Windows Sockets TIPS

Thank-you for purchasing the GCP++ TCP/IP SDK for Windows, C/C++ Edition. GENISYS Comm has compiled a wide range of documentation that you may find useful for your development task. This SDKcan be used to create TCP/IP applications by making direct function calls to the WINSOCK.DLL library.

This document is organized into 2 sections:

Section 1 describes the files included in this distribution
Section 2 discusses WINSOCK interoperability
Section 3 summarizes the steps required to write a WINSOCK application

Section 1: Files included in this distribution

doc\wsguide.doc and doc\wsguide.txt

This document, authored by the founders of the Windows Sockets specification provides an excellent overview of the standard and serves as a useful introduction to WINSOCK and it's value.

doc\winsock.hlp

This is a windows help file that documents the complete Windows Sockets v 1.1 specification. This is the source document for the standard.

doc\faq.txt

This "Frequently Asked Questions" document has been developed by several of the vendors and responds to many common questions.

doc\tips.wri

This document, provided by GENISYS Comm, describes the software and documentation included in the GCP++ TCP/IP SDK for Windows, C/C++ Edition.

winsock\cwinsock.zip

GENISYS has defined the CWINSOCK class that wraps the WINSOCK DLL calls inside a C++ object. Written in Visual C++, a sample application is included. MFC is used as a foundation. Create your own speciallized C++ classes by deriving them from CWINSOCK.

winsock\winsock.h

This is the header file that must be included within your WINSOCK application code.

winsock\winsock.lib

This is an import library for the WINSOCK specification. It can be recreated using the imlib.exe utility from any WINSOCK.DLL library.

winsock\network.cpp network.hpp

These files decribe two C++ classes, UDP_SOCK and TCP_SOCK, that GENISYS developed for their GCP++ server product. It has been modified slightly to eliminate it's dependency on other classes in the GCP++ server product. These classes may be used as a starting point for the development of your own.

winsock\pkt.cpp pkt.hpp

This class was defined by GENISYS to provide different containers that could be passed to UDP_SOCK and TCP_SOCK object. They contain state information that allows the packets to be filled or emptied under the control of an external procedure.

winsock\ws_ftp.zip

Mr. John Junod has released this (partial) FTP application into the public domain, and it is included as an example forthe reader.

winsock\wsmtpd15.zip

Mr. Ian C. Blenke has released this SMTP daemon into the public domain, and it is included as an example for the reader.

Section 2: WINSOCK Interoperability

The Windows Sockets Specification defines the interface between applications and TCP/IP stack vendors. Consequently, there exists the potential for interoperability problems, where application X may work on some stacks but not all. For example, GENISYS developed the GCP++ Server which binds directly to any underlying WINSOCK stack. Since our customers have tested it over numerous stacks, we have been placed in the position of filing numerous bug reports with the kernel vendors.

The moral of this story is to expect interoperability problems with your software if you utilize any of the less common functions. We expect this risk to decline with time, however, as TCP/IP stack code matures for each vendor.

Section 3: Writing a WINSOCK App

Writing a WINSOCK application is not easy because it requires a good understanding of the communications behavior before you can even start. This is why GENISYS offers the GCP++ Server, which provides such a high-level API.

Assuming that the GCP++ Server is not suitable, the following steps need to be taken in the design of your TCP/IP sub-system:

1. Review all the included written material to gain an understanding of what TCP/IP communications is about. Several excellent textbook are available like Douglas E. Comer's "Internetworking with TCP/IP" and W. Richard Steven's "UNIX Network Programming".

2. Will your communications be synchronous blocking, synchronous non-blocking, or asynchronous? It is recommended by the author and most Windows programmers that asynchronous functionality be used. This mechanism may seem to be more time consuming, but it will provide better results down the road.

3. Is stream I/O (TCP) or datagram I/O (UDP) to be used?

4. Are your WINSOCK calls to be hidden within a DLL or encapsulated within a C++ object? The included network.cpp and CWINSOCK.ZIP illustrates how we have used C++ to encapsulate the code. Alternatively, some developers use a DLL to localize the network code. The GCP++ Server application takes this to one extreme by locating the code in a separate application entirely.

5. How will your network code communicate with other sub-systems? We have found it most logical to

encapsulate the WINSOCK calls within C++ objects, and then to make the C++ object part of a bigger object that includes a window for asynchronous notification. In this manner the C++ network object can be polled in response to queues that arrive at the larger object's window. We call the larger object an "Agent".

Next, code and test your system. We have been very pleased with Visual C++, although we identified some problems with the IDE debugger, where the breakpoint mechanism sometimes fails. During testing, we usually use a loopback mechanism that greatly simplifies the procedure (in other words, perform connections to yourself). Most protocols can be tested using a single workstation.

Again, thankyou for purchasing this GCP++ product. Please contact tech support at GCP++@GENISYS.com if you experience any difficulties using our product.

GENISYS also provides consulting services to help you complete on-time within budget!