xLibrary User Manual (Version 1.0.1)

Contents

Overview.

xLibrary consists of two HyperCard™ stacks, **xLibrary** and **xLibrary Installer**. It provides a simple and secure way of connecting to a variety of remote 'services'. Although originally intended as a tool for librarians to connect to remote Online Public Access Catalogues (OPACs) the definition of a 'service' has been broadened to be any process running on a local or remote computer.

The **xLibrary Installer** stack allows the definition, testing, storage, installation and removal of service definitions. A service definition is a set of parameters that together define the external parameters of a 'service'. Once the service definition has been created and tested it may then be installed into the **xLibrary** stack. In this fashion an individualised **xLibrary** stack can be created for each environment.

The **xLibrary** stack allows for multiple, simultaneous services to be active at any one time. A typical session might include a serial connection to your libraries' OPAC in one window, whilst another window displays information from a 'Campus-Wide Information System' (CWIS), and a third displays a Unix Newsreader. As **xLibrary** utilises the **Communications Toolbox**, connectivity is only limited by the range of Connection and Terminal Emulation tools that are available on your machine.

Each of the stacks contain extensive online help facilities and it is recommended that they be used to complement this document.

Requirements.

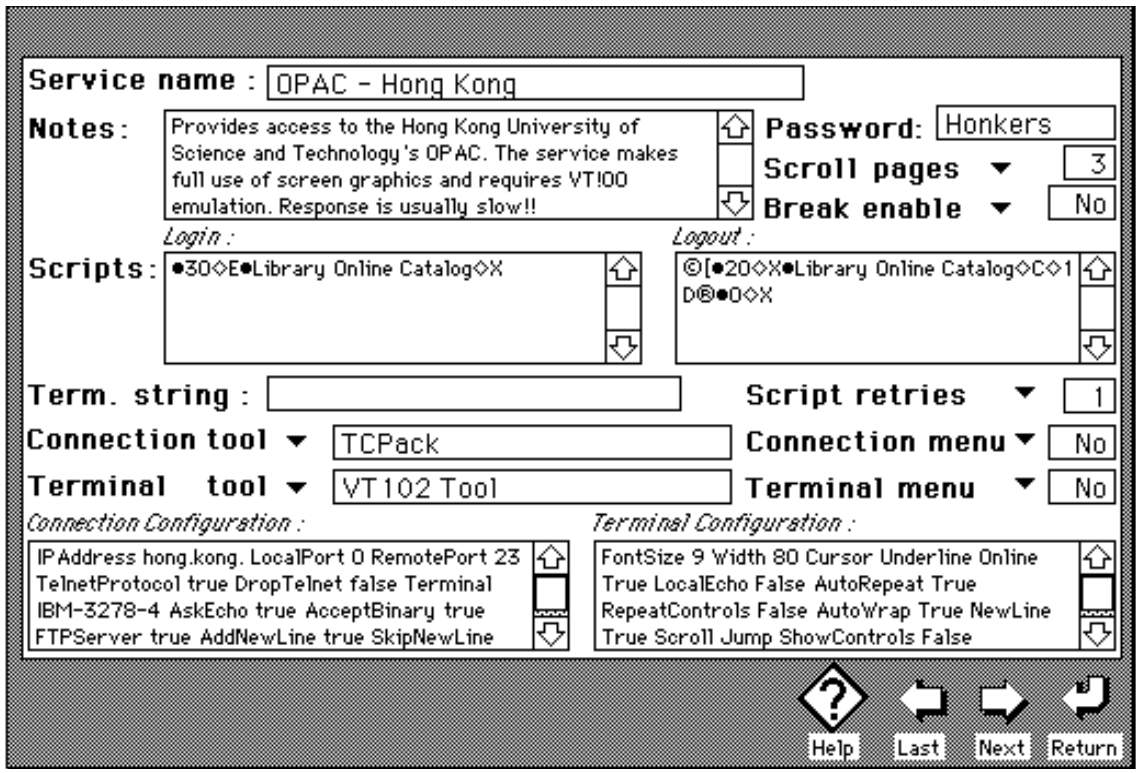
xLibrary requires a system with the Communications toolbox installed and HyperCard 2.0 or later.

Definition of Services.

