

TelQA: How to Test Terminal-based Applications



Table of Contents

1.0 Introduction.....	1
2.0 Recording the Script.....	2
3.0 Modelling the Script.....	8

1.0 Introduction

TelQA Test is designed primarily for testing web-based applications. However, many companies still use "legacy" applications designed for use with character-cell terminals such as VT100, VT220, VT320 etc. In many cases web-based front-ends have been developed to provide a modern interface to such applications, but there are often still users of the original terminal-based interface, usually connecting via terminal emulator software rather than a physical terminal. There may still be a need to test such applications and TelQA Test provides comprehensive facilities to do so.

TelQA Test enables a script to be produced by recording the interactions of a user with a terminal-based application, and as with web-based applications the interactions may be stored either as a TelQA SmartScript or as a C# or VB.NET based script. Once a script has been recorded, it can then be edited to allow large numbers of Virtual Users (VUs) to be replayed with unique identities. The same range of SmartScript features available for web-based scripts can also be used with terminal-based scripts. In fact, once the script has been generated, the full functionality of TelQA Test is available independent of the application type. It is possible to produce a test which includes both web-based and terminal-based scripts, and it is even possible to combine web and terminal interactions within a single script.

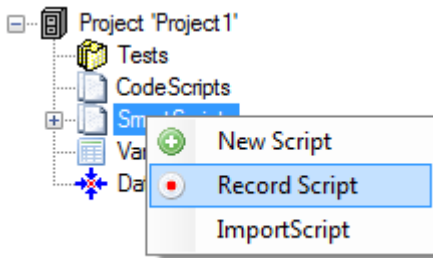
TelQA Test currently supports terminal-based applications which can be accessed using the Telnet protocol, and which support the VT range of terminals including VT100, VT220 and VT320 or other terminals which use ANSI escape sequences. If you have a requirement outside this scope then please contact TelQA Limited to discuss your requirement.

The following sections describe how to use TelQA Test to test a terminal-based application. The examples described use a simple credit card application running under OpenVMS.

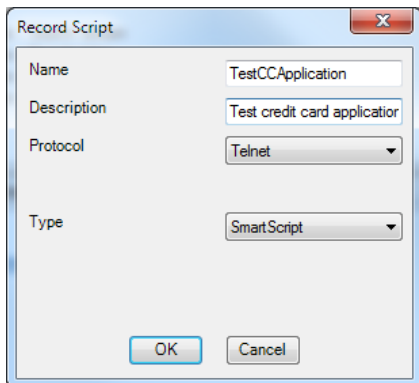
2.0 Recording the Script

By default, TelQA Test does not display the option to record a terminal-based script, as this is not required for web testing. In order to enable the feature, select the Tools>Options menu item and change the item "Support Terminal Scripts" to have the value "Yes", then click OK.

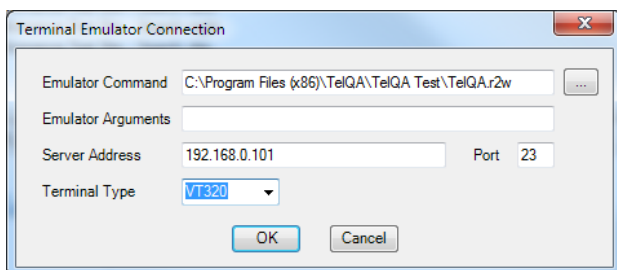
If not already open, open the Project window using View>Project. Right click the SmartScripts node of the project tree and select the Record Script option:



Enter a suitable name and description and change the Protocol value to "Telnet":



Click OK, then the Terminal Emulator Connection window will be displayed:



In order to record the terminal session, TelQA Test requires the use of a terminal emulator which the user will use to interact with the target application. Any terminal emulator which supports the VT terminal types may be used, including the HyperTerminal application which is included in Windows.

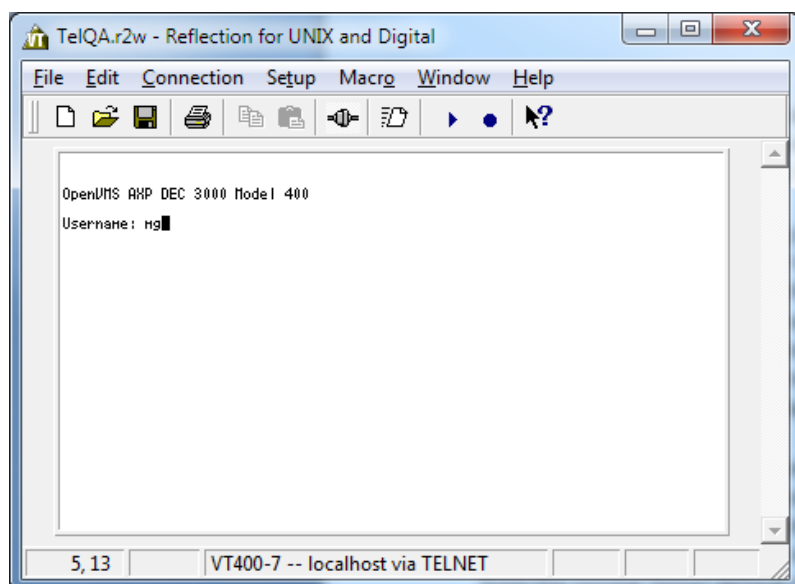
When you initiate the recording of a terminal script, TelQA Test interposes a proxy Telnet server between the terminal emulator and the application. This proxy server listens on port 8023 by default but this may be changed by altering the value of "Telnet Capture Port" in Tools>Options. The terminal emulator must be set up to connect to this port rather than directly to the target application. Most emulators allow you to save session settings to a file then invoke this file as a shortcut to connect directly without needing to set up the

connection values each time. Shortcut files for HyperTerminal and Reflection are included in the TelQA Test installation directory.

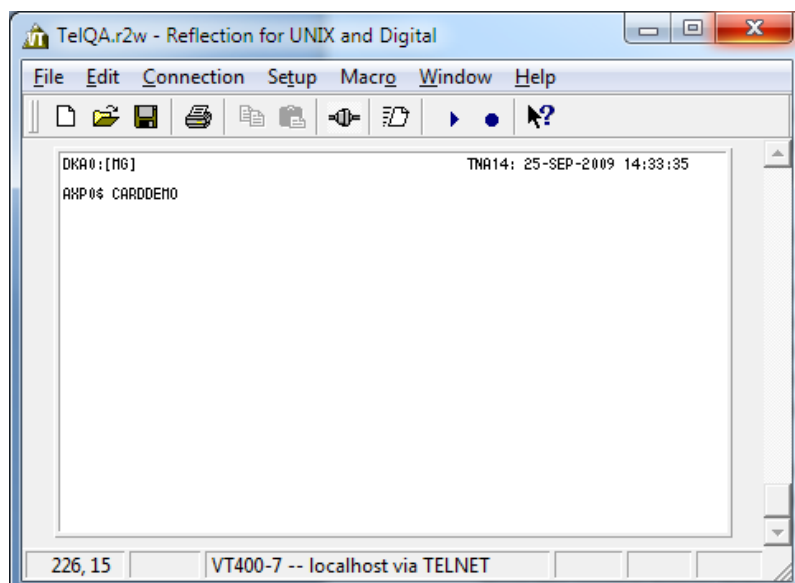
In the Terminal Emulator Connection window, set the Emulator Command by browsing to the appropriate shortcut file or directly to the emulator executable. If your emulator requires any command line arguments then these may be entered in the next field. Enter the name or IP address of the target application server in the Server Address field, and change the Port value if this differs from the normal Telnet default of 23.

Click OK to commence recording. This will start the terminal emulator, which should connect to the target application server. For the purpose of this document we will run the credit card application, create a new customer account and issue a credit card to the customer.

