Extended `friend` Declarations

The target of a **friend** *declaration* has been extended to allow designation of (1) a **type alias**, (2) a named **template parameter**, or (3) any previously declared **class type** such that if the target is not found the **friend** declaration will simply fail to compile rather than introduce a new declaration into the enclosing scope.

Description

A **friend** declaration located within a given **user-defined type** (UDT) grants a designated type (or *free* function) access to private and protected members of that class. Because the extended **friend** syntax does not affect *function* friendships, this feature section addresses extended friendship only between *types*.

Prior to C++11, the Standard required an *elaborated type specifier* to be provided after the **friend** keyword to designate some other *class* as being a **friend** of a given type. An elaborated type specifier for a class is a syntactical element having the form `<class|struct|union> <identifier>`. Elaborated type specifiers can be used to refer to a previously declared entity or to declare a new one, with the restriction that such an entity is one of **class**, **struct**, or **union**:

```
// C++03

struct S;
class C;
enum E { };

struct X0
{
    friend S;          // Error, not legal C++03
    friend struct S;   // OK, refers to S above
    friend class S;    // OK, refers to S above (might warn)
    friend class C;    // OK, refers to C above
    friend class C0;   // OK, declares C0 in X0's namespace
    friend union U0;   // OK, declares U0 in X0's namespace
    friend enum E;     // Error, enum cannot be a friend.
    friend enum E2;    // Error, enum cannot be forward-declared.
};
```

This restriction prevents other potentially useful entities, e.g., type aliases and template parameters, from being designated as friends: