

Beginning ASPECT Scripting

Table of Contents

Beginning ASPECT Scripting	1
Table of Contents	3
Introduction	4
About This Manual	6
Chapter 1 - Structure of a Script	7
Script Requirements	7
Data Types.....	8
Comments.....	10
Chapter 2 - Running Scripts	11
Icon on the Desktop	11
Exercise 2a	15
Within the Connection Directory	15
Exercise 2b	18
From the Terminal Screen	19
Using the Scheduler	20
Exercise 2c	25
Chapter 3 - Help File.....	26
Commands Listed by Function.....	27
Sections in the Help Listings.....	29
Chapter 4 - Logon Scripts.....	34
Preliminaries	34
Recording a script.....	35
Edit the Script	37
Improvements	42
Exercise 4a	43
Chapter 5 - Simple I/O	44

Standard Dialog Input Box.....	44
User Message Window	46
Exercise 5a.....	49
Exercise 5b.....	49
Chapter 6 - DOS Commands	50
File-moving Commands.....	50
Directory Commands	52
Exercise 6a.....	54
Exercise 6b.....	54
Exercise 6c.....	55
Chapter 7 - File Transfers.....	56
Transfer Protocols	56
File Transfer Commands	58
Related System Variables	59
Exercise 7a.....	60
Appendix A - Exercise Scripts	61

Introduction

If you've called an on-line service, host system or BBS before, you've probably found yourself performing many repetitive tasks. For instance, each day, you may connect to a host system, log on with your ID and password, read any new electronic mail you've received, upload a data

file and log off. Although this example session is quite simple, it can be tedious – why not let Procomm Plus do it for you, automatically?

An ASPECT script file is a simple ASCII file you create containing commands to be executed by Procomm Plus. In our example above, you could write a script that would call or connect to your host and perform the same functions, just as if you were entering them at the keyboard. To run the script, you'd select the script name from the Action Bar and click on the Run Script icon.

Other applications for scripts include:

- If you are working in a doctor's office that routinely transfer medical information and claims to insurance carriers, an ASPECT script can easily handle this task for you.
- You could create a script for non-programmers to download a database file from the home office every morning and rename the file with today's date.
- It would be easy to write a procedure that uploads the day's sales and orders to a remote computer.
- You could connect to a computer-controlled instrument and start a capture file to log the data as it scrolls across the screen.

About This Manual

Audience

This manual is intended for a user with little or no experience in programming.

Commands and Syntax

We will cover a fraction of the total ASPECT commands in this manual. It is intended to give you training in these commands and to teach you how to use the available resources. Most of the commands that are included in this manual are given with abbreviated syntax and a small amount of Help information. Our goal here is to make you aware of the types of commands that are available for various circumstances and why you would want to choose one or the other. It is expected that you will refer to the on-line Help included in Procomm Plus for specific syntax. The ASPECT Script User's Guide is another resource of information; this book is available for purchase from Symantec.

Chapter 1 - Structure of a Script

Script Requirements

Let's look at a simple script:

```
proc main
  usermsg "This is a test"
endproc
```

This meets the basic requirements of a script:

- **proc main**
There must be one Main procedure and only one. There may be other procedures but they will have unique names.
- **endproc**
Every procedure must terminate with an "endproc".
- **commands**
Any of the commands from the manual or the Help screens may be used.

Data Types

All of the variables that are used in ASPECT must be of four types:

- integer
- string
- float
- long

These variables must be declared before they can be used. The variables should have names that are meaningful, such as, `FileTransferName` or `TotalWidgets`. The names may be up to 256 characters in length but must be unique within the first 30 characters. The ASPECT compiler is not case sensitive so capital letters are used only to make it easier to read the script. By convention, key words and system variables are capitalized but that is only for readability.

Declarations

Most variables will be declared locally:

```
proc Main
  integer iTemp = 4
  string StudentName, FileTransferName
  .
  .
```

They exist and have values only in this procedure. The variables can be initialized in the declaration, as in “`iTemp = 4`”.

Global declarations are made before the “proc Main” line:

```
string MyVariable, UserAddress

proc Main
  integer NumCows

  UserAddress = "my house"

endproc

proc Second

  usermsg "I live at %s" UserAddress

endproc
```

Because UserAddress is a global variable, it can be defined in one procedure and used in another. This is a rather trivial example and would probably not be used in a script. Global variables can lead to sloppy and confusing programming. They should only be used when it is required by the commands or it makes the script easier to follow.

Predefined global variables don't need to be declared because the system already knows about them. These are:

- s0 - s9
- i0 - i9
- l0 - l9
- f0 - f9

These variables are used for communicating between scripts or over a DDE link to another program, such as Excel or Access. Yes, they can be used for “quick & dirty” programming because they don't have to be declared but that is bad technique!

System variables are reserved words that begin with \$. The list of system variables and their meanings is in the Help Files. These are read-only variables that the ASPECT system provides to the programmer in order to get information from the system.

Comments

Comments begin with a semi-colon anywhere on the line. For neatness in programming, you should stick to one format for your comments. They may always start in column 40 following a command. Another method to put the comments on a separate line starting in column 1 or column 20. There should be enough comments that someone else can pick up your script and follow the logic.

There is another less-used method of commenting which uses the `#COMMENT` syntax. This is a block of several lines of comments with `#COMMENT` preceding the block and `#ENDCOMMENT` following the block. The comment lines within the block do not have initial semi-colons.

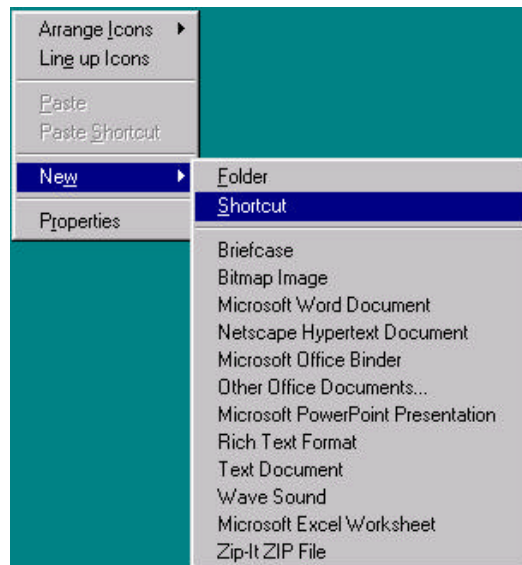
Chapter 2 - Running Scripts

There are four primary ways to run scripts: from the terminal window, with an icon on the desktop, from the Scheduler and within the Connection Directory.

Icon on the Desktop

This method of starting a script is the easiest method. It is especially efficient if you are going to be using the script often or if the script is going to be utilized by someone with little knowledge of Procomm Plus.

Right click on desktop then select New | Shortcut.



This will open a Create Shortcut Wizard.



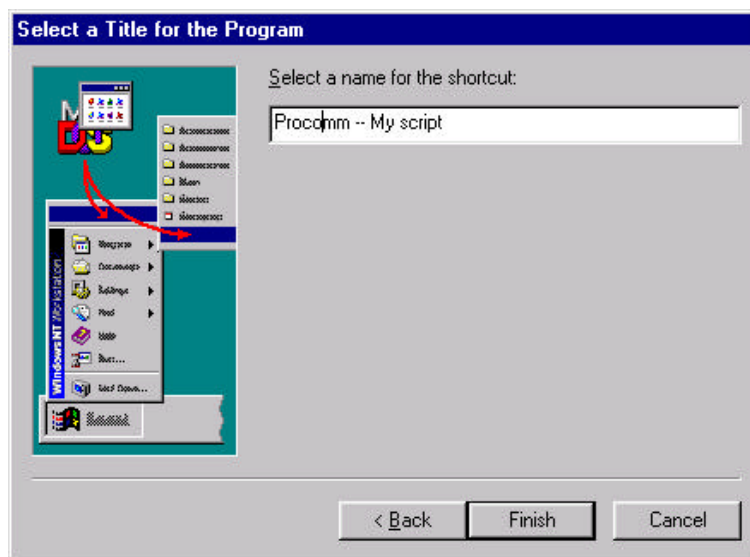
Click on the Browse button. This will open a standard Windows browse window. Select Program Files | Procomm Plus | Programs | PW4.EXE (assuming you installed Procomm Plus in the default directory)



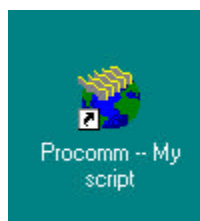
Then click inside the Command Line field at the end of the quote marks and add the name of your script with a space between the quote mark and the name.



Click on the Next button and you will be asked for a name for the icon. This can be as descriptive as you want.




After you click on Finish, you will see the resultant icon on your desktop.



All you have to do to start Procomm and execute the script is double-click on this icon. The rest is automatic!

Exercise 2a

-  Write a test script that displays a user message. Create an icon on your desktop to run your test script.