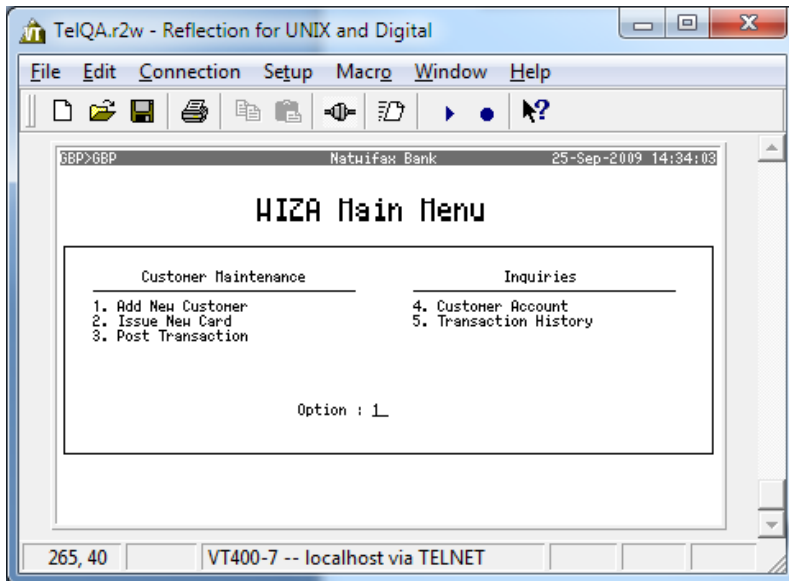
After connecting to the application server, the OpenVMS login prompt is displayed and we login:



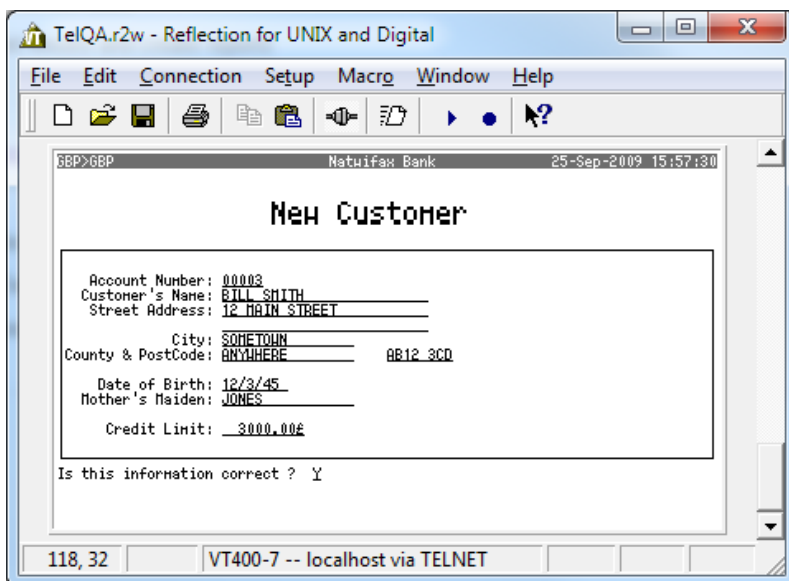
Run the credit card application:



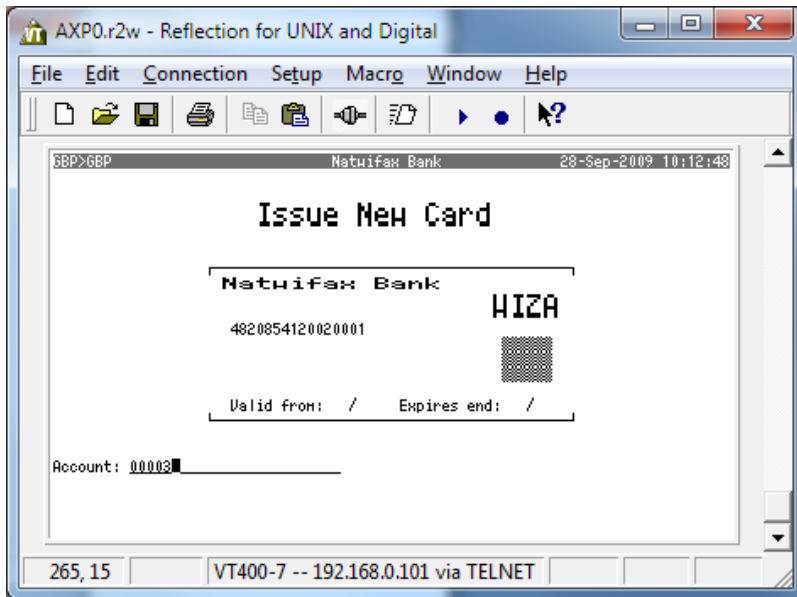
The main application screen is displayed, showing the available options. Select option 1 to add a new customer:



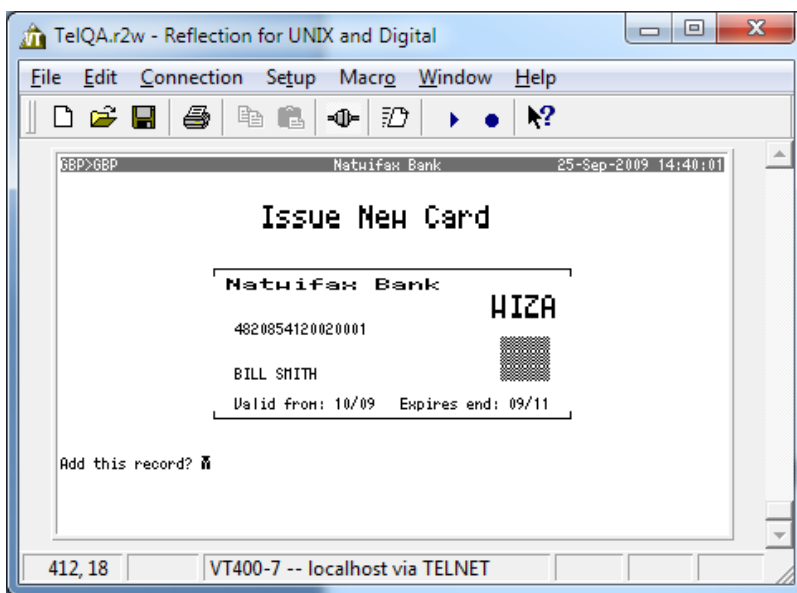
Enter the new customer's details and the application prompts to confirm that the details are correct:




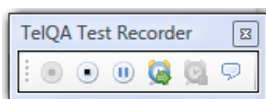
Return to the main screen, select option 2 to issue a credit card, then enter the account number which we have just created:



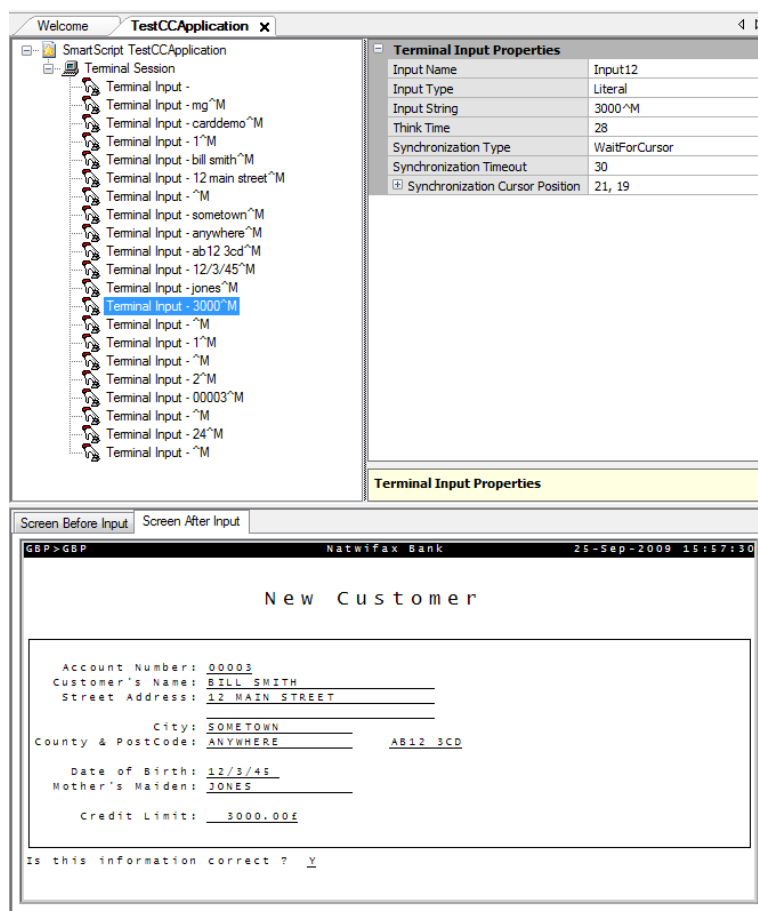
Enter a validity period of 24 months and a credit limit of £3000. The application prompts to confirm the operation:



End the recording session by pressing the  button in the TelQA Test Recorder control dialog:



The TestCCApplication script will now be displayed:



Note that for each terminal input, an image of the screen as it was immediately before and after the input is displayed in the lower pane. This will assist in modelling the script as discussed in the next section.

