INTRODUCTION

Winsock RCP32.DLL is a Microsoft Win32 Dynamic Link Library (DLL) that provides a Windows Sockets compatible function that allows you to copy files over TCP/IP using the ***rcp*** protocol. You can copy files from a Windows PC to a remote host, from the remote host to your PC, or between two remote hosts.

Winsock RCP32.DLL provides most of the services of the *rcp* command found on many Unix systems, including recursive copies. It also includes the ability to convert text files to the appropriate format for Unix or Windows.

The remote host must be a system running a Remote Shell Daemon (*rshd*) service that supports rcp, such as a Unix system or a PC running Denicomp Systems Winsock RSHD/95 or Winsock RSHD/NT.

REQUIREMENTS

Winsock RCP32.DLL requires a PC running any 32-bit Windows operating system, such as Windows NT/2000/XP/2003 or Windows 95/98/ME or any other Windows operating system that supports 32-bit Windows DLL's, a connection via TCP/IP to a remote host running ***rshd***, and any programming language that supports calls to 32-bit DLL functions, such as C and Visual Basic.

INSTALLATION

To install Winsock RCP32.DLL from a CD-ROM, insert the CD and wait for the installer window to appear. If you do not see it appear within several seconds, browse the CD and double click on the SETUP file.

Specify the directory where you want to install the files. After installation, to develop and run applications, you will need to copy the file **RCP32.DLL** to a directory in your PATH environment variable, the same directory as your executable, or your Windows directory (usually \WINNT or \WINDOWS) or Windows System directory (such as \WINNT\SYSTEM32).

The directory you selected will contain the file **RCP32.LIB**, which is the import library for RCP32.DLL. C programmers can link this file to your programs to use RCP32.DLL. You may move this file to the directory containing your other C libraries if you wish.

This directory will also contain the files **RCP32.H** and **RCPERRS.H**, which are C header files containing the function declarations for Winsock RCP32.DLL and constants that define error values returned. C programmers will "#include" these files in programs. You may move these files to a directory your compiler searches for header files if you wish.

If you are using Microsoft C++, you will need to enclose the #include of RCP32.H within an _extern "C" { } block.

DISTRIBUTION

Winsock RCP32.DLL is licensed per developer. That is, each developer (programmer) that uses it to develop applications must have a license for their system. You may distribute the RCP32.DLL file with your application that uses it royalty-free to end-users of your application if you have the appropriate developer license(s). You may distribute the RCP32.DLL file *only* with your application.

Note that per the license agreement, you may **not** distribute RCP32.DLL royalty-free if your program is simply a command line interface to RCP32.DLL or provides the end user with an Application Programmable Interface (API) into RCP32.DLL.

SAMPLE APPLICATION

The Winsock RCP32.DLL distribution includes two sample programs: VRCP, which is an interactive Visual Basic sample program, and CRCP, which is an example program written using Microsoft Visual C/C++.

To view the Visual Basic example, load the file VRCP.MAK with Visual Basic.

The CRCP Makefile for the example C program is named CRCP.MAK. It is set up to look in the current directory for the RCP32.H and RCP32.LIB files. You must modify it if you have moved these files. To compile the example program using Microsoft Visual C/C++, type the following at a Command Prompt:

```
NMAKE /F CRCP.MAK CRCP.EXE
```

The executable version of this example program is also included, so you do not need to compile it if you prefer.

Once compiled, the program functions as a scaled down RCP command. It is a "console" program and its command line syntax is:

CRCP [-r][-a][-c][-h] source dest

Where:

-r	Recursively copy "source" to "dest". Copies all files and subdirectories in "source" to "dest". "dest" must be a directory.
-a	Perform ASCII end-of-line conversions. Otherwise, contents of files are not modified.
-c	Preserve the case of filenames when using wildcards or recursive copies. Otherwise, names are converted to lowercase characters. When copying individual files, the case specified on the command line is used.
-h	When using wildcards or -r, normally hidden files are not copied. If you specify -h, hidden files will be copied.
-s	If a filename contains spaces, an underscore (_) will be substituted for each space in the destination filename on the remote host.

Do not combine command line options; separate each with a space. For example, use "-r -c", not "-rc".

The "source" and "dest" parameters either specify a file or directory on the local system or a file or directory on a remote host. At least one of these must specify a remote host.

When specifying a file/directory on a remote host, the "source" and/or "dest" parameters must be in the following format:

[user@]host:file **or** [user@]host:directory

Wildcards may be used. However, only one source and one destination can be specified. That is, you cannot specify multiple source files by separating the names with spaces as you can with the standard RCP command.

Licensing Note: CRCP is included as a sample only. The Winsock RCP32.DLL license does not permit royalty-free distribution of the DLL if your program is simply a command line interface into RCP32.DLL.

FUNCTION DESCRIPTIONS

FUNCTION: WinsockRCP

C:

```
int WinsockRCP(HWND hwndParent,
               char *lpTitle,
               char *lpSrc,
               char *lpDest,
               int iRecurFlag,
               int iAsciiFlag,
               int iConsoleFlag,
               int iCaseFlag,
               int iHiddenFlag,
               int iSpaceChar)
```

Visual Basic:

```
Declare Function WinsockRCP Lib "RCP32.DLL"
    (ByVal hWndParent As Long,
     ByVal lpTitle As String,
     ByVal lpSrc As String,
     ByVal lpDest As String,
     ByVal iRecurFlag As Long,
     ByVal iAsciiFlag As Long,
     ByVal iConsoleFlag As Long,
     ByVal iCaseFlag As Long,
     ByVal iHiddenFlag As Long,
     ByVal iSpaceChar As Long) As Long
```

Usage

The ***WinsockRCP*** function copies the Source (***lpSrc***) to the Destination (***lpDest***). The Source and Destinations are file specifications that can include a user name, a host name, and a file name, which may include wildcard characters, such as * and ?.

If any errors occur during the transfer, they will be displayed in standard Windows message boxes as they occur. See the ***WinsockRCP2*** function if you wish to handle error messages in your program.

Parameters

hwndParent:	The handle (HWND) of a window in your application. The ***WinsockRCP*** function may produce diagnostic error messages using the Windows *MessageBox* function. This is used to specify the "parent" window to the MessageBox function. It may be NULL. **Note:** If your program is Windows console application, you should specify NULL for this parameter and see the ***iConsoleFlag*** parameter for details about displaying error messages to the Console window.
lpTitle:	The title of your application or some other title. If any error messages are displayed using the Windows *MessageBox* function, this will be used as the title of the message box.

lpSrc:	The file or directory to copy. See below for the format.
lpDest:	The destination of the file(s). See below for the format.
iRecurFlag:	Specifies how directories should be copied. The options are: **0 -** Directories will not be copied. If you attempt to copy a directory, you will receive an error message. This occurs if you explicitly try to copy a directory by specifying its name or directory names are matched by the wildcard pattern you specify. If you specify a wildcard pattern that matches both files and directories, the files will be copied, but you will receive an error message for each directory matched. **1 -** Directories will be recursively copied. If you explicitly copy a directory by specifying its name, all files in that directory and the files in all subdirectories of that directory will be copied. The directory structure will be recreated on the destination system. If you specify a wildcard pattern, all directories matching that pattern will be recursively copied in addition to any matching files. **-1 -** Specifying -1 (negative one) will **not** recursively copy directories, but it will suppress any error messages if your wildcard pattern matches directories. That is, your wildcard pattern will match only **files**, not directories. This is similar to specifying 0 (zero), but you will not receive error messages if you attempt to copy directories.
iAsciiFlag:	Specifies whether or not the copy should convert the file to the proper text file format. If the file is being copied ***to*** a remote host, CR/NL (ASCII 13/10) combinations are replaced with one NL (ASCII 10). If the file is being copied ***from*** a remote host, NL's are replaced by CR/NL combinations. Use a value of 0 if the files are not to be converted or a value of 1 if they are to be converted.
iConsoleFlag:	iSpecifies whether or not your application is a Windows console application. If you specify a value of 0, error messages will be displayed in standard Windows message boxes. If you specify a value of 1, error messages will be displayed to the console window using the "printf" function.
iCaseFlag:	Specifies whether or not the case of filenames is preserved when performing recursive copies or using wildcards. This affects recursive and wildcard copies **only**. If you use a specific, individual filename, the case you use will be preserved. Windows filenames can be stored using mixed case characters, even though the filesystem is **not** case sensitive. For example, "ABC" and "abc" represent the same file. However, the directory entry is displayed in the case you used when the file was created. If you specify a value of 0, the case of filenames will **not** be preserved. All filenames will be converted to lower case characters as they are received from or sent to the host. If you specify a value of 1, the case of filenames **is** preserved. Filenames will be created on the local system in the same case as they existed on the remote host. Likewise, filenames on the local system will be created in the same case on the remote host as they appear in the directory on the local system.
iHiddenFlag:	Specifies whether or not files with the "hidden" attribute are copied when doing recursive copies or using wildcards. A value of 0 means hidden files will not be copied; a value of 1 means hidden files will be copied.

iSpaceChar:	Allows you to substitute a character for space characters (ASCII 32) in filenames.
	
	If you specify a value of 0, filenames are sent to the remote host exactly as they appear on the local system.
	
	However, some systems cannot easily handle filenames with embedded spaces. If you pass a character in this parameter (as an integer value), that character will be substituted in each filename on the remote host for each space.
	
	For example, if you pass an underscore (_) in this parameter, the filename:
	
	`"this is a file.txt"`
	
	will be created on the remote host as:
	
	`"this_is_a_file.txt"`

Source and Destination Specifications

Both the ***lpSrc*** and ***lpDest*** parameters must be of the following format:

[[User@][Host:]]{File | Dir}

User@

(optional) Specifies the user name to be used at the remote host. If this prefixes the Host: parameter, this user name overrides the user name obtained from the local system when it is sent to the *rshd*.