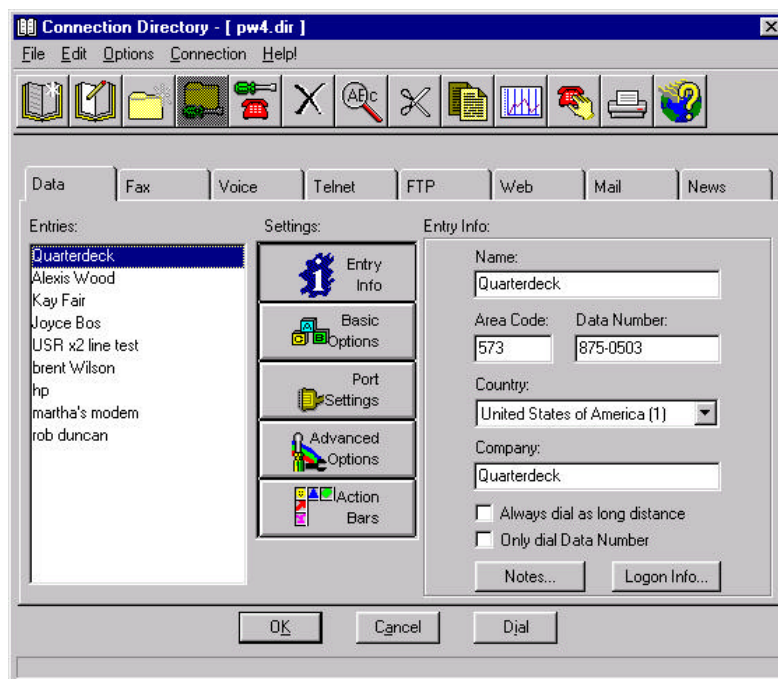
Within the Connection Directory

This method of starting a script is often used to transmit logon information to a remote system. It may also be used to automate uploads and downloads. It is beneficial if you connect to several different sites in rapid succession.

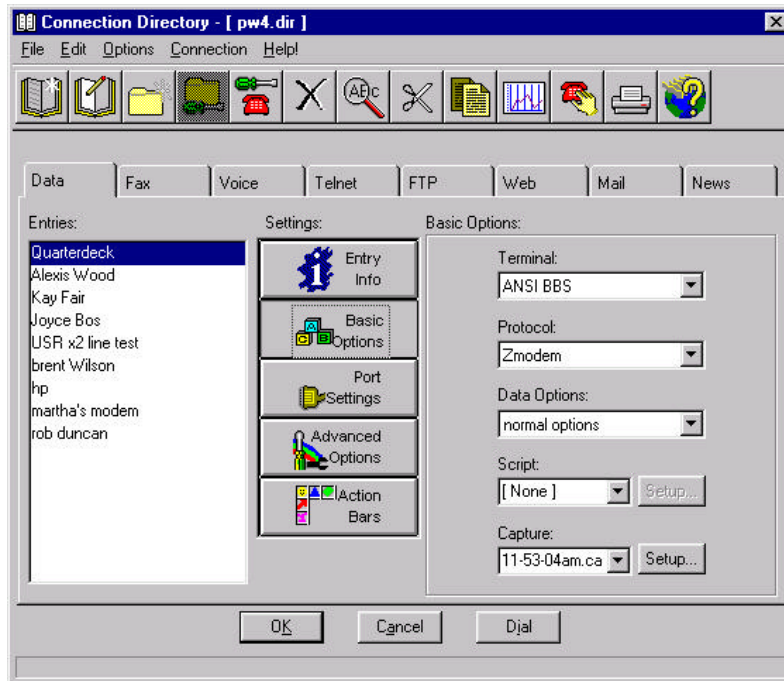
First, we will assume you have a Connection Directory entry for the remote location. You have tested this connection and it works OK.



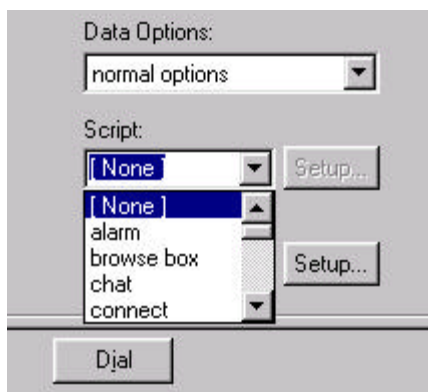
Click on the Action Bar icon button with the open phone book on it. This will open the Connection Directory.



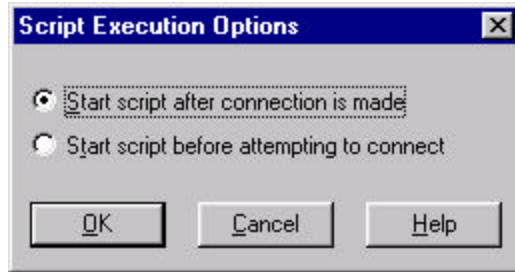
Now, click on the Basic Options button.



There is a drop-down list with all of the compiled scripts in your Scripts Path. You can select the script you want to be associated with this Connection Directory entry.



Once you have specified a script, you can go to the Setup screen.



In general, you can choose to start the script after the connection is made. The conditions that might warrant starting the script before attempting to connect would include the need to respond to the very first characters which might be transmitted after the connection. There might be a small initiation period while the script starts up before it is ready to react.

After you have specified the script in the Connection Directory, be sure to click on OK or Dial to close the Connection Directory. If you click on Cancel, your changes will not be saved.

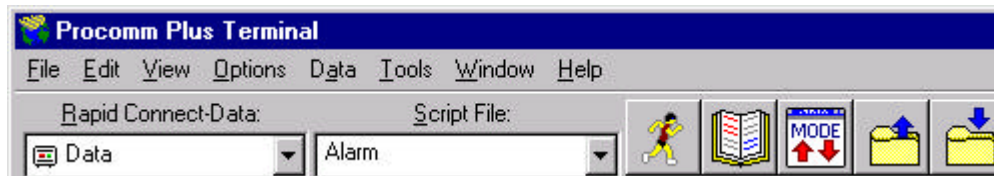
Exercise 2b



Create an entry in your Connection Directory to connect to the Symantec BBS and execute your test script.

From the Terminal Screen

If you want to run several scripts for one remote system, then you can specify them from the Terminal screen.

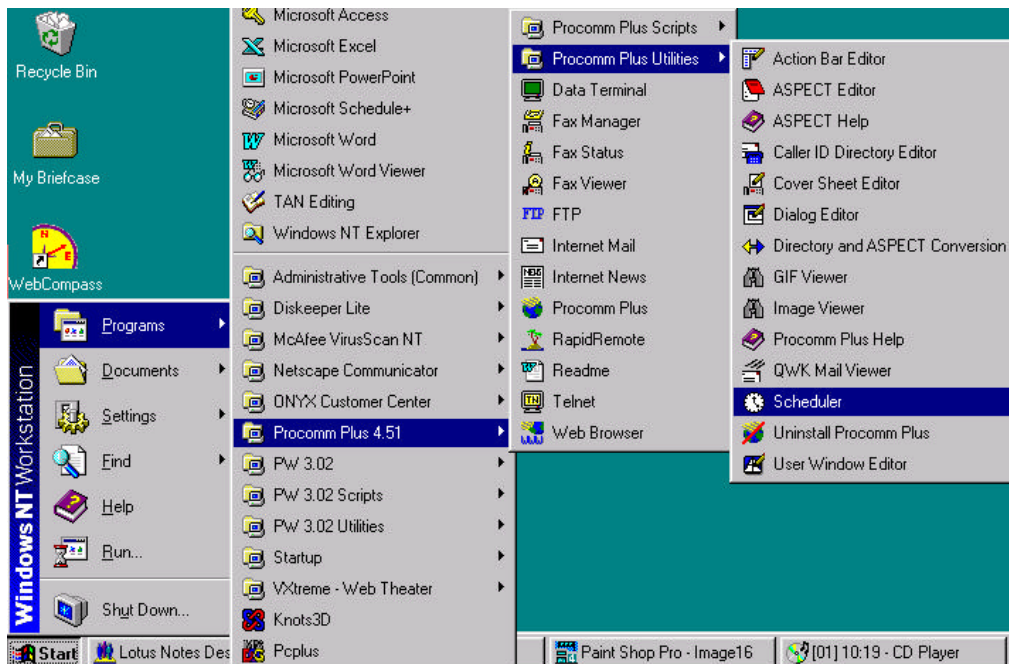


There is a drop-down list of compiled scripts in your Scripts Path. You can select one of them. If the script name is showing in the list box, then you can click on the running man icon on the Action Bar. The man will continue running as long as a script is executing. You can also click on the running man icon button to end a script.