The purpose of the **xLibrary Installer** stack is to define the services that are to be made available via the **xLibrary** stack. Each service is defined by a series of parameters that are held on a single HyperCard card. The parameters are as follows -

1. The name of the service e.g. “Library OPAC”, “NewsReader”.
2. Notes and general information about the service. N.B. Once the service definition has been installed into a copy of **xLibrary** the content of these notes are made available as part of the **xLibrary** online help facility.
3. The communications toolbox Connection tool to be used to connect to the target computer upon which the service is available e.g. “Serial Tool”.
   N.B. The choice of tools available for selection is drawn from the total number of Connection tools that have been installed on your system. Each time a selection is made the operator is presented with a tool specific interface (dialog box) by which the various communications parameters may be specified e.g. for the serial tool the baud rate, parity setting etc.
4. Whether or not any menu associated with Connection tool is to be added to the service menubar.
5. The Communications toolbox Terminal Emulation tool that will be used to display the output generated from the service e.g. “VT102 Tool”.
   The choice of tools available for selection is drawn from the total number of Terminal Emulation tools that have been installed on your system. Each time a selection is made the operator is presented with a tool specific interface (dialog box) by which the various emulation parameters may be specified e.g. for the VT102 the character set, cursor type etc.
6. Whether or not any menu associated with Terminal Emulation tool is to be added to the service menubar.
7. The script to be used to log in to the required service once connection has been made with the target computer (see section on scripting).
9. The script to be used to log out from the required service when the service is closed (see section on scripting).
10. The password to be used should it be required to password protect the service.
11. Whether or not the break key menu item is to be enabled whilst the service is active.
12. The number of screen pages to be scrolled by the Terminal Emulation tool being used.
13. A character string that will, upon detection signal the closure of the service. That is, **xLibrary** will check all input from the service against this parameter string. If a match is found then the stack will terminate the service by executing the logout script etc.
14. The number of times to retry a line of script before an error is returned.

The following is an example of a service definition card that provides accesses to the Hong Kong University Library OPAC.



The installer stack provides facilities for both creating new service definitions and editing existing definitions. The stack also provides facilities for creating and modifying the scripts associated with a service and for activating the service as a means of testing the definition.

Once a service definition has been created and tested it may then be installed into a copy of the ***xLibrary*** *stack.*

Activating a Service.

Once the service definition has been installed into a copy of the **xLibrary** stack its name will appear as an option in the ‘Go’ menu. A service may then be activated simply by selecting the associated menu option. Services can be defined in the menu to a limit of 10.

When a service is selected a connection is firstly made with the target computer. If the connection is successful the login script (if one has been defined) is executed and it's progress is depicted by a ‘Progress bar’. The login script may be cancelled at any time but if the script is run to successful completion then a new window is created for the service and this becomes the active window. If this is the first window activated then the ‘Windows’ menu is added to the menubar. If this is not the first window opened then the ‘Windows’ menu is updated to reflect the new situation. The ‘Windows’ menu provides options for manipulating the window and generally reflects the status of the **xLibrary** session. The service activation process may be cancelled at any time by typing 'Command-Period'.

A popup menu of single line macros may be associated with each active service window. The menu (if it has any items) will appear immediately below the window bar of the active window if the mouse is clicked in the window bar whilst the ‘option’ key is depressed. Items may be added to the popup menu by selecting the ‘Macros’ item from the edit menu. Each macro item is limited to 255 characters in length and may be modified or deleted by selecting the required item whilst the command key is depressed. The number of macros per service window is limited to 11. The macros may be mapped to one of two sets of keyboard keys, the extended keyboard 'Funtion keys' or the standard keyboard 'Numeric key pad' keys. This option is controlled by the 'Use Key Pad' item in the ‘File’ menu.

A text editing facility is provided as an additional option that appears at the bottom of the ‘Go’ menu. The intention of the facility is to allow the accumulation of selections from the other (service) windows. Selected text may either be copied and pasted into the window or the current service window selection can be appended to the text window via the use of the ‘Windows’ menu item, 'Accumulate Selection'. The operator has the option of importing a HyperCard file at the time that the text window is opened. This option is controlled by a toggled item in the ‘File’ menu. The intention of the option is to allow for continuity between **xLibrary** sessions.

A service window may be closed, and the service terminated in number of ways. The most direct manner is to select the closure item from the ‘Windows’ menu, this is equivalent to clicking the ‘close’ box of the window. The service may also be closed in response to the closure of the remote end of the connection, as in the case of a service ‘timing out’. The final way that a service can be closed is if **xLibrary** detects the termination string (if one has been defined for the service) in the input received from the service.

The text window may be closed by either selecting the window closure option from the ‘Windows’ menu or clicking the close box of the window. When the last window has been closed the ‘Windows’ menu is dropped from the menubar.

xLibrary Menu Options

The following chapter describes those menu items that have either been added by **xLibrary** or that have been implemented differently from the standard HyperCard implemention. All other menu items are as per HyperCard.