Specifying a Different Local and Remote User

The rcp transfers normally occur through the *rshd* protocol. The *rsh* protocol allows **two** user logins to be sent to the server: a "local" user and a "remote" user. The standard Unix *rshd* validates access for both users, but accesses files in the security context of the "remote" user.

When you use the ***User@*** construct in the file specification, it sends "User" as both the local and remote users. If you wish to send different users for the local and remote users for the *rsh* protocol, specify them in the format ***localuser!remoteuser@***. That is, separate them with an exclamation point (!) and RCP32.DLL will send the two different users.

Using the Remote Exec Daemon

Normally, rcp operates through the Remote Shell Daemon (*rshd*) on the server. However, with some systems (most Unix systems and Denicomp's Winsock REXECD/NT), you can have the rcp operate through the Remote Exec Daemon (*rexecd*). The difference is that the Remote Exec Daemon requires you to specify a password for each connection.

If you want to use the *rexec* protocol for the rcp, you can do it with function calls (explained later – see the ***WinsockRCPSetIntOption*** function) or you can do it by specifying the user name in the source or destination in the format ***user/password@***. That is, separate the user and the password with a slash (/).

When the RCP32.DLL functions see the slash as part of the user name, it switches to using the *rexec* protocol and sends the user and password as you specified them.

Host

Specifies the TCP/IP host name of the remote host when referring to files on the remote host. This host must be a system running the *rshd* service. It may be an IP address instead of a host name.

NOTE: Do **not** use a host name when referencing files on the local system. If you do, RCP32.DLL will expect that the local system is running a Remote Shell Daemon and may have other ramifications. Even if the local system is running an *rshd*, referring to local files this way is *much* less efficient and may not even work, depending on the setup of the local system. (See the note below about specifying the host in both the source and destination).

File

Specifies the file name of the source or destination file. You may use wildcard characters to copy multiple source files.

Dir

Specifies the file name of the source or destination directory. The source may be a directory only if you are using the recursive copy option. The destination must be a directory if you are copying multiple files or copying recursively.

Specifying a Host for Both Source and Destination

If you specify a host name for *both* the source and destination filenames, the rcp standard (the Unix rcp and RCP32.DLL) is to treat it as a "pass through" rcp. That is, rcp simply passes through the rcp command to the source system and tells it to perform the rcp. It does this using the *rsh* protocol

This can cause problems if the source system does not have an rcp command (or it is non-standard) or is not running a Remote Shell Daemon.

For example, if you specify:

```
Source:       systemA:/temp/file1.txt
Destination:  systemB:/temp/file2.txt
```

Internally, RCP32.DLL executes an *rsh* command like this:

```
rsh systemA "rcp /temp/file1.txt systemB:/temp/file2.txt"
```

The result of this is what is expected - /temp/file1.txt is copied from systemA to systemB. However, if systemA is not running a Remote Shell Daemon or the rcp command does not exist on SystemA or it has non-standard syntax, this may fail.

It is for this reason that you should **never** refer to local files with a host name if you cannot guarantee that the local system is running a compatible Remote Shell Daemon. (It is also much less efficient because it generates two network connections.)

Return Value

If ***WinsockRCP*** successfully copies all specified files, it will return a positive value that denotes the total number of files copied. Note that this number includes only individual **files** copied, not directory names created through recursive copies.

However, if you copy files between two remote hosts (you specify a host name in the source and destination, resulting in no files copied to or from the local PC), a success will return **zero**, not the number of files copied.

If ***WinsockRCP*** is not successful, it will return a negative number indicating the **last** error that occurred. A list of these error numbers is provided later. Note that if multiple files are to be copied, it will return a negative value if an error occurs on *any* of the transfers. ***WinsockRCP*** may have successfully copied files prior to the error and depending on the problem, may continue to copy the remaining files. If you are copying multiple files, the RCP protocol does not stop copying files when an error occurs unless it is a network error.

Notes:

A ***Host:*** *must* be specified for either the ***lpSrc*** or ***lpDest*** parameters, or both.

If a full directory path is not specified for a remote host, the path begins at the user's home directory. That is, if the file/directory name specified after the ***Host:*** parameter does not begin with a slash (/), it is assumed to reference a file/directory in the user's home directory.

For example, the file "joe@remhost:file.txt" refers to the file "file.txt" in the home directory of the user "joe" on the host "remhost". The method of determining the “home directory” is with the Remote Shell Daemon and/or operating system at the other end of the connection.

Filenames may contain either slashes (/) or backslashes (\) as directory separators, for either the host file/directory or file/directories on the PC. They will be converted to the appropriate separator.

You can copy multiple files by using wildcard characters, such as * or ?. The wildcard capabilities are determined by the operating system running where the source files are located. For example, if you are copying files from a Unix host, you can use any of the wildcard options available on Unix to match the filenames. If the source of the files is the PC, you are limited to the wildcard options available in Windows.