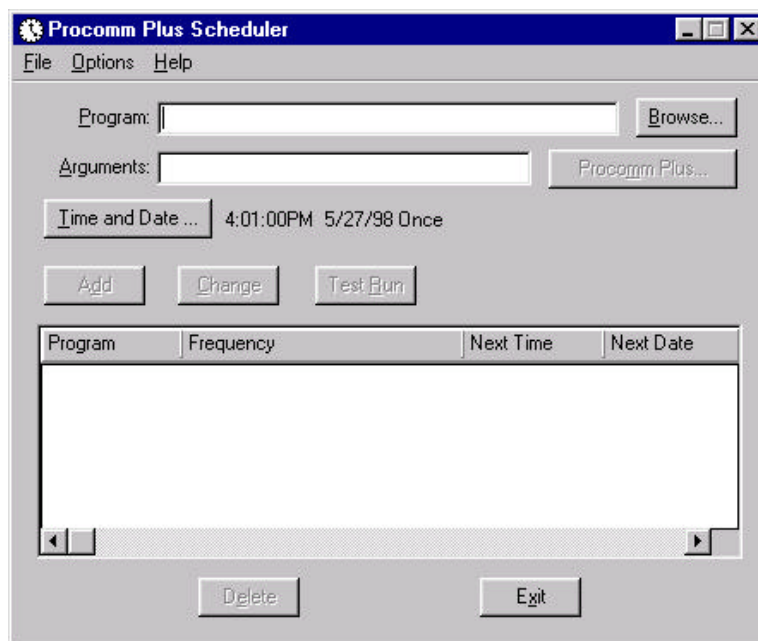
Using the Scheduler

This is a very convenient way to start a script at midnight every night when the telephone is cheaper. Perhaps, you want to get the latest database from your main office every morning at 6:00 AM. It could be that you want to upload or download files on the weekend when your mainframe isn't very busy. Whatever the reason, the scheduler is an easy way to accomplish this goal.

Go to Start | Programs | Procomm Plus | Utilities | Scheduler.



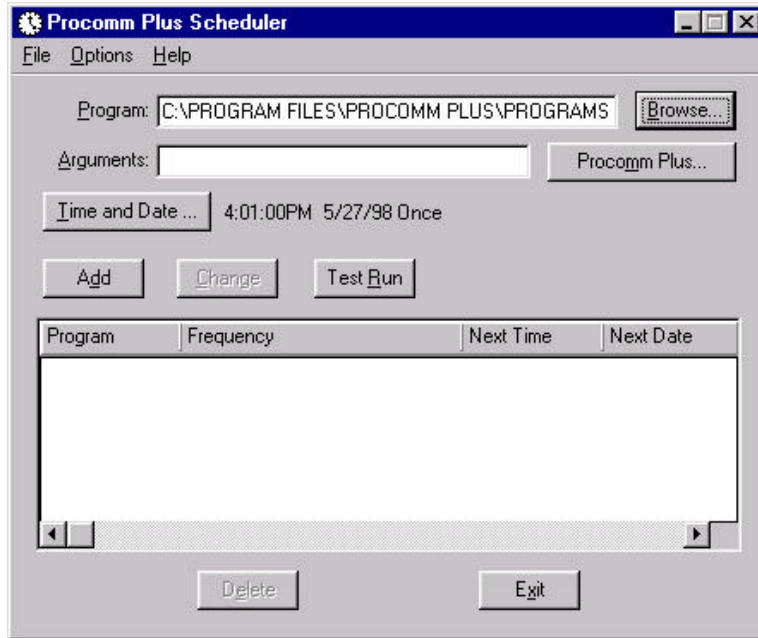
This will bring up the Scheduler window.



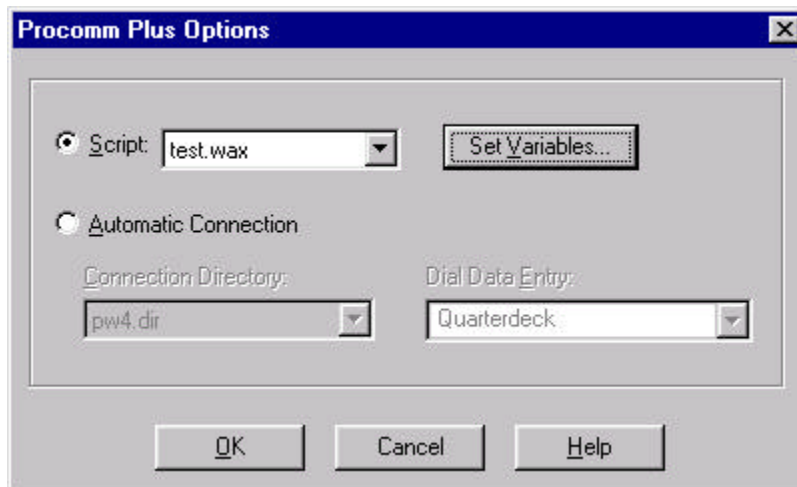
The first step is to browse for PW4.EXE.



This fills in the Program box.

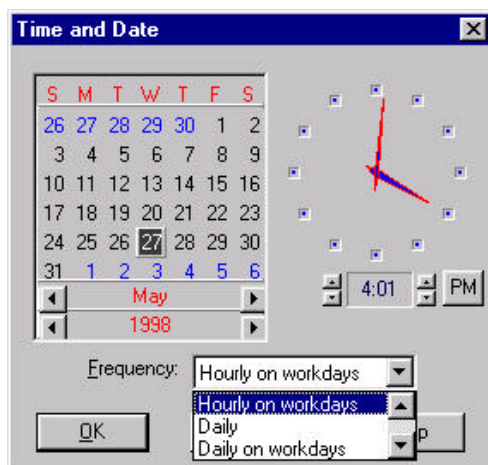


Then click on the Procomm Plus... button to specify the arguments.



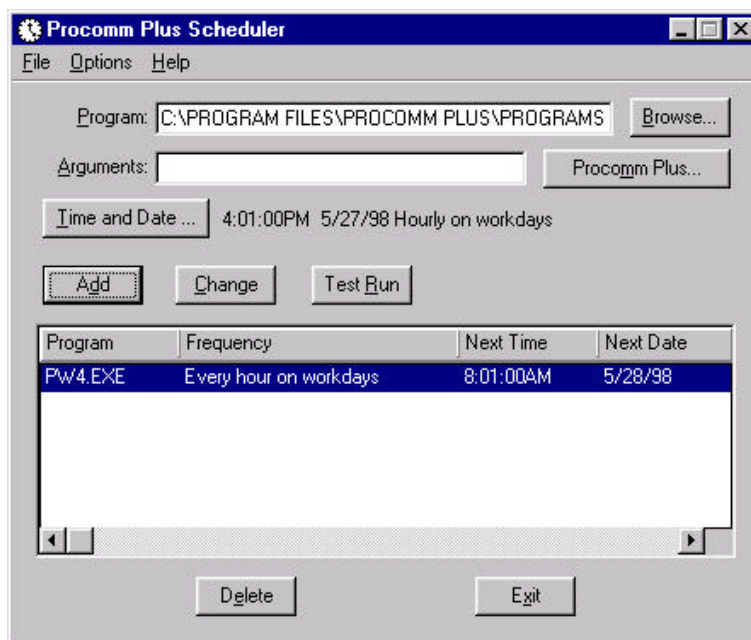
In this window, you can select the radiobutton for Script and click on the drop-down list to choose your script. The result will be to start the script from the command line that is what we did when we started the script from a desktop icon. Notice that you could alternatively choose Automatic Connection and run a script that is associated with the Connection Directory entry.

Next, you need to set up the times and dates.



On this screen, you choose the starting date and time then you specify the frequency. The frequency can be once, hourly, hourly on workdays, daily, daily on workdays, weekly or monthly. You have the ability to set up your definition of workdays including hours and days of the week.

Next click on the Add button and you will see your scheduled program in the list.



You will then have the option of changing any of the settings or doing a test run. The test is useful to be sure that you have specified the correct script.

Note that Scheduler must be running in order for the program to be started. It doesn't have to run continuously but it must be running when the event is to take place. If Scheduler isn't running when the event is scheduled, it will immediately execute the task the next time it is opened.

Exercise 2c

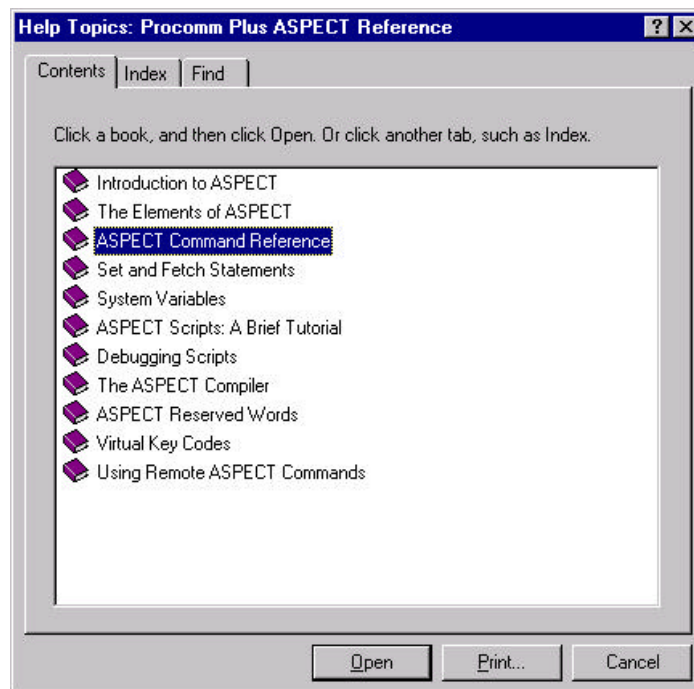


Create a scheduled event to start your test script 5 minutes from now.

