



Customer Story

Tesco Shelf Edge Pricing with Flynet Screen Tools

Mark Bain
Flynet Limited

Introduction

Elements of Tesco's shelf edge pricing solution run on an in-store UNIX server. Amongst other things, this shelf edge pricing solution offers facilities for printing reduction labels, checking and printing shelf edge and promotional labels.

This Pricing solution was accessed from DOS based hand held computers over a bespoke radio frequency network. The application on the UNIX server was tailored to control these DOS devices that included a barcode scanner and printer.

As a part of Tesco's new Mobile Shelf Edge System (MSES), these DOS devices are being replaced with Microsoft Pocket Windows devices on standard Radio Networks (IEEE 802.11). Many parts of the MSES solution rely on web browsing with Internet Explorer. However the successful Pricing portion of the solution continues to be based on the UNIX server, accessible with DEC-VT emulation.

Tesco required a VT Terminal Emulator package that could run on new Microsoft Pocket Windows devices. This new emulation had to handle many tailored components in the UNIX Pricing application. These components were designed specifically to run the DOS devices and were not very well suited for Pocket Windows.

To successfully bring the Pricing application to the devices, these main criteria had to be met:

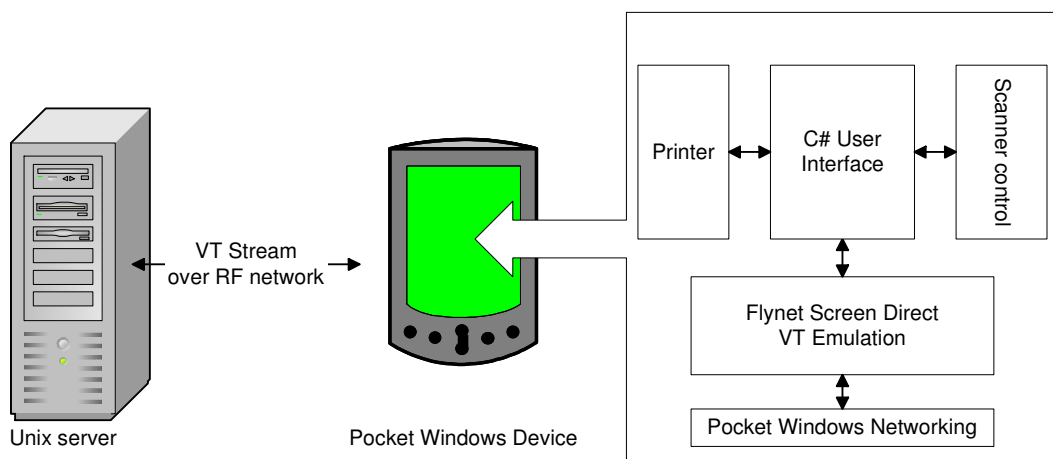
- DEC VT
- Printing
- Barcode scanner support
- Radio Frequency network aware
- Simple Installation
- Minimal use of Pocket Windows stylus

The final solution implemented for Tesco by Flynet met all these points. The purpose of this paper is to show how building the solution on Flynet Screen Connector enabled Tesco to quickly and easily attain its goals.

Overview of the solution

The core VT emulation is provided by "Flynet Screen Direct". FSD or "Direct" is a member of the Flynet Screen Tools family that runs native on the desktop or pocket device. It offers up a programmatic interface to screen based protocols such as VT, TN3270 and TN5250. This programmatic interface is designed to ease the creation of both traditional terminal emulation packages as well as more bespoke screen-scraping applications. It is very closely related to its larger brother, Flynet Screen Connector, which is designed to run server-side and is able to cope with large volumes of users.

To display the VT screens, a front-end was written in Microsoft C# for the .NET Compact Framework. This custom-written interface used some of the methods in FSD that make it easy to build traditional screen based emulators, but took it a stage further. It allowed the developers to easily customise the emulation to catch the escape codes designed to control the DOS printer and barcode scanner and convert them to use the Pocket Windows drivers and ports.



