How to Connect to a Network Through the WiFile Pro API

This document describes to Palm OS application developers how to interface with WiFile Pro to add networking capabilities to their applications and easily access their data on the network.

About WiFile Pro

WiFile Pro is a networking application that supports a number of protocols, including:

- SMB/CIFS (Windows networking)
- FTP
- WebDAV
- VFS (plug-in cards)

About SMB/CIFS

SMB/CIFS is the native networking language of Windows computers. When you look in your Network Neighborhood on a Windows desktop computer, those computers are communicating using SMB/CIFS. Since WiFile Pro is an SMB client, the user has very little setup to do on his/her device to connect to a server volume, and there is no requirement for any additional software to be installed on the server in order to enable access. Also, Mac OS X, Unix, and Netware all have SMB server products, so those OS's can be accessed as well with the same protocol.

About FTP

FTP is an old Unix standard that is available all over the Internet, and in many corporate Intranets. FTP is not very secure, but is a great way to distribute files to anonymous users. WiFile Pro supports many popular FTP servers, including those provided by Windows XP Pro, Mac OS X, and Linux.

About WebDAV

WebDAV is an up and coming standard for file transfer across the Internet. It is a series of XML extensions to standard html. While it can be relatively difficult to set up a WebDAV server, it has a number of advantages over FTP:

- Since it works over http, the Web server provides authentication services. Authentication can be done on a per user and per directory basis. Whatever your Web server supports for authentication of Web pages can be used to authenticate WebDAV connections.
- WebDAV can communicate over https, so the entire communication can be encrypted as easily as setting up secure Web pages.

- Like many FTP servers, WebDAV is open source and free. However, WebDAV is better defined and so has fewer compatibility problems.
- Many corporations are using WebDAV to provide file access behind a firewall without needing a VPN server. This setup is more secure than VPN, because it limits the user access to specific files or directories, while a VPN gives access to the entire network.

Typical Handheld Configurations

Typical ways a user might connect to the network to access files are:

1) With an 802.11 capable device that is connected to a WiFi hub attached to the network.
2) With a Bluetooth enabled device that is paired with a desktop computer or laptop computer that is attached to the network. The computer essentially becomes a router or bridge between the device and the network. Bluetooth hubs are also available.
3) With a phone enabled device, like a Treo, probably also using VPN software to tunnel through a firewall to get to a corporate network.

There are other methods available. See the Solutions Guide for a bigger list.p

Launching a File

From within the WiFile Pro application, if the user taps on a file, WiFile Pro will try to "launch" the file as described below.

If the file is a .PRC file, it will simply copy the file to the handheld and start it with a SysAppLaunch.

If the file is a .PDB file, it will look in the header of the database to find the creator of the database, and if the creator exists, it will launch the application with a goto command, specifying the database id in the goto parameters of the launch. All other goto parameters will be zero. However, before launching, WiFile Pro will also send a sysAppLaunchCmdSyncNoitfy launch code to notify the application that a new database has been added to the device.

If the file has an extension (e.g. .doc or .txt), WiFile Pro calls the exchange manager function ExgGetDefaultApplication to see if an application is installed that can handle the file, and then calls ExgPut and ExgSend to stream the file to that application. It will also look to see if the View Attachment extensions are available, and will send the file as an attachment view if so. Viewing attachments is a relatively new feature of the Palm OS that you can find out about it from the PalmSource Web site knowledge base, answer # 493, at http://kb.palmsource.com/cgi-bin/palmsource.cfg/php/enduser/std_adp.php?p_faqid=493. If your program supports a

native file format, you should consider providing a view interface to further enhance the user experience, allowing the user to download files from the network through WiFile Pro, or view them from an attached email.

Copying a File

If the user chooses to copy the file to a memory card, WiFile Pro will try to figure out a default location for that file, and give the user the option of copying to the default location, or the specified location. If the file is a .PDB or .PRC file, the user will also have the option of copying to internal memory.

WiFile Pro will call VFSGetDefaultDirectory to figure out the default location of a file. If your application implements some kind of native file support, and you have registered the application as the default application for the file type through the exchange manager, you should also register the default directory where you access these files on VFS memory cards using the VFSRegisterDefaultDirectory call. This will enable WiFile Pro to put the file in the correct place for your application to access it.