Chapter 3 - Help File

One of the most important things you can learn about a programming language is how to use the Help files. No one ever learns all the commands and other nuances but you can learn where and how to get more information.

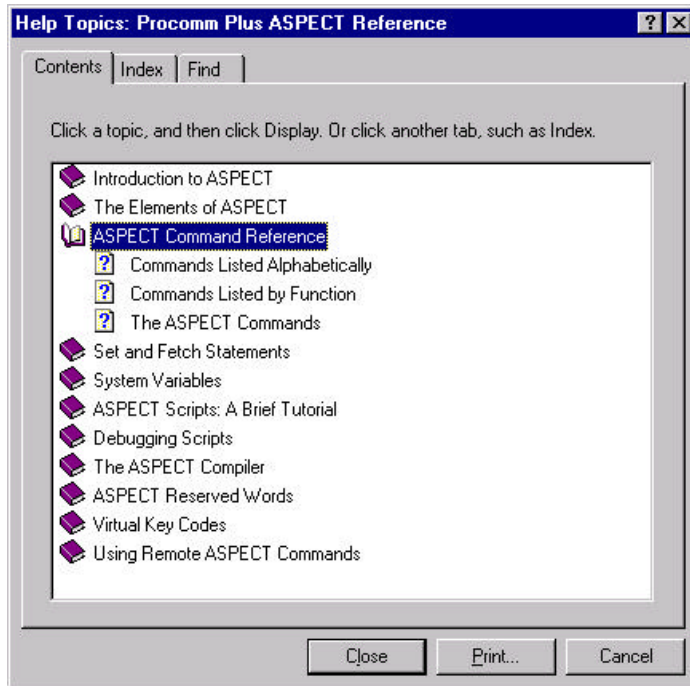
The Help information for ASPECT is available from the PW Editor screen as well as any of the modes of Procomm Plus. When you choose Script Reference under the Help menu, you will get the following screen:



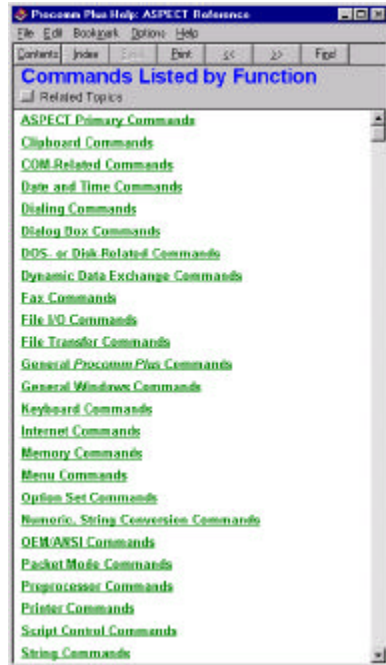
This screen lists the most important topic areas.

Commands Listed by Function

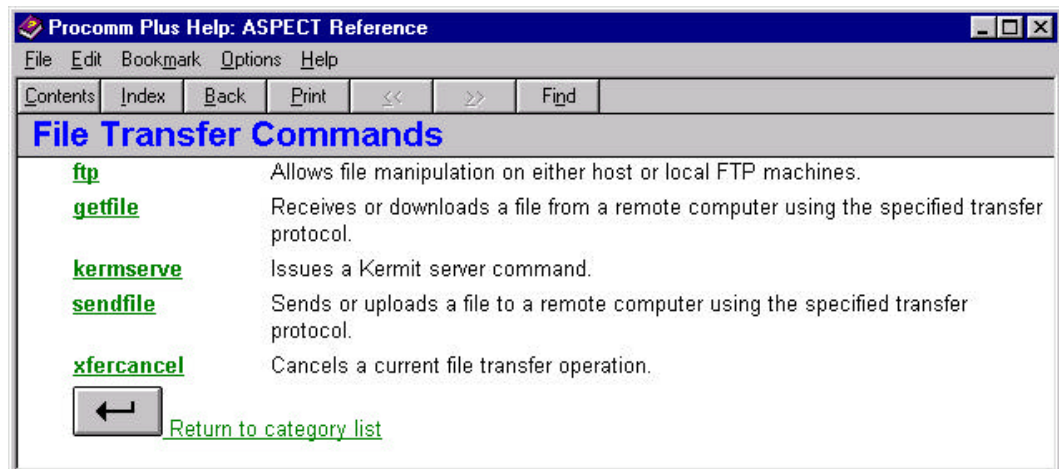
One of the most helpful places is the listing of commands arranged by function.



Choosing **Commands listed by Function** will open the following window.



This is an easy starting point to locate a command when you don't know what the command might be called. For instance, you want to send or receive a file. Let's look at the **File Transfer Commands**.

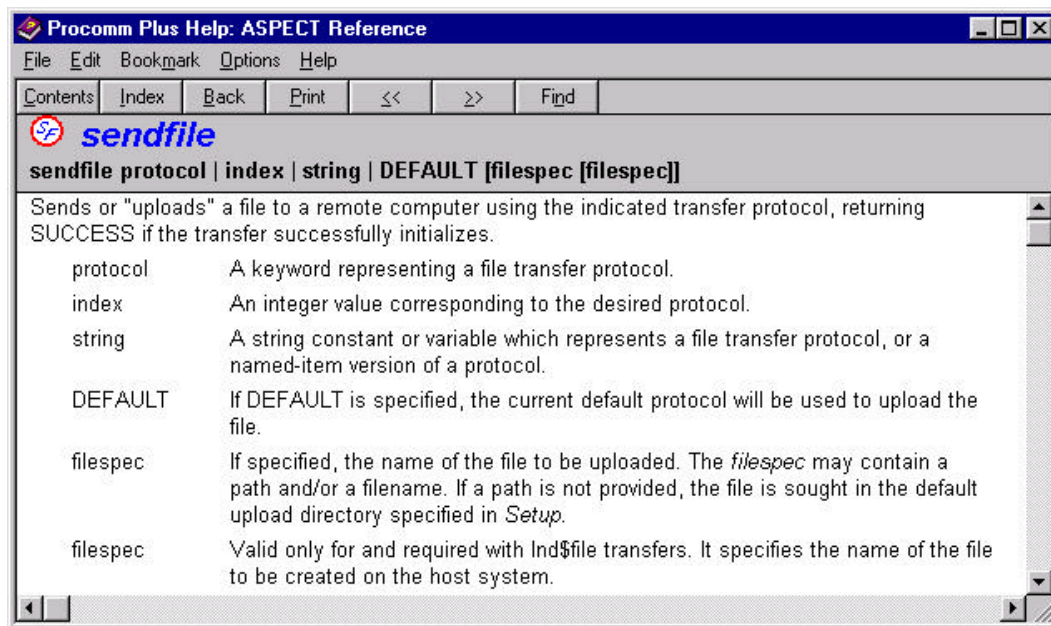


From this short list it is easy to guess that if you want to upload a file, you might use the **sendfile** command.

Sections in the Help Listings

The Command and the Parameters

Each command has up to four divisions in its Help listing. The first is the command with all of its parameters. There is an explanation for the parameters as well as an indication as to which are required and which are optional.



The hierarchy of the symbols is

- { | } Choose one of the options between the {}
- {} | {} Chose one of the options from the first {}-group or the second {}-group
- [] [] [] Choose options from the first []-group and/or the second []-group and/or the third []-group. The choices must appear in order.