The ***lpSrc*** and ***lpDest*** parameters may include only **one** file specification each. That is, you **cannot** specify multiple source files by separating them with spaces as you can with the *rcp* command. To copy multiple individual filenames (that cannot be specified using wildcards or using a recursive copy), you must call **WinsockRCP** for each file or use the ***WinsockArrRCP*** or ***WinsockArrRCP2*** functions.

If you copy multiple source files with wildcard characters, the destination the destination must be a directory.

Note that a colon (:) terminates the host name. This causes a problem when filenames on the PC require a drive letter (for example, A:). If a file name specification begins with one character between A and Z and is followed by a colon (:), ***WinsockRCP*** will interpret this as a drive letter instead of a host name. This means that ***WinsockRCP*** cannot handle one character host names.

The destination cannot contain only a drive specification (for example,. A:). It must also include a filename or a directory name. If the destination is the current directory on the drive, use "." (for example,. A:.).

Using the ASCII conversion option to transfer files **to** the remote host will slow the copy operation somewhat because it must read each file *twice*. It reads the file once to calculate the new translated file size, then reads it again to transfer the data. This is because the RCP protocol requires that the exact file size be transmitted before

the actual data in the file is sent. Without the conversion option, the file size can be found by examining the file's directory entry, but with the conversion option, the file's contents must be examined to determine the size after CR/NL combinations are replaced with NL.

The ASCII conversion option will also slow Winsock RCP when transferring files **from** the remote host, but only slightly, because it must examine the data and translate it before writing it to the file.
If transmission speed is critical, consider using utilities to translate the text files after they are transferred.

Security

The local user name determines the file access privileges ***WinsockRCP*** uses at the remote host. This name also determines the ownership and access modes of the destination file or files.

If you specify a user name in either the source or destination filenames using the "user@host" construct, that name will be used at the remote host.

If no user is specified for the source or destination along with the host name, the current Windows user is used. This is the user name you used when logging in to Windows. For example, if you logged in to Windows as the user "fred", ***WinsockRCP*** will use "fred" as the user name at the remote host. ***WinsockRCP*** will always convert this name to all lowercase characters.

RCP32.DLL provides two methods for overriding the user it uses when no user is specified in the source or destination parameters, if you do not want RCP32.DLL to use the current Windows login.

The first method is to use the ***WinsockRCPSetStrOption*** call to set the "User" option to the desired remote user. This function is described in more detail in its own section later. Your program would need to implement a method for determining the appropriate user to specify in this call.

The second method is to have RCP32.DLL get the user override from the file **WIN.INI** in the Windows directory (e.g. \WINNT or \WINDOWS) for an alternate user name.

If **WIN.INI** contains a section named **[RCP]** and contains an entry named **User** in that section, the name specified there will be used as the local user name. For example, **WIN.INI** might contain:

```
[RCP]
User=joe
```

If this appeared in **WIN.INI**, the local user name would be "joe" and ***WinsockRCP*** would use this name at the remote host.

To support multiple users, ***WinsockRCP*** will also look for a section named **[RCP-user]** in **WIN.INI** first for an alternate user name, where the "user" in the section name is the name used to log in to Windows. ***WinsockRCP*** will look at this section first; if it does not exist, it will then look at the **[RCP]** section.

For example, if "MaryJones" and "JohnSmith" are both users on the Windows PC, but their user names at the remote host are "mary" and "joe" respectively, **WIN.INI** might look like this:

```
[RCP-JohnSmith]
User=john

[RCP-MaryJones]
User=mary
```

When the Windows user "JohnSmith" runs a program using ***WinsockRCP***, "john" will be used at the remote host. When the Windows user "MaryJones" runs the program, "mary" will be used instead.

Unix Security

If the remote host is running the Unix operating system, it allows access through ***WinsockRCP*** if one of the following conditions is satisfied:

- ?? The Unix system can obtain a name for the IP address of the system calling ***WinsockRCP***.
- ?? That name is listed as an equivalent host in the /etc/hosts.equiv file on the remote host.
- ?? If the name is not in the /etc/hosts.equiv file, the user's home directory on the remote host must contain a .rhosts file that lists the name that name.
- ?? If the .rhosts file is used, it must be owned by either the user specified or "root", and only the owner should have read and write access. That is, it must have permissions of 0600.

These rules are determined by the Remote Shell Daemon (rshd), not Winsock RCP32.DLL. If the source or destination system is not a Unix system, the rshd on that system may have other rules and requirements.

