Desktop Connection Library Programmer's Guide

by Paul Guyot

May 3, 2005

As long as there are ill-defined goals, bizarre bugs, and unrealistic schedules,
there will be Real Programmers willing to jump in and solve the problem,
saving the documentation for later.

Contents

1 What is the Desktop Connection Library?

The **Desktop Connection Library** or DCL for short is a framework of C++ and Obj-C classes to connect to Newton devices via the Dock/Connection built-in application and to manipulate Newton data.

The DCL can be used to either write general purpose connection and Newton-data handling software such as Newton package installers, Backup utilities, Synchronization tools, Data exchange programs, package creators and to add connection and/or Newton data handling capabilities to existing programs.

2 The Kallisys Reflexive License

The Desktop Connection Library is released under the **Kallisys Reflexive License** . This means that the DCL is open source and copy-left (any work based on the DCL should be open source). Additionally, the KRL introduces high quality standards regarding the code. The source code of any product based on the DCL should include, in readable form, a definition of the domains of functions, variables, classes and any element that can be referenced. Variables could be named with a single letter but in that case their full role should be explicited in comments.

Additionally, the KRL requires that the source code could be compiled with a freely available development environment and produce with it a working binary. This means that it rules out Excel Macros and REALBasic programs.

Please refer to the KRL for details.

The Desktop Connection Library is (mostly) commented in French, but this is not mandatory for works based on it of course.

3 The K Libs

The Desktop Connection Library uses general purpose libraries called the K Libs. These libraries are designed to be portable and some code there actually runs on NewtonOS. The DCL is provided with the K Libs. You are free to use (under the KRL) the K Libs independantly from the DCL.

- **K Crypto** includes a simple DES encryption and decryption class. This is very primitive, seems to work and is actually used by the DCL with a particular define to switch from regular DES to NewtonOS-compatible DES. Indeed, Apple implemented a DES-like challenge in the dock protocol.
- **K Defines** includes headers for portable definitions. You might need to modify this file (cf section Porting the DCL to another platform)
- **K GP** is a Genetic Programming framework. This is included in the K Libs for historical reasons and it is not used within the DCL.
- **K Math** includes 64 bits math classes based on 32 bits math classes. This is used here and there in the Desktop Connection Library.
- **K Misc** includes some miscellaneous utilities such as a Base64 class.
- **K Tests** is a set of debugging macros that can be used on various platforms including NewtonOS. These macros are used within the DCL.
- **K Unicode** includes classes to handle conversions between UTF-16 (a very similar encoding to NewtonOS's native unicode encoding[1]) and various encodings such as MacRoman, Latin-1 or UTF-8.

4 The structure of the DCL

The Desktop Connection Library is organized in 9 main parts.

4.1 Exceptions

Error within the DCL are propagated using exceptions. Exceptions include a code which is unique to each Exception class and, optionally, an error code describing the error (usually a platform-dependent code).

[1]NewtonOS handles an early version of Unicode defined when all characters fitted on 16 bits

4.2 Streams

The streams are an abstraction for input/output operations with files and pipes. TDCLStream abstract class, through a couple of methods that subclasses need to override, provide various input/output operations. Some streams like files can be rewinded, they are subclasses of TDCLRandomAccessStream. TDCLStdStream and TDCLMemStream are two utility streams used in sample codes and in tests.

4.3 Interfaces

Interfaces between the DCL, the application and the OS is provided through Interfaces classes. Several classes provide an abstraction for files, threads and system utilities and implementation for various OSes. TDCLApplication is the base class for the interface between the DCL and the application and it includes a wide set of callbacks for connection-related operations.

4.4 Communication Layers

The communication layers are responsible for handling the communication with the Newton. This can be more than just the physical layer, for there are communication layers doing TCP/IP.

A communication method is represented with a TDCLCommLayer class. For example, there is a class for AppleTalk connections on MacOS X or for BSD Sockets.

Connections themselves are represented with objects of the TDCLPipe class which derive from TDCLStream.

Two main kind of communication layers are supported. Asynchronous (or interrupt-based) communication layers should derive from TDCLCommLayer. Synchronous (or thread-based) communication layers should derive from TDCLSyncCommLayer. The DCL comes with layers for various connection methods and for various OSes such as AppleTalk on MacOS X, BSD Sockets and serial on POSIX platforms.

4.5 Servers

Server objects are built just on top of the Communication Layers. They are responsible for handling one or several communication layers and dispatching connections to Links.

One can define a class deriving from TDCLServer for custom server behavior. However, TDCLSimpleServer and TDCLOneLinkServer should be sufficient for most uses. TDCLSimpleServer is a server with a single Communication Layer and a single Link (typically like Newton Package Installer). TDCLOneLinkServer is a server with one link but several communication layers (for applications, like Newton Connection Utilities, listening on several interfaces).

4.6 Links

The links represent the application interface with the Newton. The minimum connection utility should define its own link.

Several base class links are available for various amounts of features. TDCLDockLink provides the minimum set of features and can be used with applications not willing to provide any standard service. TDCLLoadPackageLink is a simple link to handle challenge-less Newton Package Installer-like connections with the Newton. The only possible operation is loading a package. TDCLFullDockLink class provides the full set of standard connection features through link engines. The set of engines can be customized.

4.7 NewtonScript Objects

Classes in the NS Objects package are either classes for NewtonScript objects such as frames, arrays, binaries, strings, symbols or immediates or classes for encoding and decoding of these objects in various formats (XML, NSOF, Packages, Text).

4.8 Packages

TDCLPackage is a class for a Newton package. It allows analysis and compilation of packages through various classes for the package parts (subclasses of TDCLPkgPart).

4.9 Data

Classes in the data directory are specialized in the analysis and the conversion of Newton data from/to desktop data such as conversion of RTF rich text to NewtWorks or clParagraph formats.

5 Writing an application based on the DCL

6 Porting the DCL to another platform

7 Writing a communication layer

8 Adding more exchange capabilities

9 References

10 Contact