Here is a command we can use:

getpizza {index | string | CURRENT} {string | USUAL} [string | THICK | THIN] [timeval] [NOTES filespec]

The meanings of the delimiters and words are:

{ }	necessary
[]	optional
 	or
index	integer value corresponding to a position in a list (usually zero-based)
string	string variable or information in quotes
UPPER_CASE	Use the word as is with special meaning. Since ASPECT is not case-sensitive, the actual word within the script need not be capitalized.
filespec	file specification in quotes or string variable

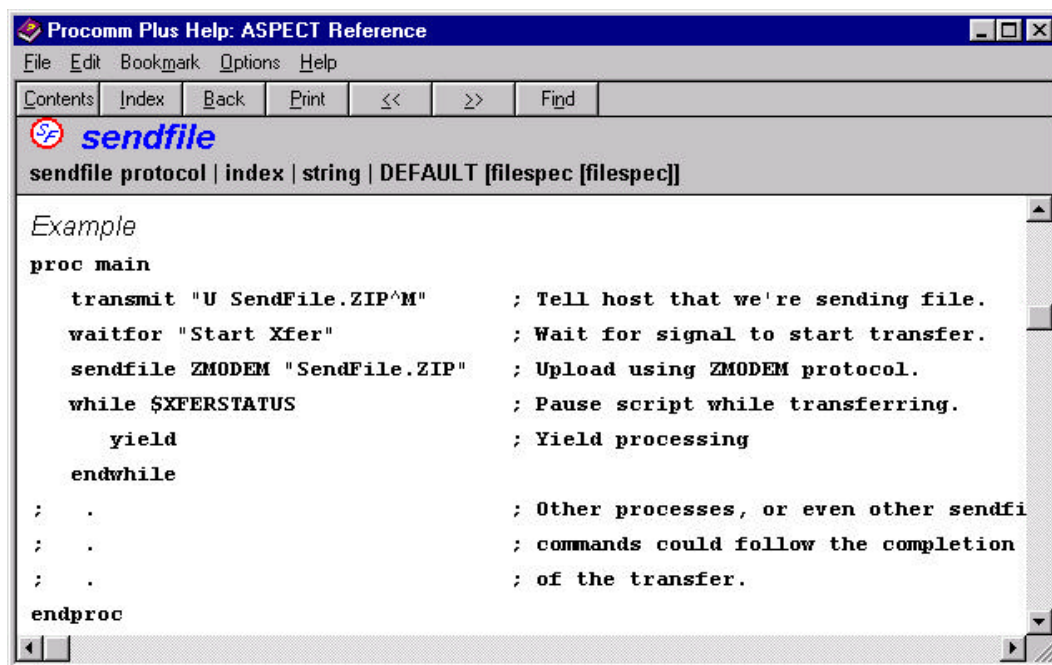
The generic variables are described in the order they appear.

Index	the index of the pizza shop as it appears in Setup
string	name of the pizza shop
CURRENT	the pizza shop selected in Current Options
string	flavor of desired pizza
USUAL	the pizza shop will use what you have in your preferences which are stored in their records
string	crust type
THICK	thick crust
THIN	thin crust
timeval	delivery time – if no time is specified, delivery will be ASAP
NOTES	key word signaling that a notes file exists which will be

	transmitted to the pizza shop
filespec	file name for the NOTES file

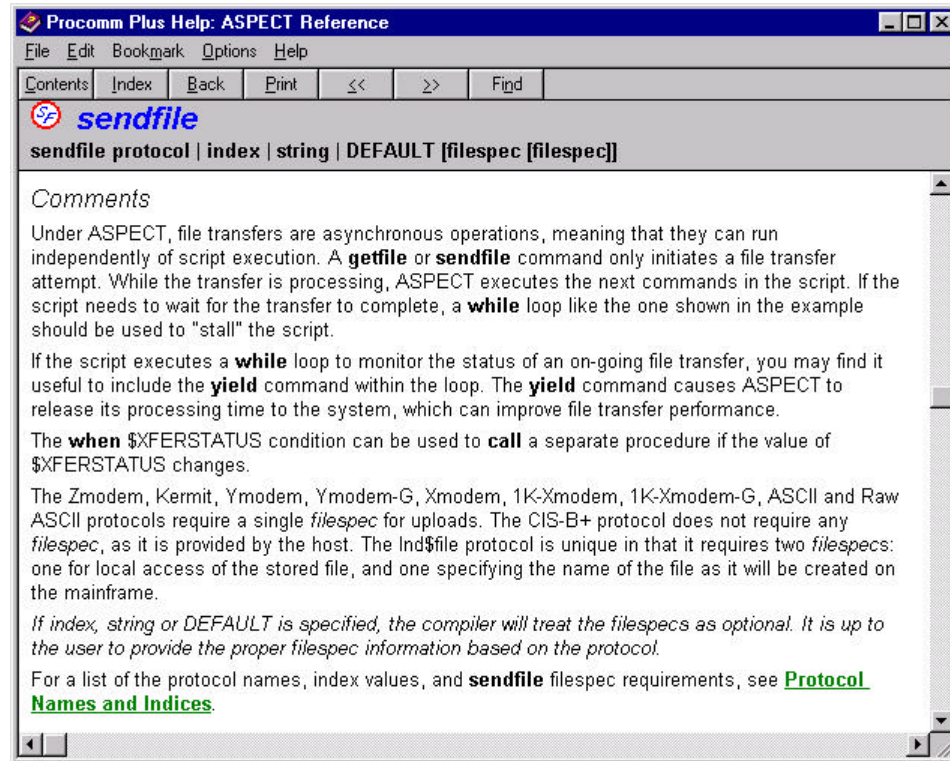
Source Code Example

The second part of the Help listing is usually an example. This may be a complete procedure that can be compiled and executed or it may be a snippet of code that may be pasted into your script. Not all commands have examples; some of the commands may refer you to other commands to see the example in that listing.



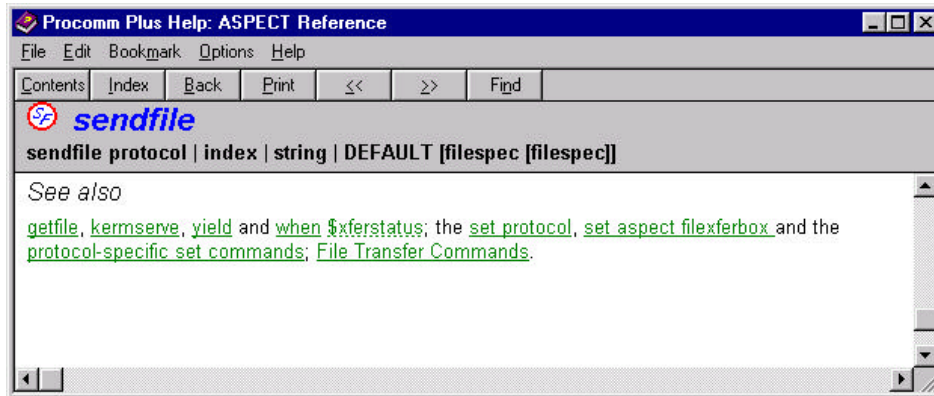
Comments

The third part is comments. This may be just a few lines or several pages of information.



See Also

The fourth part is “See Also”. This is extremely useful if you almost know what command you want but you may want to check out a few closely related commands. Alternatively, there may be commands that augment the usefulness of the command you are looking at. These are all active links so that you can click on them and get to the extended information.



Chapter 4 - Logon Scripts

A great deal of the time, Procomm is used to log on to a remote site and transfer a file. This can be achieved with little or no human intervention.

Preliminaries

Let's assume we are going to create a script to connect to a remote system and transfer a file. This may be connecting to Blue Cross several times a day to transfer insurance claims. Another example would be to dial in to NAPA headquarters the first thing every morning and download the list of parts that are being shipped today to your auto repair shop.

Before you write the script, several questions must be answered:

- Do you want to use the Connection Directory to make the connection?
- Do you want to create an icon on the desktop to do the entire operation?
- Can you make the connection and do the file transfer manually?
- Should the script control the emulation, protocol and line settings?
- What will be the level of user intervention?

- How much error checking and recovery do you need?
- What is the “computer comfort level” of the person who will run this script?

Recording a script

The simplest way to get a basic script is to make the connection manually and record all the requests the remote system sends to you and all the key strokes you type in from this side. As we will see later, this script almost always needs some editing and fine-tuning but it is easier than starting from scratch.

Recorder Icon Button on the Action Bar



This button with the “camcorder” is at the right end of the Action Bar. It may be necessary to scroll the Action Bar to see the button. If we first assume that we will be using the Connection Directory to make the connection, then you should click on the Recorder Icon Button before you click on the entry in the Connection Directory. After you click on the

Recorder button, it will change to an animated icon with a person holding the camcorder and there will be a red light moving on the camcorder.

Connection Directory

Choose the entry from the Connection Directory and make the connection. This assumes that you have set up the entry with the necessary emulation, baud, line settings, etc.

Manual Logon and File Transfer

Perform your normal logon and file transfer actions. The recorder will be retaining the queries from the remote system as well as your keystrokes.

Log Off and Terminate the Recorder

Generally, you will want the script to log off the remote system and hang up the telephone. Do these actions and then click on the Recorder Icon Button to end the recording. You will be asked to name the script. The default name is RECORD.WAS but it is usually preferable to give the script a name you can associate with the remote system.

Edit the Script

Here is the script I get from recording a session to the Symantec BBS.

```
;Recorded script. Editing may be required.
proc main
  waitfor "^@^@^@^@Please enter your FULL name: "
  transmit "no name^M"
  waitfor "^@^@^@^@Is this you? "
  transmit "y"
  waitfor "^@^@^@^@Enter Your Password: "
  transmit "password^M"
  waitfor "-Press Any Key-"
  transmit "^M"
  waitfor "                ^M^J"
  transmit "f"
  waitfor " ^[[47m ^[[0;30;47mÄ^[[1;37mÄÄÄÄÄÄÄÄÜ ^[[44m"
  transmit "d"
  waitfor "^[[44m"
  transmit "10^M"
  waitfor "-More-"
  transmit "^M"
  waitfor "Press <Enter> by itself to exit. "
  transmit "^M"
  waitfor "File Name? "
  transmit "examples.exe^M"
  waitfor "Choose one (Q to Quit): "
  transmit "z"
  waitfor "-More-"
  transmit "^M"
  waitfor "Press <Enter> by itself to exit. "
  transmit "^M"
  waitfor " ^[[47m ^[[0;30;47mÄ^[[1;37mÄÄÄÄÄÄÄÄÜ ^[[44m"
  transmit "g"
  waitfor ";30;41mÄ^[[1;37mÄÄÄÄÄÄÄÄÜ ^[[44m^M"
  transmit "1"
endproc
```

There are a few things to notice about this script. First, it consists solely of two commands: TRANSMIT and WAITFOR. One of the useful things to know about these two commands is that they haven't changed since the initial version of ASPECT. A simple logon script, recompiled, will usually work in any version of Procomm. Second, the script could use some judicious editing to make it easier to read and more robust.

