User’s Guide to SRTTI

1.0 RTTI (Run-Time Type Information.)

ANSI RTTI is supported on NT in VC++5 and HPPLUS in aCC. For the other platforms a package ***Simple RTTI*** exists and can be found in /afs/cern.ch/sw/lhcxx/specific/@sys/HISTOOGRAMS/pro/code/rtti.h for UNIX or Z:\P32\lhcxx\specific\windows_nt\histoograms\pro\code\rtti.h for NT.

2.0 USER’S GUIDE TO SRTTI: (Taken from rtti.h)

Assume that you have the following objects in your program:

```
TYPE1 &a = ...; // both TYPE1 and TYPE2 must be RTTI capable classes.

TYPE2 &b = ...; // see DEVELOPER’S GUIDE TO SRTTI for details.
```

- How to check if two objects are of the same class ?

```
1) rtti_same_class(a,b)
2) a.get_typeid() == b.get_typeid()
```

- How to check if an object is of the specified class ?

```
1) a.get_typeid() == TYPE2::class_typeid()
```

- How to retreive a class name (char *) ?

```
1) a.get_nameid()
2) TYPE1::class_nameid()
```

* ***IMPORTANT NOTICE:*** **since class names are bound statically at compile time to classes and typeid identifiers are bound dynamically at program startup, it is always safer to use class names if the RTTI info must be saved to persistent storage. It is particularily important if the same persistent info may be used on different compilers (different versions, different platforms). All possible effort has been made to ensure full portability but since compilers are ususaly bad, they do not neccesserily have to follow specifications of the standard.**

2.0.1 DEVELOPER’S GUIDE TO SRTTI:

- What to do to achieve RTTI capabilities for your class?

  1) derive your class from **SimpleRTTIObject** or any class that already has RTTI capabilities. YOU SHOULD USE **VIRTUAL** INHERITANCE if you derive from SimpleRTTIObject directly.

  2) in class declaration, put a statement **DECLARE_RTTI_CLASS(***class_name***)**

  3) in class definition file, put a statement **IMPLEMENT_RTTI_CLASS (***class_name***)**

- :----/header file: myclass.h/----

```
class MyClass : virtual public SimpleRTTIObject    {
  ...
  DECLARE_RTTI_CLASS(MyClass) // don’t put ; here

   // DO NOT  put semicolon after this statement
   // if you want to add lines after this statement you should explicitly
   // use access-specifiers (public/protected/private) otherwise access specifier
   // is undefined

public:
   // more code
};
```

---/implementation file: myclass.cpp/----

```
 IMPLEMENT_RTTI_CLASS(MyClass) // don’t put ; here   ...
```

2.0.2 DETAILED NOTES (What do DECLARE_RTTI_CLASS and IMPLEMENT_RTTI_CLASS do ?)

In other words: How to make a class RTTI-capable by hand?

- Each derived class should override virtual method get_typeid().
- It also must provide static method class_typeid()   (these methods should be overriden in the same way they are written for SimpleRTTIObject).
- The same work must be done for get_nameid() and class_nameid().
- It also must provide global or static variable, where type identifier will be stored, recommended implementation should mimic SimpleRTTIObject::type_identifier static data member.
- This MUST NOT be normal member (it MUST be created only once per class).
- This data member is of internal type SimpleRTTIObject::TypeIdentifier.
- In case of using OODBMS (Objectivity) this data mamber should be allocated in application data space and not as database persistent object.