Example

The CRCP program included with the distribution shows a full working example of the ***WinsockRCP2***() function call. It is similar to the ***WinsockRCP***() function call illustrated here:

```
// This example copies the file "system.doc" in the user tom's home
// directory on the host "remhost" to the directory "\doc" on the PC

int result;

char *rtitle = "Winsock RCP32.DLL Sample";
char *src = "tom@remhost:system.doc"
char *dest = "/doc"

result = WinsockRCP(NULL,rtitle,src,dest,0,0,1,0,0,0);

if (result < 0)
  MessageBox(NULL,"Remote Copy Was Not Successful",rtitle,MB_OK);
else
  MessageBox(NULL,"Remote Copy Was Successful",rtitle,MB_OK);
```

FUNCTION: WinsockRCP2

C:

```
int WinsockRCP2 (char *lpSrc,
                 char *lpDest,
                 int iRecurFlag,
                 int iAsciiFlag,
                 int iCaseFlag,
                 int iHiddenFlag,
                 int iSpaceChar,
                 char *lpErrMsg,
                 int iErrLen)
```

Visual Basic:

```
Declare Function WinsockRCP2 Lib "RCP32.DLL"
    (ByVal lpSrc As String,
     ByVal lpDest As String,
     ByVal iRecurFlag As Long,
     ByVal iAsciiFlag As Long,
     ByVal iCaseFlag As Long,
     ByVal iHiddenFlag As Long,
     ByVal iSpaceChar As Long,
     ByVal lpErrMsg As String,
     ByVal iErrLen As Long) As Long
```

Usage

The ***WinsockRCP2*** function performs the same function as the ***WinsockRCP*** function, but does not display error messages in Windows message boxes or to a console window. Instead, it returns the error messages in the area you provide. You can then display them, ignore them, or process them as you require.

Parameters

lpSrc	The file or directory to copy. See ***WinsockRCP*** for details.
lpDest	The destination of the file(s). See ***WinsockRCP*** for details.
iRecurFlag	Specifies how directories should be copied. See ***WinsockRCP*** for details.
iAsciiFlag	Specifies whether or not the copy should convert the file to the proper text file format. See ***WinsockRCP*** for details.
iCaseFlag	Specifies whether or not the case of filenames will be preserved in recursive or wildcard copies to a remote host. See ***WinsockRCP*** for details.
iHiddenFlag	Specifies whether or not hidden files will be copied. See ***WinsockRCP*** for details.
iSpaceChar	Allows you to substitute a character for space characters in filenames. See ***WinsockRCP*** for details.
lpErrMsg	A pointer to an area of memory where any error messages generated during the copy can be stored. It **cannot** be NULL. This is explained in more detail later.
iErrLen	The total size in bytes of ***ErrMsg***. This must be greater than or equal to 1.

This function operates in the same manner as ***WinsockRCP***, but if any errors occur during the transfer, the error messages will be accumulated in ***lpErrMsg***. Unlike ***WinsockRCP***, they will not be displayed in message boxes or to a Console window. When the function returns, if the return value is negative, the text of any error messages will be stored in ***lpErrMsg***.

Note that multiple errors can occur, since the RCP protocol does not necessarily stop transferring files on certain errors. If multiple errors do occur, each error message will be stored in ***lpErrMsg*** and they will be separated by newline characters (ASCII 10). Hence, ***lpErrMsg*** will be suitable for use with the Windows MessageBox function if you desire.

For example, if you are copying a number of files and you receive two "Permission denied" errors, ***lpErrMsg*** may contain (using C notation): "Permission denied\nPermission denied" where "\n" is the newline character.

lpErrMsg should be large enough to hold at least a few error messages. ***WinsockRCP2*** will not exceed the size of ***lpErrMsg*** as specified by the ***iErrLen*** parameter. If the area is not large enough to hold an error message, it will not be stored. It is possible that the text of all error messages will not be stored in ***lpErrMsg*** if a large number of errors occur.

The size of ***lpErrMsg*** (and the value specified for ***iErrLen***) must be greater than zero. If you specify a very small value for ***iErrLen***, no error text will ever be stored in ***lpErrMsg*** since no message will fit, but return values will still be negative if errors occur.

Declaring WinsockRCP2 in Other Languages

If you are using a language other than C and Visual Basic, please keep in mind that the area of memory pointed to by the ***lpErrMsg*** parameter in the ***WinsockRCP2*** function call is modified by the DLL.

Be sure when creating declarations in your language for the RCP32.DLL functions that you keep this in mind. Some languages require that you specify which parameters passed to DLL's will be modified and should be passed as a pointer, not as a copy of the variable value in the program. You may receive errors if you do not do this.

Using WinsockRCP2 in Languages with Dynamically Allocated Strings

For those using languages with dynamically allocated strings, such as Visual Basic, you must *allocate space* in the ***lpErrMsg*** parameter before calling ***WinsockRCP2***. Your program will most likely abort if you do not.

Normally, when you declare a string in one of these languages, no memory is allocated for the variable until you store data in it. If you attempt to use the string variable without filling it with some value to force the language to allocate memory for it, the pointer passed may not be valid.

In Visual Basic, this is normally done with the **String$** function. For example:

```
Dim emsg As String

emsg = String$(256,Chr$(0))
```

This will store 256 Chr$(0) values (the null character) in the **emsg** variable. This will force Visual Basic to allocate 256 bytes of memory for **emsg**. When you pass **emsg** to ***WinsockRCP2***, there will be 256 bytes available to store error messages.