The WiFile Pro Exchange Library

WiFile Pro includes an exchange library to let you communicate with WiFile Pro from within your application.

The WiFile Pro exchange library registers the following schemes that you can use to ask for files:

- File:
- SMB:
- WebDAV:
- DAV:
- WiFile:
- _send and _get

The WiFile Pro exchange library also implements the ExgRequest function. This function will let you specify the path of a file on the network, and will open that file in the Palm OS application that is registered to handle that file's type. In other words, using ExgRequest, you could pass a URL to WiFile Pro that points to a .JPG file, which would then cause WiFile Pro to stream that file to the default picture viewing application on the device.

The following sections describe in more detail how to use the WiFile Pro exchange library to access files on the network.

WiFile Pro URLs

When you get or send data through the exchange library, you specify the URL to the file that you want in the *name* field of the exchange socket that you send. Depending on the format of the URL, and what you are trying to do, you can specify a URL that causes the exchange library to popup a dialog that will allow the user to pick a file, save a file, or just grab a file straight from the network with no user interaction. These dialogs are similar to what you would see in desktop software when you choose Open or Save As from the File menu.

If you specify a URL that allows the user to choose a file, you might want to know the URL of the file that the user ended up choosing. For example, you might want to later read this same file, or save to that file. To get this URL, call the ExgControl function and pass the exgLibCtlGetURL value as the *op* parameter. This control value is a new addition to the exchange manager API and is described in the new Bluetooth Exchange Library section of the Palm OS Programmer's Companion manual. A sample of how you might call the function is below. Note that in the sample, you call the function twice, once to get the size of the URL, and the second time with a pointer that has enough memory allocated to hold the URL.

Code Sample For Getting the Current URL

```
ExgCtlGetURLType getUrl;
UInt16 getUrlLen;

// First get the size of the URL
getUrl.socketP = exgSocketP;
getUrl.URLP = NULL;
getUrl.URLSize = 0;
getUrlLen =  sizeof(getUrl);
ExgControl(exgSocketP, exgLibCtlGetURL, &getUrl, &getUrlLen);

// Now get the URL
getUrl.URLP = MemPtrNew(getUrl.URLSize);
ExgControl(exgSocketP, exgLibCtlGetURL, &getUrl, &getUrlLen);

// getUrl.URLP points to a null-terminated URL string
// describing the remote device, for example,
// "wifile://connection/dir1/dir2/file.ext"
...
// Free the URL after you're done with it
MemPtrFree(getUrl.URLP);
```

The URL will be available after calling ExgGet, ExgPut, or after calling ExgAccept in response to a sysAppLaunchCmdExgReceiveData launch code.

URL Formats

Palm OS URLs start with a scheme. WiFile Pro registers the following schemes:

wifile	Communicates through any of the listed connections in WiFile Pro.
file	Transfers files to and from memory cards through the VFS manager
dav webdav	Communicates to WebDAV servers using the http protocol
davs webdavs	Communicates to WebDAV servers using the https protocol
ftp	Communicates to FTP servers

WiFile Pro URLs work with connections listed in the WiFile Pro application. To use a WiFile Pro URL, use a single slash after the scheme and before the connection. The possible WiFile Pro URLs are:

URL Format	Get or Send	Description
wifile: wifile:/	Get or Send	Ask the user for the location to get or save. WiFile Pro will start with the most recently open location on the network and put up a dialog allowing the user to browe to a WiFile Pro location.
wifile:/*	Get	Return a tab delimited list of the connection names in WiFile Pro. You can use this to build a URL programmatically, or implement your own browser.
wifile:file.ext	Send	Use this format in a "Save As" situation. WiFile Pro will open the most recently used directory, and default the file name to "file.ext". The user can change the name and pick a new location if desired and save the file.
wifile:/connection/dir1/.../file.ext	Get or Send	Open or save a particular file on the network. If saving, the file is over-written without asking.
wifile:/?connection/dir1/.../file.ext	Send	Use this when saving a file if you would like the user to be asked whether to over-write the file before saving if the file already exists.
wifile:/connection/dir1/…/*	Get	Returns a tab delimited list of the contents of a directory. Directories will end with a forward slash (/). Use this to implement your own browser, or build a directory tree.
wifile:/connection/dir1/…/dir/	Get or Send	Will prompt the user to get or send the file using the save or open dialog, but will default the dialog to show the given location.

