16/03-0103 = WG21 N1520, J16/04-0056 = WG21 N1616) described the need to extend the current language to support a wider range of `friend` declarations and proposed specific wording to be incorporated into the C++ Standard. At the Redmond (October, 2004) meeting, the Evolution Working Group approved the proposal for recommendation to the Committee and referred it to the Core Language Working Group for review of the wording. Revision 2 (J16/04-0162 = WG21 N1722) incorporated a change to allow cv-qualified types to specified as friends.

Following the Redmond meeting, discussion among Committee members revealed that there were different understandings of the lookup to be applied to the names in extended `friend` declarations. According to 7.3.1.2¶3,

> When looking for a prior declaration of a class or a function declared as a `friend`, and when the name of the `friend` class or function is neither a qualified name nor a *template-id*, scopes outside the innermost enclosing namespace scope are not considered.

Some people (including the author) assumed that the *simple-type-specifier* form would naturally have the same meaning as the *elaborated-type-specifier* form. Because the two are so apparently similar, they felt that it would be confusing to treat them differently. Others noted that, unlike the *elaborated-type-specifier* form, the *simple-type-specifier* version cannot declare a new entity; it must refer to an existing declaration and is thus similar to the qualified name and *template-id* cases, for which the lookup is not restricted to the innermost enclosing namespace.

At the April, 2005 meeting in Lillehammer, the latter position was adopted, so that only the function and *elaborated-type-specifier* forms of `friend` declarations should have the restricted lookup. The wording below reflects that decision but is otherwise identical to the previous version of this paper.

## II. Proposed Wording

1. Change the last normative sentence of 7.3.1.2¶3 from:

> When looking for a prior declaration of a class or a function declared as a `friend`, and when the name of the `friend` class or function is neither a qualified name nor

a *template-id*, scopes outside the innermost enclosing namespace scope are not considered.

to

If the name in a `friend` declaration is neither qualified nor a *template-id* and the declaration is a function or an *elaborated-type-specifier*, the lookup to determine whether the entity has been previously declared shall not consider any scopes outside the innermost enclosing namespace. [*Note:* the other forms of `friend` declarations cannot declare a new member of the innermost enclosing namespace and thus follow the usual lookup rules. —*end note*]

2. Change 9.2¶7 from

The *member-declarator-list* can be omitted only after a *class-specifier*, an *enum-specifier*, or a *decl-specifier-seq* of the form `friend` *elaborated-type-specifier*.

to

The *member-declarator-list* can be omitted only after a *class-specifier* or an *enum-specifier* or in a `friend` declaration (11.4).

3. Delete the following wording from 11.4¶2:

An *elaborated-type-specifier* shall be used in a friend declaration for a class. [*Footnote:* The *class-key* of the *elaborated-type-specifier* is required.]

4. Add the following as a new paragraph following 11.4¶2:

A `friend` declaration that does not declare a function shall have one of the following forms:

`friend` *elaborated-type-specifier* `;`
`friend` *simple-type-specifier* `;`
`friend` *typename-specifier* `;`

[*Note:* a `friend` declaration may be the *declaration* in a *template-declaration* (clause 14, 14.5.3). ] If the type specifier in a `friend` declaration designates a (possibly cv-qualified) class type, that class is declared as a friend; otherwise, the `friend` declaration is ignored. [*Example:*

```
class C;
typedef C Ct;
```

```
class X1 {
    friend C;        // OK: class C is a friend
};

class X2 {
    friend Ct;       // OK: class C is a friend
    friend D;        // error: no type-name D in scope
    friend class D;  // OK: elaborated-type-specifier declares new class
};

template <typename T> class R {
    friend T;
};

R<C> rc;             // class C is a friend of R<C>
R<int> Ri;           // OK: "friend int;" is ignored
```

—*end example*]