Doc No: SC22/WG21/N1602=04-0042

Project: JTC1.22.32

Date: Thursday, February 12, 2004

Author: Francis Glassborow

email: francis@robinton.demon.co.uk

Class Scope Using Declarations & private Members.

1 The Problem

Currently it is not possible to use a class scope `using` declaration to inject an overload set from a base class into a derived class if that set contains any `private` members. This has an unfortunate side-effect on class design. It has the effect of determining what a base class designer may do in order to inhibit certain alternatives. E.g. in the following trivial case:

```
class base {
 void foo(long &);  // prevent potential narrowing
 int i;
public:
 void foo(int);     // must supply an int
};
class derived: public base{
public:
 using base::foo;
enum values{/*….*/};
void foo(values);
};
```

The compiler is required to issue a diagnostic for the `using`-declaration even if the user of the class never calls `foo()` with a `long`.

2 The Proposal

Change the place where the error regarding the private member manifests from the point of the `using`-declaration to the place where (if any) overload resolution selects the private member of the base class.

3 Discussion

This proposal has minimal effect on existing code (it changes code that is currently ill-formed to code that is well-formed). However it removes what appears to be a limitation on class designers. While I would have preferred a more comprehensive change to the way that class scope `using`-declarations work as regards access, this minimal change does solve most of the problems associated with the current specification.

One of the conceptual problems is that using declarations are about names but access is about members. 7.3.3 para 14 even refers to a *member name*. I do not think that there is actually any such concept as access to a member name. The correct place, in my opinion, to consider access is in regards to an individual member. Perhaps we need to add the

concept of 'uncallable in any context.' Actually, for the purposes of inhibiting certain potential calls that is not a bad idea. Perhaps we should allow `private private:` to have that meaning. There are several places where that would clean up code (e.g. ensuring non-copy semantics for a class).

Changes to the Working Paper

This proposal will require rewriting 7.3.3 paras 14 & 15 so as to remove the current constraint and replace it with a constraint on using a selected private member that has been provided via a `using`-declaration. Before word-smithing this proposed change I need to know if there really are serious implementation problems.