File Menu

Import Text File

If this menu option is toggled on, (represented by a tick mark preceeding the option) then a HyperCard file will be prompted for whenever the 'Accumulated Selections' text window is opened. If a file selection is made then its contents will be read into the new window. If the option is toggled off then no file will be prompted for. The setting of this option may be reversed at the time that the text window is opened by depressing the option key when the ‘Accumulated Selections’ item is selected from the ‘Go’ menu.

Use Key Pad

If this menu option is toggled on, (represented by a tick mark preceeding the option) then certain stack functions are mapped to the keys of the numeric key pad. If the option is toggled off and an extended keyboard is present then the functions are mapped to the special keys of the keyboard. The mappings are described by the table below :

Function	Extended Keyboard	Numeric Key Pad
Activate Macros 1-11	F5 through F15	0 through 9 and .
Clear window	del	Clear
Change window - Right	pg up	+
Change Window - Left	pg dn	-
Zoom window		*
Show selection (text window) or scroll to cursor (service window)	end	/

Edit Menu

Undo

Provides the normal 'Undo' feature for the 'Accumulated Selections' text window but it is deactivated for a service window were its action is inappropriate.

Cut

Provides the normal 'Cut' feature for the 'Accumulated Selections' text window but it is deactivated for a service window were its action is inappropriate.

Copy

Provides the normal 'Copy' features for both service and text windows.

Paste

Provides the normal 'Paste' features for both service and text windows.

Clear

Provides the normal 'Clear' feature for the 'Accumulated Selections' text window but it is deactivated for a service window were its action is inappropriate.

Select All

If the window is a 'Accumulated Selections' text window then its entire contents are selected, however if it is a service window then only the current terminal emulation screen is selected, the cached (scrolled) area is not affected.

Edit Macros...

This option is only available for service windows and results in the display of the 'Macros' dialog box. This dialog box allows the definition of single line character strings (Macros) that are to be associated with the window. Once a macro has been defined via the dialog box it is placed in the next position of a popup menu that appears immediately below the window's drag bar when the mouse is clicked in the bar whilst the 'Option' key is depressed. Selecting an macro from this popup menu results in that macro being sent to the service. For further details about macros see 'Macros Menu' below.

Go Menu

This menu contains the names of up to 10 installed services plus the text window option 'Accumulated Selections'. For further information on any of the menu options select that option from the menu whilst depressing the 'shift' key.

Windows Menu

Window Names

The first options in the 'Windows' menu contains the names of the windows that have been activated. The currently active window is denoted by a leading "•" character. Windows that have been hidden are denoted by a leading "◊" character. A visible window may be made the currently active window by a) Clicking anywhere in that window b) Selecting its name from this menu or c) Using one of the special keys for swopping windows as described in the 'Use Key Pad' option in the 'File' menu.

Show Window

This option will only be active if one or more windows have been hidden, and will display and activate the most recently hidden window.

Hide Window

This option will hide the currently active window.

Close Window

Closes the currently active service by logging the service out (if the service has a logout script associated with it) and disconnecting from the target computer.

Clear Window

If the window is an 'Accumulated Selections' text window then its entire contents are cleared, however if it is a service window then only the current terminal emulation screen is cleared, the cached (scrolled) area is not affected.

Break Window

Sends a long break (1 second) signal to the service.

Smooth Scrolling / Faster Scrolling

This option allows the choice between smoother and slower or faster and jerkier text display for a service window. The option is deactivated for the 'Accumulated Selections' text window were the action is inappropriate.

Print Selection

This option will only be active if there is a selection in the currently active service window. The option will cause the selection to be printed.

Print Text

This option will only be active if the 'Accumulated Selections' text window has been opened and contains data. The option will cause the entire contents of the window to be printed.

Accumulate Selection

This option will only be active if there is a selection in the currently active service window and the 'Accumulated Selections' text window has been opened. The option will cause the selection to be appended to the text window.

Save Text

This option will only be active if the 'Accumulated Selections' text window has been opened and contains data. The option will cause the entire contents of the window to be saved to a HyperCard file via the standard file selection dialog (N.B. any text styling will be lost).