WAITFOR Command

waitfor string [integer | FOREVER] [MATCHCASE] [RAW] [STRIP]

The first parameter in this command is the string of characters that the remote system will be sending to you. The first information the script above is waiting for is

```
"^@^@^@^@Please enter your FULL name: "
```

This consists of several nulls (see Caret Translation below) followed by "Please enter your FULL name: (space) ". The rule of thumb for a **waitfor** command is to specify the least amount of characters possible. The reason for this is that ASPECT will require a complete match of the entire string; if there is a noisy connection, you can increase your chances of a match by specifying fewer characters. You should especially be careful of including spaces in your **waitfor** strings because you may not see them or count them exactly right. In this case, you can edit the string to be "name" or "name:".

The second parameter is a specification of how long to wait. If there is nothing specified, the default is 30 seconds. You may give a number of seconds or use the keyword **FOREVER**. The latter really does mean forever so, if the string is not received, the script could look like it is hung up.

The next parameter is **MATCHCASE**. This would require the characters from the remote system to exactly match the case of the listed string.

The **RAW** parameter would allow the incoming data to be retrieved unchanged, bypassing caret translation, the Translate Table and 8th-bit stripping if it is enabled for the current emulation.

If **STRIP** is specified, the target characters would be stripped from the incoming data stream before they reach either the terminal screen or ASPECT.

This command sets the **SUCCESS/FAILURE** flags. It is a **success** if the target string is received within the time specified. It is a **failure** if the

target string is not received and the script progresses to the next ASPECT command only because the time limit was exceeded.

Now, understanding the **waitfor** command, we can clean up the logon script for the QD BBS. The first pass would remove the null (^@) characters and reduce the recognizable prompts to something unique. The second pass is a little more time-consuming. Because the QD BBS uses ANSI emulation escape codes to display the menus, the recorder captured some of these codes. These look like ^[[1:37m, ^[[0;30;47m, etc. While these would function adequately for the **waitfor** command, they make the script look like it is written in Sanskrit! A better solution would be to watch the file transfer procedure again, write down the last word in each menu and use that as the target. Now is also a good time to add comments.

The resultant edited and commented script now looks like this:

```
;Recorded script. Editing may be required.
;Logon script for QD BBS
proc main
  waitfor "name"           ;name and password
  transmit "no name^M"
  waitfor "you?"
  transmit "y"
  waitfor "Password"
  transmit "password^M"
  waitfor "-Press Any Key-"
  transmit "^M"
  waitfor "Settings"       ;first menu
  transmit "f"
  waitfor "Settings"       ;file transfer menu -- choose download
  transmit "d"
  waitfor "<ENTER>"         ;libraries -- choose 10
  transmit "10^M"
  waitfor "-More-"
  transmit "^M"
  waitfor "exit"           ;activity list -- choose download
  transmit "d^M"
  waitfor "Name?"         ;request for file name
  transmit "examples.exe^M"
  waitfor "Quit):"        ; which transfer protocol -- ZMODEM
  transmit "z"
  waitfor "-More-"
  transmit "^M"
  waitfor "exit"          ;activity list -- choose to exit
  transmit "^M"
  waitfor "Settings"      ;file transfer menu -- choose goodbye
  transmit "g"
  waitfor "two"           ;goodbye menu -- confirm
  transmit "1"
endproc
```

An important consideration of the **WAITFOR** command is that it must be actually waiting before the string it is waiting for appears. The portion of code below is an example of bad programming practice. The script is displaying a user message and waiting for the user to click on "OK". The string "Enter your password" will be sent from the remote system but the script will not see it. After the user clicks on "OK", the script then proceeds to the WAITFOR line but it is too late to see the transmission.

```
waitfor "Enter your name"
transmit MyName
usermsg "Name Sent"
waitfor "Enter your password" FOREVER
```



```
transmit MyPassword
```

This is an exaggerated example but it illustrates the problem. If there is a need for user input, it should be in the script at a point where it does not interfere with WAITFOR commands. An easy place for this is at the beginning of the script. Another place would be immediately after a WAITFOR command because there is usually a short period of time before the remote system expects a response.

TRANSMIT Command

transmit string [RAW]

The first parameter is the string to be transmitted. The second parameter, not required, is whether to transmit the string exactly as it is or to do Caret Translation. See below for a discussion of Caret Translation.

It is important that you know and specify whether a carriage return is to be used at the end of the transmission. You will see this included as a **^M** in some of the transmissions. Including an unneeded carriage return can cause as many problems as not including a necessary one.

Note that the strings for the transmissions can not be masked. Thus, the password is available to anyone who can read the source code. One way around this situation is to provide only a compiled version of the script to users who don't need to know the password.

Caret Translation

The first 32 characters of the ASCII table are non-printable characters. These are functions such as carriage return, line feed, form feed, escape, bell and others. These are keyed in as Control - Letter and this is represented by Caret - Letter in ASPECT and all of Procomm. The caret is ^ (shift 6). Some of the common functions are:

- ^M – carriage return
- ^L – form feed
- ^J – line feed
- ^G – bell
- ^[-- escape
- ^H – backspace
- ^@ -- null

Improvements

The version of the script that is listed above will function adequately if there are no problems. You may wish to add some user-friendly or error checking features. We will cover simple user I/O in the next chapter but here are some things you might want to consider:

- Ask the user for name and password
- Obtain the name and password from the Connection Directory
- Ask the user for the file name
- Check that the file transfer was successful
- Provide an alternative route if the file transfer failed
- Make the connection in the script, not with the Connection Directory
- Choose the modem and make all its settings
- Specify the emulation
- Allow the user to change transfer protocol both on the local system and the remote system

Exercise 4a



Record a logon script to connect to the QD BBS and download a file. Edit it to clean it up and add comments.

Chapter 5 - Simple I/O

Eventually, you will want to display a message to the user or query the user for input. ASPECT provides many ways to do this. In this manual, we will cover some of the simplest methods. The Intermediate Manual will include descriptions and exercises covering Dialog Boxes which allow your scripts to use pushbuttons, edit boxes and drop-down lists as well as other standard Windows GUI features.

Standard Dialog Input Box



This input window was obtained with the following code. Granted, this isn't enough code to utilize the input information but you can see it is an easy command to use.

```
proc main
```

```
string InputString  
  
sdlginput "Beginning ASPECT Class" "Enter your name" InputString  
  
endproc
```

SDLGINPUT Command

sdlginput title prompt strvar [strlength] [MASKED] [DEFAULT]

The first parameter is a string for the title. This is what appears in the title bar of the input window.

The second parameter is a string for the prompt. This appears over the edit box and instructs the user what to write in the edit box.

The next parameter is a string variable that will contain the user-input after the user clicks on the OK button. Remember that all local variables must be declared at the top of the procedure.

The **MASKED** feature can be used to mask the input. This could be used if the user is entering a logon or a password.

If **DEFAULT** is specified, the current value of the string variable will be displayed in the edit box. This could be used to suggest a default input to the user.

User Message Window



This user message window was generated using the simplest features of the **usermsg** command as shown in the code below.

```
proc main  
  
    usermsg "ASPECT is a wonderful language!"  
  
endproc
```

USERMSG Command

usermsg string | {formatstr arglist}

The most straightforward use of this command is to supply a string as in the example above.

The most versatile use of this command is to provide a string that contains some formatting information and follow this with the arguments to be formatted. All output to the screen is in string format so this converts numbers to strings on the fly. This is also a simple way to join two or more strings together.

Some of the Format Specifiers

- **d or i** The parameter is converted to signed decimal notation.

- **s** The string parameter is displayed up to the first null character or until the precision value is reached.
- **f** The parameter is output in the form [-]dddd.dddd, where dddd is one or more decimal digits. The number of digits displayed after the decimal point is determined by the precision specification. The default precision is 2. An example would be %6.3f which specifies 3 decimal digits with a maximum of 6 digits.
- **c** The integer parameter is output as a single ASCII character.
- **l** A lower case "l" may be used before the d, i, o, x, or X format specifiers to specify that the corresponding argument is a long value.