Supporting VT for the Pricing application

The VT used by the Pricing application is mainly based around VT220. The initial sign on is to a standard UNIX login screen. The sign on screen is the traditional 80 columns by 25 lines, in telnet mode. The username and password are prompted for at the bottom of the screen.

After a successful UNIX sign on, the Pricing application starts automatically. The application was designed to run on DOS based handhelds that had a screen size of 16 columns by 8 lines. The application in effect only uses the top left hand side of the available 80x25 screen.

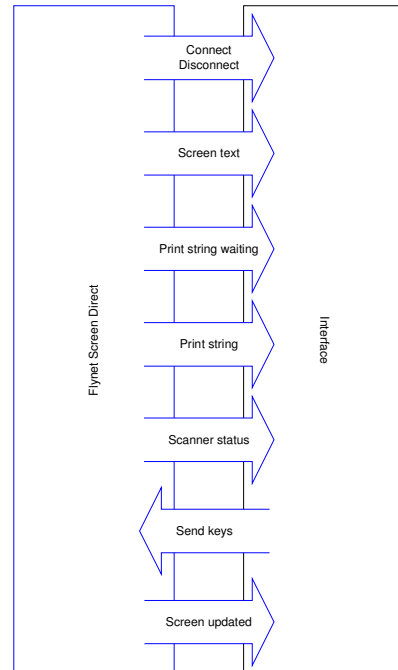
This is mostly standard VT220, with a few extras. The VT feature for sending print strings to attached printers is used, as well as bespoke escape codes that were designed to instruct the DOS based system to turn on and off the barcode scanner.

Flynet Screen Direct handles print strings by offering a method that returns whether a print job has been sent or not. The string can then be read from another property.

The modularity of FSD made it easy to add a new method which returned whether the scanner should be on or off. The method returns the last instruction sent from the application for the mode of the barcode scanner.

To keep the configuration of the final emulation to a minimum, a simple way of devising the UNIX server's address was formulated. The IP schema for each store was the same, and the UNIX server was always x.x.0.1, with x.x being the domain for that particular store.

The connection routine therefore only needs to work out the IP address of the Pocket Windows device it is running on, and substitute 0.1 for the last two octets of the address. This eliminated the need for any configuration. As long as the device was correctly set up to run on the store's network, it should be able to function. Re-configuration after maintenance or repair work on any particular device is also not necessary.



Barcode scanner support

The barcode scanner has to be instructed whether it should scan or not. This was designed to stop accidental scanning of a product when a quantity or menu navigation was expected. The UNIX application sends non-standard VT escape codes to indicate whether the scanner should be on or off.

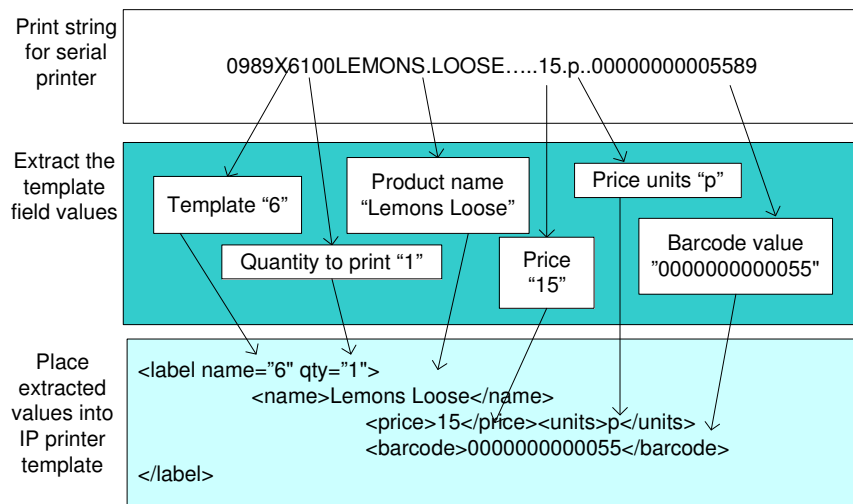
The Pocket Windows device manufacturer provided .NET classes that allowed control of the barcode scanner. Normally, when scanning with this device, the scanned value would be sent through the keyboard buffer. It was impossible to tell if the user had typed or scanned a value. The barcode scanner .NET class was used to catch the scan as an event and stop it being sent through the keyboard buffer. Dependent on whether the scanner should be on or off, as dictated by the UNIX application, the value could be sent or not to the VT emulation.