Macros Menu

A popup menu of single line character strings (Macros) may be associated with each active service window. The menu (if it has any items) will appear immediately below the window bar of the active service window if the mouse is clicked in the window bar whilst the ‘option’ key is depressed. (N.B. if the menu does not have any items, i.e. no macros have been defined for the window then attempting to display the menu will result in a single 'beep'). A maximum of 11 items may be added to the popup menu by selecting the ‘Edit Macros’ item from the edit menu. Each macro item is limited to 255 characters in length and may be modified or deleted by selecting the required item whilst the command key is depressed. The macros may be mapped to one of two sets of keyboard keys, the extended keyboard 'Funtion keys' or the standard keyboard 'Numeric key pad' keys. Selecting an item from the menu results in the contents of the associated macro being sent to the service.

Tool Menus

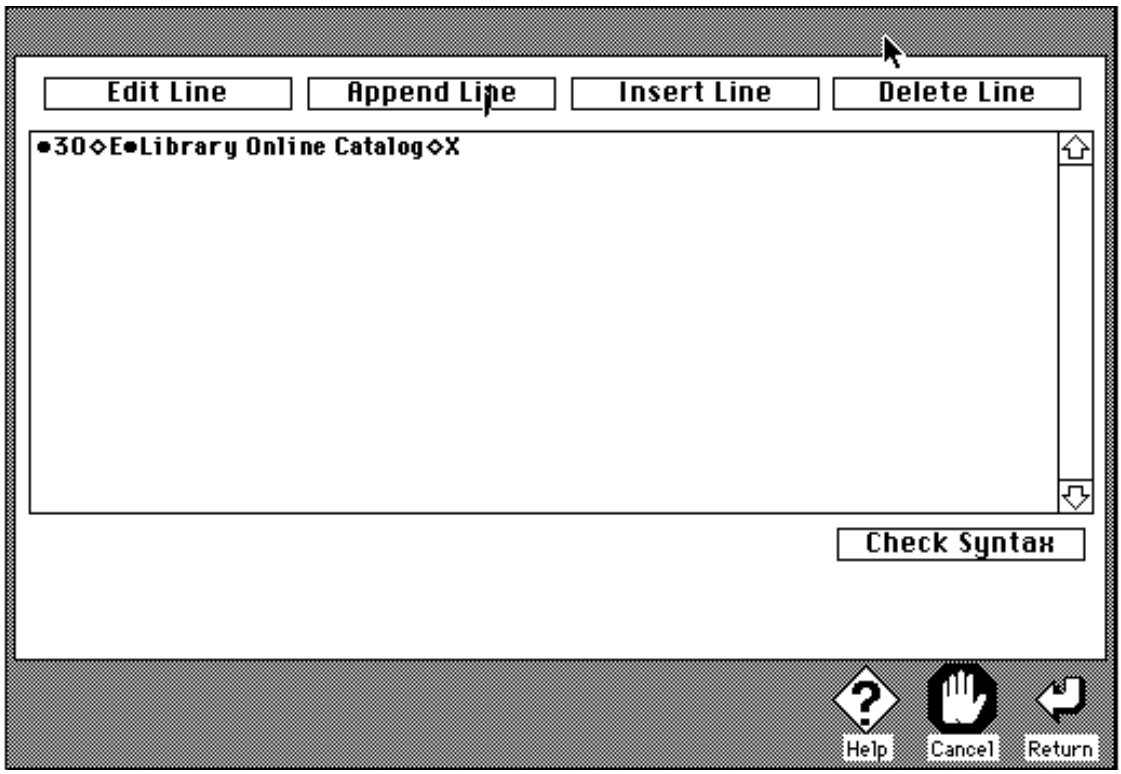
The menu bar for a service **may** contain up to 2 additional menus. These menus are associated with the software tools used to connect to a particular service. Whether or not the additional menus will be present will depend on the tools used and the parameter settings (parameters 4 &6) for a particular service.