Escape sequences

Another ingredient in formatting is non-printing characters, such as Carriage Return and Line Feed. These can make the output easier to read. This form of command doesn't use the Caret Translation that was covered in the previous chapter. For this command, they are accomplished with the use of the backtick, ```, in combination with a letter. The backtick is usually the key just to the left of the 1 key. All the possibilities for non-printing characters are in Help under Escape Sequences. Here are a few of the most common:

<code>`n</code>	New line/line feed
<code>`r</code>	Carriage return
<code>`f</code>	Page feed
<code>`"</code>	Quote

Examples

```
proc Main
    integer NumTotal

    NumTotal = 35 + 926
    usermsg "The total of the numbers is`n`n`r%i" NumTotal
```

```
endproc
```



This example uses two line feeds (`\n\n`) and one carriage return (`\r`) to space the output. Notice that there are no spaces between the `\r` and the `%i` so that the 961 starts exactly at the beginning of the line.

```
proc Main  
  
    float AverageNum  
  
    AverageNum = (32.8 + 48.3 + 97.5) / 3.00  
    usermsg "The average of the numbers is %8.3f" AverageNum  
  
endproc
```



Here the number being formatted is a floating-point number. In the format string, there is only one space between the `is` and `%`. However, in the output there are 2 blank spaces because of the `8.3f` formatting.

Exercise 5a



Use 2 standard dialog input boxes to query the user for first name and last name. Output the results with the last name 2 lines below the first name.

Exercise 5b



Modify your logon script to query the user for the name of the file to be downloaded.

Chapter 6 - DOS Commands

These commands can be used to copy, delete or rename files just as you would in DOS. In fact, nearly all of the DOS commands for file and directory manipulation can be duplicated with ASPECT commands. Those that aren't duplicated can be performed directly with the ASPECT **DOS** command but that is material for another chapter!

If you regularly upload or download files, you will want to be able to keep track of your efforts. You may move files to another directory after they have been sent. You could even have a script that is scheduled to start up every hour and transfer any files that are in the upload directory then move the successfully transferred files. You could log on to a remote site and download today's information, renaming the file with today's date.

File-moving Commands

COPYFILE Command

copyfile FromFileSpec ToFileSpec

copyfile works similar to the DOS COPY command. However, copyfile does not support wildcards. Script execution is suspended until the file is copied; this is unusual, as most commands are asynchronous. The FileSpec should be the fully qualified path and filename in order to avoid surprises.

One of the syntax conventions is the ASPECT Help files is that **filespec** can contain the fully qualified path or just a file name. When a command specifies **filename**, it is asking for the name without any path information and it will assume the current applicable directory. This may be the upload, download, programs or ASPECT directory depending on what command you are using.

```
proc Main

  copyfile "c:\program files\procomm plus\pw4.dir" "c:\program files\procomm plus\upload"
  if SUCCESS
    usermsg "File copied OK"
  else
    usermsg "File copy failed"
  endif

endproc
```

Notice, in this example, the **copyfile** commands obeys common DOS practices. I specified a file name in the first parameter but only a directory name in the second parameter.

DELFILE Command

delfile FileSpec

This command will delete a file. The filespec may include a full directory path. If the path isn't specified, the current User Path is assumed. Wildcards are not allowed for the file specification.

RENAME Command

rename FromFileSpec ToFileSpec

If no path is supplied, rename searches the User Path for the source file. Files can be renamed to a different directory on the same drive, which is equivalent to moving them from one directory to another.

Directory Commands

CHDIR Command

chdir pathname [ASPECTPATH]

This command will change to the specified path or disk. Note, this is different behavior from the DOS **chdir** command. Optionally, the new path may become the ASPECT directory that is specified in the Setup Options.

GETDIR Command

getdir diskID strvar

This command returns the current working directory on the specified disk. DiskID is an integer number designating the drive. 0 indicates the current drive, 1 specifies drive A, 2 for drive B and so on, up to 26 for Z. This will return the current User Path referred to in previous commands.

MKDIR Command

mkdir pathname

This command will create a new directory following the standard DOS rules. If no path is specified, then the new directory will be a subdirectory of the current directory.

RMDIR Command

rmdir pathname

This command will remove an empty directory using the specified path and file name. If no path is specified, a sub-directory of the current directory will be assumed.

DIR Command

dir filespec [strvar]

The **dir** command displays a standard file listing and optionally returns a filename selected by the user. This is an easy way to open

Exercise 6a



Create a new directory (.....\ASPECT\Your name) then copy your pw4.dir file to that directory. Rename the file to YourName.dir.

Exercise 6b



Modify your logon script to copy the downloaded file to this new directory and delete it from the Download directory.

Exercise 6c



List the contents of the new directory and let the user choose one file. Confirm that this was the chosen file with a user message.

Chapter 7 - File Transfers

So far, in this manual, we have done basic downloads using ZMODEM and default settings. In reality, uploads and downloads usually require more settings and error checking. Procomm Plus has many options available to control file transfers.

Transfer Protocols

Protocol	Batch	Autodownload	Error Check	Restart
Zmodem	y	y	y	y
Kermit	y	y	y	n
Xmodem	n	n	y-n	n
Ymodem	y-n	n	y-n	n
ASCII	n	n	n	n

Zmodem

This is one of the most common protocols and it is very easy to use. It is usually available on all PCs and even some UNIX or VAX machines. Zmodem is the only one that has a restart capability; that is, if a transfer is

aborted before it finishes, zmodem can detect this and just transfer the remainder of the file. If the option is activated, zmodem can do an autodownload. This means that a file can be received with no user interaction. The reason for this is that the file name and size are carried in the first packet of a zmodem transfer so the receiving system has the information it needs to store the file.

Kermit

Kermit is slower and more difficult to use but it is also the only transfer protocol that can exist on every operating system known to mankind. People use Kermit when there is no other protocol common to both the sending and receiving system. Kermit, like zmodem, can do an autodownload because it also carries the necessary information in the first packet of the file transfer. Since the file name is part of the transfer, it is not possible to rename the file during the transfer for either Kermit or zmodem.

Xmodem and Ymodem

These protocols exist in several configurations and are almost completely only on PCs. Some of the versions do batch transfers but none of them does autodownloading. Some of the versions don't do error checking; instead, they rely on the error correcting modem to handle all of this.

File Transfer Commands

SENDFILE Command

sendfile protocol | index | string | DEFAULT [filespec [filespec]]

The uploading protocol may be specified by inserting the actual word or with a variable that refers to the protocol. There is a list of the protocols and their indices in the Help files if you want to specify a protocol that way. The key word DEFAULT will allow you to use the current protocol.

The first **filespec** is the name of the file on the local machine. It is required for all protocols. However, if you don't use the protocol name, then the compiler will treat this as non-required and may not flag the error.

The second **filespec** is the name the file will have on the remote system. This is required only for Ind\$File transfers involving an IBM mainframe computer.

GETFILE Command

getfile protocol | index | string | DEFAULT [filespec[filespec]]

This command is similar to the **sendfile** command. The first **filespec** is the name the file will have on the local machine after the file is received. The second **filespec** is used only for Ind\$File transfers.

Related System Variables

`$XFERSTATUS`

File transfers are asynchronous with respect to ASPECT. This means that the script will continue to the next command after the file transfer is started. Usually, we want to know if the transfer was successful or not so that we can notify the user or take corrective measures. The system variable `$XFERSTATUS` will report on the file transfer activity.

The values for `$XFERSTATUS` are:

- | | | | |
|---|----------------------------|---|-------------------------------|
| 0 | not busy | 2 | successfully sent or received |
| 1 | busy, sending or receiving | 3 | transfer aborted |

```
iTemp = 1                ;Set iTemp value to 1 (for loop.)
sendfile ZMODEM "sendfile.txt"
while iTemp <= 1          ;While transfer is going on.
    iTemp = $XFERSTATUS    ;Store value in $XFERSTATUS in iTemp
    yield                 ;Yield processor time.
endwhile
FileTransferName = $XFERFILE
if iTemp == 2              ;Test the value of iTemp for success
    usermsg "File successfully sent -- %s" FileTransferName
else                       ;or failure of file
    errormsg "File transfer failed!."
endif
```

The reason for writing the WHILE-loop this way is so that we can determine if the file transfer was successful or not. Once `$XFERSTATUS` is set to 2 (successful) or 3 (failed) within the WHILE loop and it is accessed by the script, `$XFERSTATUS` is set back to zero. This is why it is important to use the parameter `iTemp`; it preserves the next to the last value of `$XFERSTATUS`. When the file transfer has completed, the value of `$XFERSTATUS` is zero (not busy) but we want to know if the transmission was successful or not.

The yield command inside of the while loop releases CPU time to the transfer process. Otherwise, the computing power is tied up in the while loop and the transfer is much slower.

If the file being transferred is small, as it has been in our Exercises up to now, then the asynchronous nature of the file transfer will not be observed. The file will transfer within the time limit of the subsequent WAITFOR command and all will seem OK. The problems appear with the transfer of large files, especially if the transfer fails.

\$XFERFILE

This system variable contains the name of the transferred file. This can be useful if you do a regular transfer of the contents of a directory and you need to know the name of the files.

