



# DTSCPlus LIBRARY USE

Kent Sandvik/DTS

## INTRODUCTION

This kit contains a set of utility classes that could be used with any C++ related projects (MacApp, Bedrock, private libraries). Feel free to copy these, modify them, rewrite them, tune them, reimplement them, or do anything interesting with the code.

The core idea was that programmers want to tweak classes, and that's fine. There's no big framework, just loose classes that could be used wherever. More like components than classes tied to a huge architecture. This is the reason I created those collection classes as well, to be independent of any framework.

## STATE OF THE CODE

Well, it's sort of alpha-beta, many of the classes have been tested inside-out, however the classes have not been fully tested with various projects, so be careful.

## HOW TO USE THESE CLASSES

Look at the test source files, eg. UserTerminationTest.cp, for insight into how to use the classes.

## PERFORMANCE ISSUES

These classes were originally designed for Shared Library Manager use. This is the reason all member functions are virtual. It is also easier to make something for SLM, and if something is overhead, feel free to tweak it.

You might want to change any of the light-weight member functions into inline code, copy the code from the .cp file into the .h file.

## SYNTACTICAL ISSUES

The code conforms to MPW C++ coding (Miss Manners) standards. It is formatted using CDent (available on the developer CD).

Other interesting issues, why the use of goto:s, man? Well, the new style of class design is to have a defined entry and exit point -- for the sake of later debugging. So

sometimes it makes sense to jump to a common exit point. Maybe goto:s will have a come-back?

The iterator should all look the same, and behave the same way. That's an important aspect when designing classes. All destructors are virtual. If you won't use SLM, and you want to avoid vtables, get rid of the virtual statements.

### **DEBUGGING**

The classes use assertion macros quite a lot in order to catch sudden toolbox bugs, you might revise the way this is handled. In general each member function should return a Boolean if the execution went OK or not. In future exception handling will be perfect for this situation.

### **WHAT's THAT MINI-FRAMEWORK?**

It's an embryo to make a small simple C++ shell for quick prototyping. This shell might evolve in future to contain full AppleScripting support.

### **ANY OTHER QUESTIONS**

Well, the question is the answer. Long term plans might be to wire together such SLM C++ components using AppleScripting as the glue.

Kent Sandvik