Overview

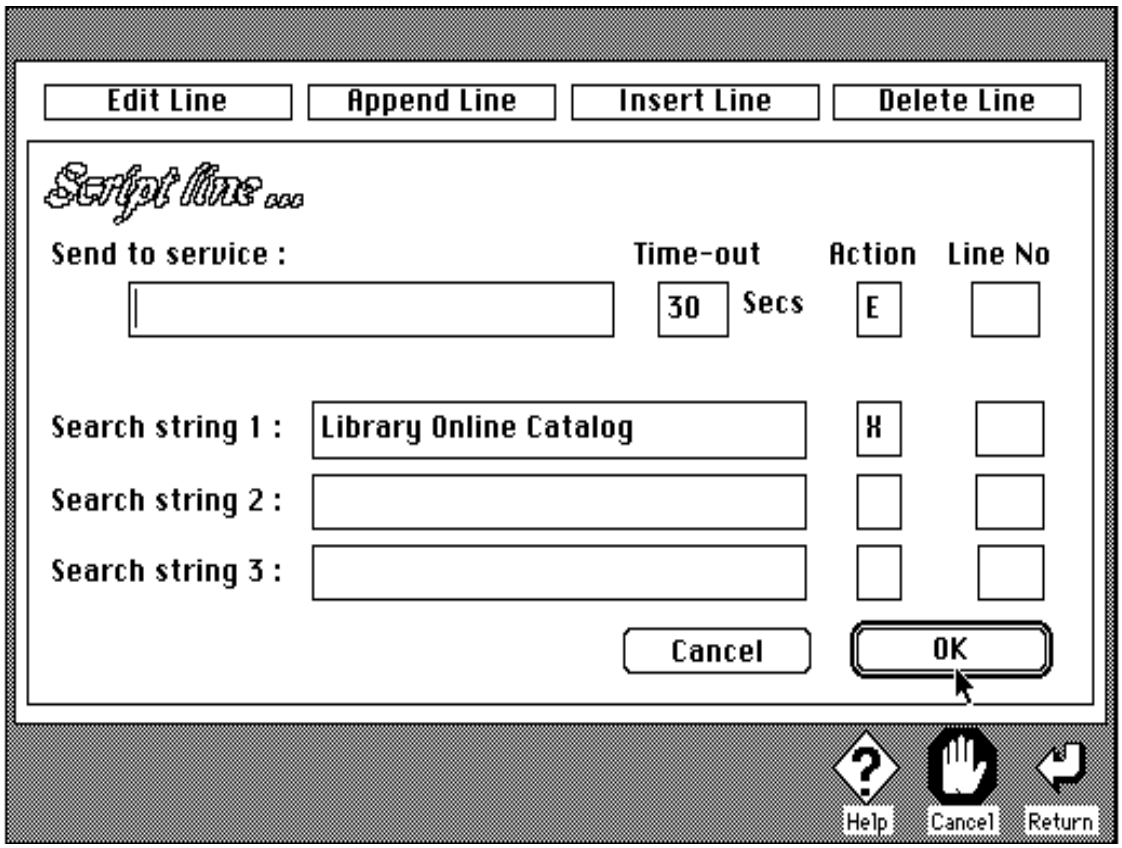
A rudimentary scripting language is supported to provide the ability to login to and logout from a service. Although somewhat sparse the language is powerful enough to cover most requirements.

A Script consists of one or more command lines. Each command line may contain up to five elements, each of which is delimited by a 'bullet' (•) character. Certain command elements consist of sub-elements, each of which is delimited by a 'diamond' (◊) character.

Scripts are created/modified via the 'Script' card in the **xLibrary Installer** stack. The following example of a 'Script' card contains a two line script the second of which is denoted in **bold** type to indicated that it is the currently active line..



The currently active script line is edited via a specialised dialog invoked by clicking the 'Edit Line' button or double-clicking the script line itself. The resulting dialog box is constructed from the various elements and sub-elements of the script line.



Each script element is described below :

Command Element

If present, the command element is interpreted as follows -

If the command element contains the word 'PASSWORD' then the word is replaced by the contents of the script variable PASSWORD (see sub-section 'Action to be taken' below) and the resulting character string is sent to the service.

If the first character of the command element = "∞" (option 5) then a long break signal is sent to the service otherwise the command element is edited as per the following rules and the resultant character string is sent to the service.

If the command is proceeded by a "@" (shift 2) character then the remaining command element is to be sent 'slowly' to the service, i.e. 6 characters per second.

The command element may contain character pairs that together define a control character. That is, the character "©" (option g) is interpreted as defining the following character to be a control character. The following character is converted to its numeric control value by subtracting 64 from its character value, e.g. ©A = control A and ©[= escape.

All occurrences of the character "®" (option r) are replaced by carriage- returns.

If the command element contains a "Ω" (option z) character then this is interpreted as a request to block out the remaining characters of the element with 'bullet' (•) characters. This facility is intended for those cases where script execution display has been specified and sensitive data

(e.g. passwords) is contained within the command.

Time-out Element

This element may contain up to 3 sub-elements, they are as follows :

a) Time Period.

The amount of time, in seconds to allow for a successful response to the transmitted command element (N.B. A successful response is defined as input being received that matches one of the character strings defined in 'Search Elements' below). If the time limit is exceeded a variety of actions may be taken depending on the setting of the following sub-element.

b) Action to be Taken.