When replaying a terminal-based script, some form of synchronization is normally required to ensure that the application is in the correct state to receive each input. For a screen-based application like the credit card application, synchronization using the cursor position usually works well. This approach requires that the cursor is positioned at a specific point on the screen before the input is sent to the application. The highlighted input above shows the text "3000" followed by the Return key (represented by ^M or Control-M) being entered in the Credit Limit field, with the cursor at column 21 of line 19. Note that the screen image below shows the state of the screen *after* the input.

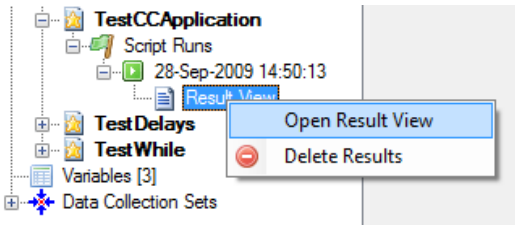
The script is created using the "WaitForCursor" synchronization type by default. However, some applications may be less amenable to synchronization using the cursor position, particularly if the input occurs within an area of the screen which may scroll. In such cases the synchronization type "WaitForScreenText" may be more suitable. This requires a specified text string to be present within a specified area of the screen before an input is sent. The area may be specified as a rectangle of any size up to the full screen size.

Each input has a synchronization timeout property which defaults to 30 seconds. If the specified synchronization criterion is not met within this time, the VU will be aborted. The abort message in the event log will show the input at which loss of synchronization occurred, and the actual and expected cursor positions in the case of cursor synchronization, or the text string which was not found in the case of text synchronization.

After the new script has been generated, you can replay it using the toolbar Script Run button:



After the Script run has completed, open the Result View from the Script Run node in the Project Window.



Check that the Event Log does not contain errors. Errors are displayed in red. The contents of the log should be similar to the example below (with different dates).

Time Stamp	Source	Event
28-Sep-2009		
14:50:13.14	Project1.\$TestCCApplication	Test Started
14:50:13.15	Project1.\$TestCCApplication	Sampling Started
14:50:13.15	Project1.\$TestCCApplication.\$Sequence	Sequence Started
14:50:13.15	Project1.\$TestCCApplication.\$Sequence.1	Sequence Iteration 1 Started
14:50:13.16	Project1.\$TestCCApplication.\$Sequence.1.TestCCApplication	Task Started
14:50:13.52	Project1.\$TestCCApplication.\$Sequence.1.TestCCApplication	Task Iteration 1 Started
14:50:25.30	Project1.\$TestCCApplication.\$Sequence.1.TestCCApplication	Task Iteration 1 Completed
14:50:25.30	Project1.\$TestCCApplication.\$Sequence.1.TestCCApplication	Task Completed
14:50:25.30	Project1.\$TestCCApplication.\$Sequence.1	Sequence Iteration 1 Completed
14:50:25.30	Project1.\$TestCCApplication.\$Sequence	Sequence Completed
14:50:25.30	Project1.\$TestCCApplication	Sampling Completed
14:50:25.30	Project1.\$TestCCApplication	Test Completed

Display the Script Run Viewer by clicking the tab. Note that the full script used for the playback is listed, together with a trace of the script steps actually executed.

```
Start SmartScript TestCCApplication
Terminal Session "192.168.0.101:23" connected
Terminal Input Input1: "mg^M"
Terminal Input Input2: "carddemo^M"
Terminal Input Input3: "1^M"
Terminal Input Input4: "bill smith^M"
Terminal Input Input5: "12 main street^M"
Terminal Input Input6: ""^M"
Terminal Input Input7: "sometown^M"
Terminal Input Input8: "anywhere^M"
Terminal Input Input9: "ab12 3cd^M"
Terminal Input Input10: "12/3/45^M"
Terminal Input Input11: "jones^M"
Terminal Input Input12: "3000^M"
Terminal Input Input13: ""^M"
Terminal Input Input14: "1^M"
Terminal Input Input15: ""^M"
Terminal Input Input16: "2^M"
Terminal Input Input17: "00003^M"
Terminal Input Input18: ""^M"
Terminal Input Input19: "24^M"
Terminal Input Input20: ""^M"
Terminal Session "192.168.0.101:23" disconnected
End SmartScript TestCCApplication
```