WiFile Pro also implements a standard URI scheme that is common when referring to objects on the Internet or an intranet. Supported formats are as follows:

URL Format	Get or Send	Description
<scheme>://user:password@server:port/dir1/../file.ext	Get or Send	Open or save a particular file on the network. If saving, the file is over-written without asking.
<scheme>:///?user:password@server:port/dir1/.../file.ext	Send	Use this when saving a file if you would like the user to be asked whether to over-write the file before saving if the file already exists.
file:///[?]volume/dir1/.../file.ext	Get or Send	Open or save a particular file on the network. If saving, the file is over-written without asking, unless the question mark is present

Where:

- *<scheme>* is one of smb, ftp, dav, davs, webdav or webdavs.
- *User:password@server:port* is the user name, password, server name and TCP/IP port of a computer and account on that computer. User and password are optional if the shared directory is publicly accessible. The server name can be an IP address, domain name, or for SMB shares, a NETBIOS name. Note that if your network requires a Scope ID, Domain Name or special character encoding, you cannot use this format.
- *Dir1*/../is the path to the file from the root of the specified share. If no path is specified, the file will be on the root.
- *File.ext* is the name of the file to open or save.

Getting Data from the Network to Your Application

If you would like to allow users to access data directly from the network while running your application, you can use the relatively new ExgGet mechanism.

Depending on the URL that you pass in the socket when you call ExgGet, the exchange manager will route the call to the WiFile Pro exchange library, and the library will either return the specified file immediately, or put up a dialog in your program that allows the user to browse the servers he has set up and pick a file from a server to copy down to you. See the section on URL formats for more information on creating a URL.

Here is an example of calling WiFile Pro with ExgGet. This example also calls the ExgControl function to get the URL that the user chose.

```
Err DoGetRequest(void)
{
   ExgSocketType exgSocket;
```

```
   Err error;
   UInt32 totalBytes = 0;
   MemHandle textH;
   UInt16 id[1] = {exgRegSchemeID};
   Char *type[1] = {exgGetScheme};

   ExgCtlGetURLType getUrl;
   UInt16 getUrlLen;

   MemSet(&exgSocket, sizeof(exgSocket), 0);
   // Allocate memory for all buffers that will hold response
information from
   // the replying application.
   exgSocket.description = (Char *)
MemPtrNew(exgMaxDescriptionLength + 1);
   exgSocket.type = (Char *) MemPtrNew(exgMaxTypeLength + 1);
   exgSocket.name = exgSocket.name = (Char *) MemPtrNew(1024 +
1);
   MemSet(exgSocket.description, sizeof(exgMaxDescriptionLength)
+ 1, 0);
   MemSet(exgSocket.type, sizeof(exgMaxTypeLength) + 1, 0);
   MemSet(exgSocket.name, 1024 + 1, 0);

   // Initialize the name field to make a local connection
   // that only lets the user open and see files that end with
.txt.
   StrCopy(exgSocket.name, "smb:*.txt");

      if (!error) {
            error = ExgGet(&exgSocket);
            if (!error) {
                  UInt32 bytesRec;
                  char buf[100];
                  Boolean done = false;

                  // now get the URL of the chosen file in case we
want to send the info back
                  // First get the size of the URL
                  getUrl.socketP = exgSocketP;
                  getUrl.URLP = NULL;
                  getUrl.URLSize = 0;
                  getUrlLen =  sizeof(getUrl);
                  error = ExgControl(exgSocketP, exgLibCtlGetURL,
&getUrl, &getUrlLen);

                  if (!error) {    // perhaps the library doesn't
support URLs
                        // Now get the URL
                        getUrl.URLP = MemPtrNew(getUrl.URLSize);
                        ExgControl(exgSocketP, exgLibCtlGetURL,
&getUrl, &getUrlLen);
```

```
                        SetCurrentURL (getUrl.URLP);      // copy
the URL somewhere so we can use it later
                  }
                  MemPtrFree(getUrl.URLP);

                  while (!done) {
                        bytesRec = ExgReceive(&exgSocket, buf, 100,
&error);
                        if (error) {
                              break;
                        }
                        totalBytes += bytesRec;
                        if (!bytesRec) {
                           done = true;
                        } else {
                              PutData (buf, bytesRec);    // do
something with the data we get back from the sender
                        }

                  }
                  error = ExgDisconnect(&exgSocket, error);
            }
      }

   MemPtrFree(exgSocket.description);
   MemPtrFree(exgSocket.type);
   MemPtrFree(exgSocket.name);

   return error;
}
```