This sub-element defines the action to be taken if no valid response is returned from the service in the specified time period. The various action codes are as follows :

C Continue execution at the relative script line defined by the next sub-element.
X Terminate the script successfully.
A Abort the script and display an error message.
E Ask the operator if a further time period is required (to allow for slow processors).
R Decrement the retry counter associated with this line. If the resulting value is zero then abort the script and display an error message, otherwise continue from the relative script line defined by the next sub-element. The number of times a line of script will be tried before an error is posted may be defined differently for each service (via option 14 of the service parameters).
P Request input from the operator via a 'Password' dialog box. The operator response is masked for security and the resulting input is loaded (with carriage-return appended) into the script variable called PASSWORD. This variable may then be referred to in the command element (see above). The script is continued from the script line defined by the next sub-element.

c) Relative Line Number.

This is a positive or negative value that is added to the current line number to obtain the new script line number from which to continue execution. For example, to continue execution from the script line immediately prior to current one use value -1, to continue execution from the line following the current one use value 1.

Search Elements

The script line may contain up to 3 further elements. Each of the elements defines a character string to search for in the input received from the service. Each element consists of up to 3 sub-elements as defined below :

a) Search String.

The character string to search for in the service output. The string may contain character pairs that together define a control character. That is, the character "©" (option g) is interpreted as defining the following character to be a control character. The following character is converted to its numeric control value by subtracting 64 from its character value, e.g. ©A = control A and

©[= escape.The search is case sensitive.

b) Action to be Taken.

This sub-element defines the action to be taken if search string is detected in the service output :

C Continue execution at the relative script line defined by the next sub-element.
X Terminate the script successfully.
A Abort the script and display an error message.
P Request input from the operator via a 'Password' dialog box. The operator response is masked for security and the resulting input is loaded (with carriage-return appended) into the script variable called PASSWORD. This variable may then be referred to in the command element (see above). The script is continued from the script line defined by the next sub-element.

c) Relative Line Number.

This is a positive or negative value that is added to the current line number to obtain the new script line number from which to continue execution. For example, to continue execution from the script line immediately prior to current one use value -1, to continue execution from the line following the current one use value 1.

Scripting Examples.

®•10◊E•login◊C◊1•$◊X

This script line contains 4 elements and is interpreted as follows:
A carriage-return is sent to the connection. The output from the connection is scanned for one of the 2 character strings:"login" or "$" . The search is continued until one of the strings is detected or 10 seconds has elapsed. In the latter case (considered to be the 'failure' case) the action to be taken is defined by the code "E". The E code results in a dialog box being shown that asks the operator whether a further time period (10 seconds) is to be spent searching for valid output. If one of the search strings is detected then the action taken depends on the code associated with the string. The two cases are as follows

"login"	C◊1	script execution will continue from the next script line.
"$"	X	the script is terminated successfully.

@tim®•10◊A•Password◊P◊1

This script line contains 3 elements and is interpreted as follows. A character string "tim" (with carriage-return attached) is sent to the connection at a rate of 6 characters per second. The resulting output from the connection is then searched for the single character string "Password". The search is continued until either the string is detected or until 10 seconds has elapsed. In the latter case (considered to be the 'failure' case) the action to be taken is defined by the code "A". This code results in the script being terminated with an error message. If the search string is detected then the action taken is defined by the code "P". This code results in a dialog box being shown that allows the operator to input a masked password. Once the operator has done this and clicked the "OK" button, the value entered will be loaded (with carriage-return attached) into the script variable PASSWORD and script execution will continue from the next

script line.

PASSWORD•10◊R◊-2•$◊X•Login denied©G◊A

This script line contains 4 elements and is interpreted as follows. The contents of the script variable PASSWORD are sent to the connection. The resulting output from the connection is then searched for one of the 2 character strings "$" or "Login denied©G" (**N.B. the last character of the second string is <control> G - the bell character).** The search is continued until either one of the strings is detected or until 10 seconds has elapsed. In the latter case (considered to be the 'failure' case) the action to be taken is defined by the code "R". This code results in the retry counter associated with this line being decremented by 1. If the resulting value is positive then script execution is continued from the first line of scripting (current line minus 2). If the resulting value is zero then script execution is terminated with an error message. If one of the search strings is detected then the action taken depends on the code associated with the string. The two cases are as follows -

"$" - A script execution is terminated successfully.

"Login denied©G" - X script execution will be terminated with an error message.