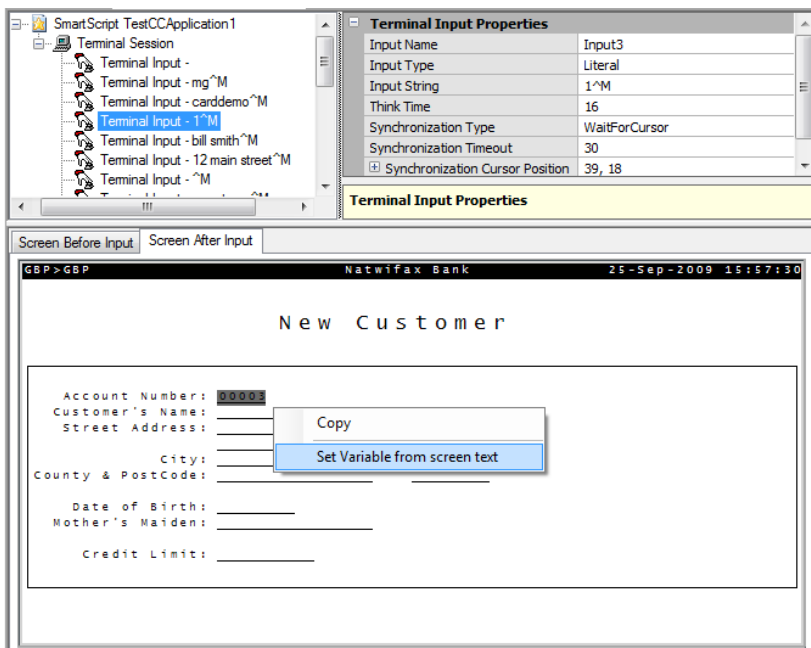
Close the Script Run tab and Results View tab.

3.0 Modelling the Script

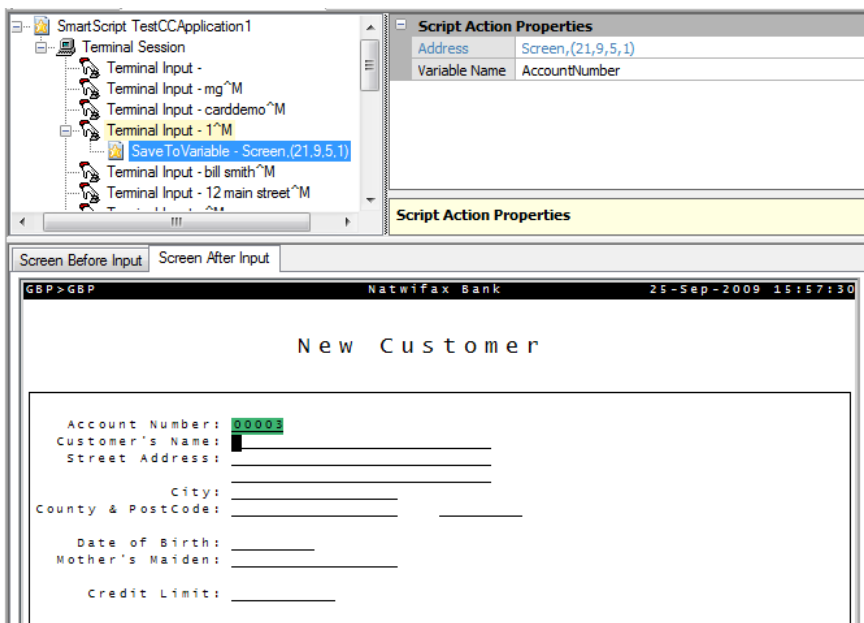
After a new script has been recorded it will generally be necessary to "model" it in order to make it usable within a multi-VU test. This involves replacing fixed values entered during the recording with values generated dynamically during the test.