Requesting Data for Another Application

The WiFile Pro exchange library supports the ExgRequest call. This call allows you to request a particular file, and have that file sent to the application that is registered to handle it. For example, you could send a request to show a .txt file, and that file will popup in the Memo Pad application, or whatever application is installed and registered to handle .txt files.

To do an exchange request, call ExgRequest with a new socket set up in the following way:

name – Put a pointer to the full URL into the name field of the socket.

goToCreator – If you would like to launch the target application with the View extensions, put the creator of your application in this field of the socket. The View extensions will keep the “ask” dialog from showing, and will make the target application

immediately launch you again after the user taps the Done button. If you do not want the data to be "Viewed", set the goToCreator to zero. Note that if you specify a goToCreator, but the target application does not support View mode, then the data will be sent normally, with the user being asked whether to accept the data, and then the data will be saved locally on the device.

goToParams – Fill these with the parameters that you want your application to be launched with after a View.

target – If you know the creator id of the application to which you want the data sent, put the creator ID in the target field. If you leave the target at zero, then the default application for that particular kind of data will be launched. If you set the target to 1, then the user will be asked which application to send the data to if there is more than one application that has registered for that kind of data.

Saving Data from your Application to the Network

Since the WiFile Pro exchange library registers for the generic "_send" scheme, any application that currently supports the standard Palm send mechanism will be able to send data through WiFile Pro to a networked drive.

To save a file that you originally got through the ExgGet mechanism, you will need to first get the URL of the file that you received by calling ExgControl as described above, and then put a pointer to that URL into the name field of the socket when you call ExgPut. The file will be saved back to the network without asking the user. This would be a standard "Save" call.

To save the file, and ask the user where to put the file, like in a "Save As" command, you will simply put the file name after the "wifile:" prefix, with no slashes. WiFile Pro will pre-fill the save dialog with the name you specify, and then allow the user to change the name, and browse to a different directory to save to.

To have the user be asked where to put the file, and have no name pre-filled in the dialog, simply pass the "wifile:" prefix in the name field and nothing else.

The Ultimate Experience

The ultimate user experience when working with WiFile Pro is as follows:

1) User browses to a file on the network using WiFile Pro.
2) User double-taps the file to open it.
3) WiFile Pro then sends the file to the program that handles files with the given file extension.
4) The file is displayed in the destination program using the View exchange manager extensions.
5) The user modifies the file and presses the Done button.

6) The destination program asks whether to save the file back to the network.
7) If yes, the program sends the file through the exchange library with the same URL and the file is saved to the network.

To enable your program to provide this experience, you will need to make the following changes to your program:

1) Implement the View exchange manager extensions so the user can open the file in your program without having to save it locally. Make sure you can receive a file from the exchange manager that is sent to you from an email program like SnapperMail. This mainly involves changes to how you handle the sysAppLaunchCmdExgReceiveData launch code.
2) Check the goToCreator that is sent in the exchange manager socket. If a creator code is specified, you know you are handling a View document.
3) Check for a URL. If the URL begins with “wifile:”, then save the URL so you can send the file back later, if the user changes it. Good places to save a URL like this are in a preference, feature or AppInfo block of a database.
4) When the user presses the Done button, see if you are in View mode, and have a URL. If so, and the user changed the data, ask the user whether to save the information back to the network. If the user agrees, call ExgPut, putting the URL in the *name* field of the socket. The file will be saved.