In Powerbuilder, this is normally done with the **Space** function:

```
String emsg

emsg = Space(256)
```

This will fill the **emsg** variable with 256 spaces, and therefore allocate 256 bytes for the variable.

Return Value

If ***WinsockRCP2*** successfully copies all specified files, it will return a positive value that denotes the total number of files copied. Note that this number includes only individual **files** copied, not directory names created through recursive copies.

However, if you copy files between two remote hosts (you specify a host name in the source and destination, resulting in no files copied to or from the local PC), a success will return **zero**, not the number of files copied.

If ***WinsockRCP2*** is not successful, it will return a negative number indicating the **last** error that occurred. A list of these error numbers is provided later. Note that if multiple files are to be copied, it will return a negative value if an error occurs on *any* of the transfers. ***WinsockRCP2*** may have successfully copied files prior to the error and depending on the problem, may continue to copy the remaining files. If you are copying multiple files, the RCP protocol does not stop copying files when an error occurs unless it is a network error.

Example

For a full working example of the ***WinsockRCP2*** function in C, see the CRCP program included in the distribution.

```
// This example copies the file "system.doc" in the user tom's home
// directory on the host "remhost" to the directory "\doc" on the PC
// If any error occurs, we will display it.

int result;

char *src = "tom@remhost:system.doc"
char *dest = "/doc"
char errmsg[256];

result = WinsockRCP2(rtitle,src,dest,0,0,0,0,0,errmsg,sizeof(errmsg));

if (result < 0)
  MessageBox(NULL,errmsg,"RCP Error!",MB_OK);
```

FUNCTION: WinsockArrRCP

C:

```
int WinsockArrRCP (HWND hwndParent,
                   char *lpTitle,
                   char **lpArgArr,
                   int iArgCount,
                   int iRecurFlag,
                   int iAsciiFlag,
                   int iConsoleFlag,
                   int iCaseFlag,
                   int iHiddenFlag,
                   int iSpaceChar)
```

Usage

The ***WinsockArrRCP*** function is similar to the ***WinsockRCP*** function, but accepts a vector (array) of strings containing the source and destination filenames. This allows you to copy multiple source files/directories in a single call when the files cannot be named using a single wildcard pattern or recursive copy.

Any errors that occur are displayed in standard Windows message boxes or to the console window. See the ***WinsockArrRCP2*** function if you wish to handle the displaying of error messages within your program.

The strings in the array should be filenames of the format:

[[User@][Host:]]{File | Dir}

The **last** string in the array will be considered the **destination**.

The array must contain at least **two** strings (a source and a destination).

Parameters

The parameters to the ***WinsockArrRCP*** function are the same as those passed to the ***WinsockRCP*** function, except for the ***lpSrc*** and ***lpDest*** parameters.

WinsockArrRCP substitutes the following parameters for ***lpSrc*** and ***lpDest***:

lpArgArr	Pointer to an array of strings. This array must contain two or more strings. The **last** string in the array will be considered the destination of the copy.
iArgCount	The number of strings in the array. This is a number greater than or equal to two (2).

Notes

The same rules for ***WinsockRCP*** apply to this function. Refer to that section. However, here are a few important points:

A ***Host:*** must be specified for either the source files/directories or the destination file/directory (or both). The destination is always the **last** string in the array.

If you copy multiple source files, the destination the destination must be a directory.

Important Note:

The ***WinsockArrRCP2*** function **may modify** the values in the array of strings passed to it. To avoid confusion and problems with directory separator differences between operating systems, ***WinsockArrRCP2*** will convert all backslashes (\) that appear in the array of strings to slashes (/). After the ***WinsockArrRCP2*** completes, the array of strings will be changed in your program if it contained any backslashes.

If you require that the strings be maintained in their original form, make a copy of the array and pass the copy to ***WinsockArrRCP2***.

Example

```
// This example copies the files "plan.xls" and "bills.xls" in the
// current directory to the "/u/hold" directory on the "unix" system.

int result;

char *rtitle = "Winsock RCP32.DLL Sample";
char *args[] = { "plan.xls", "bills.xls", "unix:/u/hold" };

result = WinsockArrRCP(NULL,rtitle,args,3,0,0,1,0,0,0);

if (result < 0)
  MessageBox(NULL,"Remote Copy Was Not Successful",rtitle,MB_OK);
else
  MessageBox(NULL,"Remote Copy Was Successful",rtitle,MB_OK);
```

FUNCTION: WinsockArrRCP2

C:

```
int WinsockArrRCP2 (char **lpArgArr,
                    int iArgCount,
                    int iRecurFlag,
                    int iAsciiFlag,
                    int iCaseFlag,
                    int iHiddenFlag,
                    int iSpaceChar,
                    char *ErrMsg,
                    int iErrLen)
```

Usage

The ***WinsockArrRCP2*** function is similar to the ***WinsockRCP2*** function, but accepts an array of strings containing the source and destination filenames.