A simple example of this requirement is the use of the account number in the credit card application. A new account number is generated by the application when the Add New Customer option is selected. This number is subsequently entered manually when the Issue New Card option is selected, in order to specify the customer to whom the card is to be issued. If you run the script as recorded you will note that the same value, 00003, is entered each time as the account number for the credit card issue, rather than the value generated by the Add New Customer function.

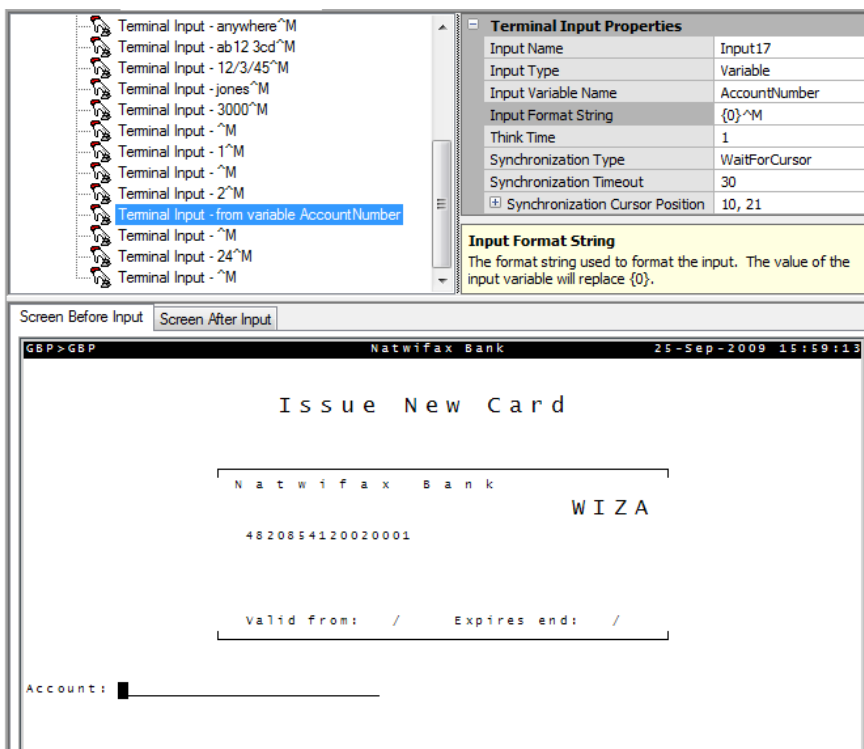
In order to fix this problem, we will extract the account number issued by the Add New Customer function, save it in a variable, and enter this variable value as the account number for the Issue New Card option. First, select the input corresponding to the selection of the Add New Customer option and select the Screen After Input which shows the account number allocated by the application. Select this by holding down the mouse at the start of the field and dragging until the whole field is selected as indicated by a grey background. Now right-click and select the option to Set Variable from screen text:



A Save To Variable Node will be added below the Terminal Input and the selected text will be shown coloured to indicate that it is being used as the source of a variable. Change the variable name in the property grid to a meaningful value, e.g. AccountNumber.



We now need to use the AccountNumber variable as the source for the account number entered when issuing a new credit card. Select the input corresponding to the entry of the account number, change the Input Type to Variable and select the variable name "AccountNumber". Leave the Input Format String as "{0}^M"; this will ensure that a carriage return is entered following the account number.



If you now run the script again and view the Script Trace, you'll note that the original input of 00003 has been replaced by the account number displayed in response to Add New Customer.

The following are some examples of other modelling techniques that could be used within terminal-based scripts. Note, however, that most of these are equally applicable to web-based or other types of script:

1. Inputs such as name, address etc. may be obtained from List, Table or Database variables. The TelQA Getting Started Guide includes examples of the use of List and Table variables for entering dynamic data.
2. The selection of input data may be randomized using a RandomInteger variable in conjunction with a Switch script step. Again, this technique is illustrated in the Getting Started Guide.
3. Conditional script processing may be achieved by using If/Then/Else script steps, checking the text within an area of the screen to determine the action to be taken.