Exercise 7a



Modify your logon script to report on the success of the file transfer and the name of the transferred file.

Appendix A - Exercise Scripts

Exercise 4a



Record a logon script to connect to the QD BBS and download a file. Edit it to clean it up and add comments.

```

*****
;
;* Exercise4a
;*
;*
;* MAIN
;* The Main procedure is modified from a recorded script to the Qdeck BBS.
;*
;*
;* Calls: none
;* Modifies globals: none
*****
proc Main
    waitfor "name"          ;name and password
    transmit "no name^M"
    waitfor "you?"
    transmit "y"
    waitfor "Password"
    transmit "password^M"
    waitfor "-Press Any Key-"
    transmit "^M"
    waitfor "Settings"      ;first menu
    transmit "f"
    waitfor "Settings"      ;file transfer menu -- choose download
    transmit "d"
    waitfor "<ENTER>"        ;libraries -- choose 10
    transmit "10^M"
    waitfor "-More-"
    transmit "^M"
    waitfor "exit"          ;activity list -- choose download
    transmit "d^M"
    waitfor "Name?"         ;request for file name
    transmit "examples.exe^M"
    waitfor "Quit):"        ; which transfer protocol -- ZMODEM
    transmit "z"
    waitfor "-More-"
    transmit "^M"
    waitfor "exit"          ;activity list -- choose to exit
    transmit "^M"
    waitfor "Settings"      ;file transfer menu -- choose goodbye
    transmit "g"
    waitfor "two"           ;goodbye menu -- confirm
    transmit "1"
endproc

```

Exercise 5a

Use 2 standard dialog input boxes to query the user for first name and last name. Output the results with the last name 2 lines below the first name.

```

*****
/*
/* Exercise5a
/*
/*
/* MAIN
/* The Main procedure queries for the first name and the last name then
/* reports the result.
/*
/*
/* Calls: none
/* Modifies globals: none
*****
proc Main
    string FirstName      ;variable for first name
    string LastName       ;variable for last name

                                ;query for first and last name
    sdlginput "Exercise 5a -- Part 1" "Enter your first name" FirstName
    sdlginput "Exercise 5a -- Part 2" "Enter your last name" LastName

                                ;output the full name
    usermsg "Your complete name is:\n'r%s\n'n'r%s" FirstName LastName

endproc

```

Exercise 5b

Modify your logon script to query the user for the name of the file to be downloaded.

```

*****
;* Exercise 5b
;*
;* MAIN
;* The Main procedure is modified from a recorded script to the Qdeck BBS.
;* The user is queried for the name of the file to download.
;*
;* Calls: none
;* Modifies globals: none
*****

proc Main

    string NameOfFile

    waitfor "name"          ;name and password
    transmit "no name^M"
    waitfor "you?"
    transmit "y"
    waitfor "Password"
    transmit "password^M"
    waitfor "-Press Any Key-"
    transmit "^M"
    waitfor "Settings"      ;first menu
    transmit "f"
    waitfor "Settings"      ;file transfer menu -- choose download
    transmit "d"
    waitfor "<ENTER>"        ;libraries -- choose 10
    transmit "10^M"
    waitfor "-More-"
    transmit "^M"
    waitfor "exit"          ;activity list -- choose download
    transmit "d^M"
    waitfor "Name?"         ;request for file name
    sdlginput "File Transfer Example" "Name of file to download" NameOfFile
    transmit NameOfFile     ;send the name of the file
    transmit "^M"           ;send a carriage return
    waitfor "Quit):"        ; which transfer protocol -- ZMODEM
    transmit "z"
    waitfor "-More-"
    transmit "^M"
    waitfor "exit"          ;activity list -- choose to exit
    transmit "^M"
    waitfor "Settings"      ;file transfer menu -- choose goodbye
    transmit "g"
    waitfor "two"           ;goodbye menu -- confirm
    transmit "1"
endproc

```

Exercise 6a

Create a new directory (.....\ASPECT\Your name) then copy your pw4.dir file to that directory. Rename the file to YourName.dir.



Note: the first example assumes the script is run from the\aspect directory and so it can use short notations to refer to a subdirectory. The second example can be run from any directory.

```

*****
;* Exercise 6a
;*
;*
;* MAIN
;* Creates a new directory (..\ASPECT\Your name) then copies pw4.dir
;* file to that directory. Renames the file to YourName.dir.
;*
;* Calls: none
;* Modifies globals: none
*****
proc Main

    mkdir "Kay"
    chdir "Kay"
    copyfile "\program files\procomm plus\pw4.dir" "pw4.dir"
    rename "pw4.dir" "kay.dir"

endproc

```

OR

```

proc Main

    mkdir "\program files\procomm plus\aspect\Kay"
    copyfile "\program files\procomm plus\pw4.dir" "\program files\procomm plus\aspect\kay\kay.dir"

endproc

```


Exercise 6b

Modify your logon script to copy the downloaded file to this new directory and delete it from the Download directory.

```

*****
/
/*
/
/* Exercise6b
/
/*
/
/* MAIN
/* The Main procedure is modified from a recorded script to the Qdeck BBS.
/* The change for 6b is to copy the downloaded file from the Download directory
/* to a different directory and delete it from the Download directory.
/*
/
/* Calls: none
/* Modifies globals: none
/*
/
*****

proc Main
    waitfor "name"          ;name and password
    transmit "no name^M"
    waitfor "you?"
    transmit "y"
    waitfor "Password"
    transmit "password^M"
    waitfor "-Press Any Key-"
    transmit "^M"
    waitfor "Settings"      ;first menu
    transmit "f"
    waitfor "Settings"      ;file transfer menu -- choose download
    transmit "d"
    waitfor "<ENTER>"        ;libraries -- choose 10
    transmit "10^M"
    waitfor "-More-"
    transmit "^M"
    waitfor "exit"          ;activity list -- choose download
    transmit "d^M"
    waitfor "Name?"         ;request for file name
    transmit "examples.exe^M"
    waitfor "Quit):"        ; which transfer protocol -- ZMODEM
    transmit "z"
    waitfor "-More-"
    transmit "^M"
    waitfor "exit"          ;activity list -- choose to exit
    transmit "^M"
    waitfor "Settings"      ;file transfer menu -- choose goodbye
    transmit "g"
    waitfor "two"           ;goodbye menu -- confirm
    transmit "1"

    copyfile "\program files\procomm plus\download\examples.exe" "kay\examples.exe"
    delfile "\program files\procomm plus\download\examples.exe"

endproc

```

Exercise 6c

List the contents of the new directory and let the user choose one file. Confirm that this was the chosen file with a user message.

```

*****
/* Exercise 6c
/*
/*
/* This script lists the contents of the new directory and lets the user
/*   choose one file. The choice is confirmed with a user message.
/*
/*
/* Calls: none
/* Modifies globals: none
*****
Proc Main
  string ChosenFile

  chdir "\program files\procomm plus\aspect\kay"
  sdglopen "Exercise 6 -- List of files" "*" SINGLE ChosenFile

  usermsg "You chose %s" ChosenFile

endproc

```

Exercise 7a

Modify your logon script to report on the success of the file transfer and also the name of the transferred file.

```

*****
;* Exercise 7a
;*
;* The Main procedure is modified from a recorded script to the Qdeck BBS.
;* The change for 7a is to monitor and report on the success/failure of
;* the file transfer and also report the name of the transferred file.
;*
;* Calls: none
;* Modifies globals: none
*****
proc Main
  integer iTemp = 1
  string FileTransferName
  waitfor "name"          ;name and password
  transmit "no name^M"
  waitfor "you?"
  transmit "y"
  waitfor "Password"
  transmit "password^M"
  waitfor "-Press Any Key-"
  transmit "^M"
  waitfor "Settings"      ;first menu
  transmit "f"
  waitfor "Settings"      ;file transfer menu -- choose download
  transmit "d"
  waitfor "<ENTER>"        ;libraries -- choose 10
  transmit "10^M"
  waitfor "-More-"
  transmit "^M"
  waitfor "exit"          ;activity list -- choose download
  transmit "d^M"
  waitfor "Name?"         ;request for file name
  transmit "examples.exe^M"
  waitfor "Quit):"        ; which transfer protocol -- ZMODEM
  transmit "z"
  while iTemp <= 1        ;While transfer is going on.
    iTemp = $XFERSTATUS   ;Store value in $XFERSTATUS in iTemp
    yield                 ;Yield processor time.
  endwhile
  FileTransferName = $XFERFILE
  waitfor "-More-"
  transmit "^M"
  waitfor "exit"          ;activity list -- choose to exit
  transmit "^M"
  waitfor "Settings"      ;file transfer menu -- choose goodbye
  transmit "g"
  waitfor "two"           ;goodbye menu -- confirm
  transmit "1"
  if iTemp == 2           ;Test the value of iTemp for success
    usermsg "File successfully received -- %s" FileTransferName
  else                   ;or failure of file
    errormsg "File transfer failed!."

```

```
endif  
endproc
```