This allows you to copy multiple source files/directories in a single call when the files cannot be named using a single wildcard pattern.

This function is similar to the ***WinsockArrRCP*** function, but does not display error messages in Windows message boxes or to the Console window. Instead, it returns the error messages in the area you provide. You can then display them, ignore them, or process them as you require.

The strings in the array should be filenames of the format:

[[User@][Host:]]{File | Dir}

The **last** string in the array will be considered the **destination**.

The array must contain at least **two** strings (a source and a destination).

Parameters

The parameters to the ***WinsockArrRCP2*** function are the same as those passed to the ***WinsockRCP2*** function, except for the ***lpSrc*** and ***lpDest*** parameters.

WinsockArrRCP2 substitutes the following parameters for ***lpSrc*** and ***lpDest***:

lpArgArr	Pointer to an array of strings. This array must contain two or more strings. The **last** string in the array will be considered the destination of the copy.
iArgCount	The number of strings in the array. This must be greater than or equal to two (2).

Notes:

The same rules for ***WinsockRCP2*** apply to this function. Refer to that section. However, here are a few important points:

A ***Host:*** must be specified for either the source files/directories or the destination file/directory (or both). The destination is always the **last** string in the array.

If you copy multiple source files, the destination the destination must be a directory.

Important Note:

The ***WinsockArrRCP2*** function **may modify** the values in the array of strings passed to it. To avoid confusion and problems with directory separator differences between operating systems, ***WinsockArrRCP2*** will convert all backslashes (\) that appear in the array of strings to slashes (/). After the ***WinsockArrRCP2*** completes, the array of strings will be changed in your program if it contained any backslashes.

If you require that the strings be maintained in their original form, make a copy of the array and pass the copy to ***WinsockArrRCP2***.

Example

```
// In this example, we will simply use the standard command line arguments
// (argv[]) and pass them through to the WinsockArrRCP2 function.  If you
// modify the CRCP example program included with this package to do this
// you will be able to specify multiple source files on the command line.
// If any error occurs, we will display it.

main(int argc, char **argv)
{
       int result;

       char errmsg[256];

       // Skip argv[0] - that's the program name!
       result = WinsockArrRCP2(&argv[1],argc-1,0,0,0,0,0,errmsg,
                               sizeof(errmsg));

       if (result < 0)
         MessageBox(NULL,errmsg,"RCP Error!",MB_OK);

}
```

FUNCTION: WinsockRCPSetIntOption

C:

```
int WinsockRCPSetIntOption (char *lpOptionName,
                            int iOptionValue)
```

Visual Basic:

```
Declare Function WinsockRCPSetIntOption Lib "RCP32.DLL"
    (ByVal lpOptionName As String,
     ByVal iOptionValue As Long) As Long
```

Usage

The ***WinsockRCPSetIntOption*** function sets options that affect the operation of the Winsock RCP32.DLL functions. This function is used when setting options that take an integer value.

Parameters

lpOptionName	This is a string indicating the option to be set.
iOptionValue	This is the value to associate with the option.

Options Available

Debug	Specify a non-zero value to turn on the RCP32.DLL trace and 0 to turn it off. This has no effect if you do not also set the DebugFile option using the WinsockRCPSetStrOption to specify a filename. Currently, the values available or 1, 2, 3, and 4. Each gives an increasing amount of trace detail. A value of 4 causes a byte-by-byte dump of the data received or send via rcp to be recorded in the trace file, so the file can become very large. Default value: 0
ConnTimeout	Specify the number of seconds RCP32.DLL will wait to establish a connection to the server. If the connection is not established within that amount of time, the rcp function will return with an error. Default value: 60
RecvTimeout	Specify the number of seconds RCP32.DLL will wait to receive a block of data from the server during an rcp copy. If no data is received during this time, the rcp function will return with an error. Default value: 60
SendTimeout	Specify the number of seconds RCP32.DLL will wait for a block of data to be sent to the server. Default value: 60
RetryError	Set to the value of a Windows Sockets error number. If that error occurs when connecting to the server, it will retry the connection again, up to the number of times specified by the RetryCount option. Default value: 0
RetryCount	Set to the number of times you want RCP32.DLL to retry connecting to the server when one of the following TCP/IP errors occur: Address in use, Connection refused, Connection shutdown, Connection timed out, or the error number specified in the RetryError option. Default value: 1
NoProbe	When this is set to 1, RCP32.DLL does not probe for an unused local port number to use for the connection to the Remote Shell Daemon. Instead, it records the last local port used by each call and uses the next lower port number for the next connection. This helps to solve a "connection timeout" error that can occur on some networks, but adds a small overhead for

	each connection. Default value: 0
UseRexec	Set to 1 to use the Remote Exec Daemon (rexecd) on the server instead of the Remote Shell Daemon. When this is set to 1, it automatically sets the port number to use to 512 (the standard rexec port) and will look for the password in the source or destination specification or the Password option set using the WinsockRCPSetStrOption. The rexecd requires a password to be specified to complete the connection. Default value: 0
Port	Specify the port number to be used to connect to the server. The default is 514, which is the standard port for the Remote Shell Daemon. You can set this to 512 if you wish to execute the rcp through the Remote Exec Daemon (rexecd), but you must specify a password, either in the source or destination specification or using the Password option using the WinsockRCPSetStrOption.
BlockSize	Specifies the block size that the rcp functions will use to read and write data to files and send and receive data from the network. Deafult value: 4096
Blocking	Uses Winsock blocking calls when set to 1. This will disable the ability to specify connection/send/receive timeouts. Default value: 0

Return Value

The ***WinsockRCPSetIntOption*** function normally returns a value of 0. However, it will return a value of –1 if you overflow the internal table used to store options. This should never occur if you do not pass invalid option names to the function.

FUNCTION: WinsockRCPSetStrOption

C:

```
int WinsockRCPSetStrOption (char *lpOptionName,
                            char *lpOptionValue)
```

Visual Basic:

```
Declare Function WinsockRCPSetStrOption Lib "RCP32.DLL"
    (ByVal lpOptionName As String,
     ByVal lpOptionValue As String) As Long
```

Usage

The ***WinsockRCPSetStrOption*** function sets options that affect the operation of the Winsock RCP32.DLL functions. This function is used when setting options that take a string value. The string may have a length of up to 128 bytes.

Parameters

lpOptionName	This is a string indicating the option to be set.
lpOptionValue	This is the value to associate with the option.

Options Available

DebugFile	Specify the filename to hold trace output. This is used in conjunction with the Debug integer option. See the description of the Debug option for more details.
User	Use this to specify the user to send to the remote host for the rcp connection. This is used when the user is not specified in the source or destination file specification, to override the current Windows user.
Password	When the integer option UseRexec is set to 1, you can use this option to specify the password to be sent to the Remote Exec Daemon server for the user's login.

Return Value

The ***WinsockRCPSetIntOption*** function normally returns a value of 0. However, it will return a value of –1 if you overflow the internal table used to store options. This should never occur if you do not pass invalid option names to the function.

Example

```
// This example copies the file "system.doc" in the user tom's home
// directory on the host "remhost" to the directory "\doc" on the PC
// If any error occurs, we will display it.
//
// Instead of using the Remote Shell Daemon on the server, we will use
// the Remote Exec Daemon, so we must also specify a password for the
// connection.  We will also override the current Windows user with
// the user “mary”.

int result;

char *src = "remhost:/usr/system.doc"
char *dest = "/doc"
char errmsg[256];

WinsockRCPSetIntOption(“UseRexec”,1);
WinsockRCPSetStrOption(“User”,”mary”);
WinsockRCPSetStrOption(“Password”,”HadALittleLamb”);

result = WinsockRCP2(rtitle,src,dest,0,0,0,0,0,errmsg,sizeof(errmsg));

if (result < 0)
  MessageBox(NULL,errmsg,"RCP Error!",MB_OK);
```

FUNCTION: WinsockRCPCancel

C:

```
int WinsockRCPCancel(void)
```

Visual Basic:

```
Declare Function WinsockRCPSetStrOption Lib "RCP32.DLL"() As Long
```

Usage

The ***WinsockRCPCancel*** function cancels the currently executing rcp function. The ***WinsockRCP**** function will return with an error status of –22 if it was canceled during the actual file transfer, indicating it was canceled.

Important Note: The ***WinsockRCPCancel*** function **must** be called from the *same thread* that initially called the ***WinsockRCP**** function to be canceled.

RETURN VALUES FOR ERRORS

When an error occurs during a file transfer with ***WinsockRCP, WinsockRCP2***, ***WinsockArrRCP,*** or ***WinsockArrRCP2,*** a negative value is returned indicating the **last** error that occurred. Constants for these error numbers are defined for you in the files RCPERRS.H (for C) and RCPERRS.BAS (for Visual Basic). The following are the possible values:

Error Number	Meaning
-1	No host specified for either filename.
-2	Unacceptable user name before @ sign in filename.
-3	The error message was received from the remote host.
-4	Lost the connection to the host prematurely (network error)
-5	Cannot allocate memory.
-6	Can't get user from Windows or WIN.INI.
-7	Target is ambiguous.
-8	Invalid file type to send (a directory was specified without recursion).
-9	No match for wildcard on remote host.
-10	RCP Protocol Error.
-11	Can't get information about a directory on the PC.
-12	Error in the wildcard pattern used.
-13	Can't get information about a file to send.
-14	Can't open a file to send.
-15	Error reading a file while sending.
-16	Cannot get the filenames in a directory.
-17	Pathname is too long.
-18	Cannot open a file to be received for writing.
-19	Error adjusting file size after receiving. Possibly out of disk space.
-20	Error writing file while receiving. Possibly out of disk space.
-22	The transfer was canceled by the WinsockRCPCancel function.

SUPPORT

Support is available via E-Mail:

Internet: support@denicomp.com
WWW: http://www.denicomp.com