Printing

The UNIX application uses standard VT escape codes to indicate that a certain string is destined for a printer attached to the VT emulation, and not the screen. FSD indicates that a print string is waiting, and that string can be retrieved.

The Pricing application pre-formats these print strings for a specific printer. All the DOS device had to do was send the string to a serial port. FSD made this print string available as a property and this value could be sent to the serial port resulting in a successful print.

However, these serial printers are old and expensive to maintain. Tesco wants to phase in new IP based printers. The problem was that the UNIX application could only send one type of pre-formatted print string, and both printers had to be supported.



The solution was to detect which printer was being used and dissect the serial printer's print string and reformat it into a string suitable for the new IP printer.

Both printers can use template based print instructions, i.e. give the printer a template name, and the values for that template. This made it easy just to extract template values from the serial print string and put them into the new IP printer template format.

Radio frequency network aware

The Pocket Windows devices run on a Radio Frequency (IEEE 802.11) network. With the best will in the world, black spots or interference can cause network connectivity to drop. VT emulation over telnet requires a constant connection to the host. If any keystrokes are sent during a moment of network drop-out the Pocket Windows IP socket layer will lose the connection. This would mean having to re-connect, sign on and return to the place in the application where you were before.

This issue could easily be circumnavigated by checking the status of the network before sending a keystroke. If the network is okay, send the keys. If the network is experiencing a drop out, display a message asking the user to move back into range then send the keystrokes. This would avoid the IP layer closing the socket.

A quick call to a networking API is used to ascertain the status of the network.

The Interface

The interface or screen display had to be easily read, and minimise the need to use the Pocket Windows stylus.

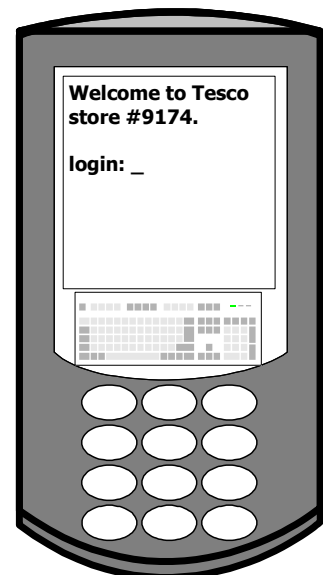
A bold text display area was placed to cover the majority of the screen. This would allow the display font to be as large as possible. During the sign on stage, the bottom two lines from the 80x25 screen are displayed.

The stylus is needed during the UNIX sign on stage to enter the username and password. This is tapped out on the on-screen keyboard.

After successfully logging into the UNIX box, the Pricing application itself starts. This uses the first 8 lines of the display, to a maximum of 16 characters across.

Menu navigation is by number key press and can be done via the hardware numeric keypad on the Pocket Windows device – so no need to use the stylus.

The only other time non numerical input is required is when answering a “Yes” or “No” question. To avoid having to use the stylus to hit the on-screen keyboard, the Interface scans the display text for a question mark or “Y/N”. If the text is present, the Interface displays large “Yes” and “No” buttons. These can easily be pressed with a finger.





The Interface uses the “OnKeyPress()” event to send the keystrokes to the Flynet Screen Direct VT emulation. The key stroke is not displayed at this time. As with all VT applications, it is not known where (if at all) the character will appear.

A separate ScreenSync thread monitors the VT screen display. If the VT screen is updated, the ScreenSync thread fires an update event back to the Interface. The Interface then knows that it needs to redraw the screen.

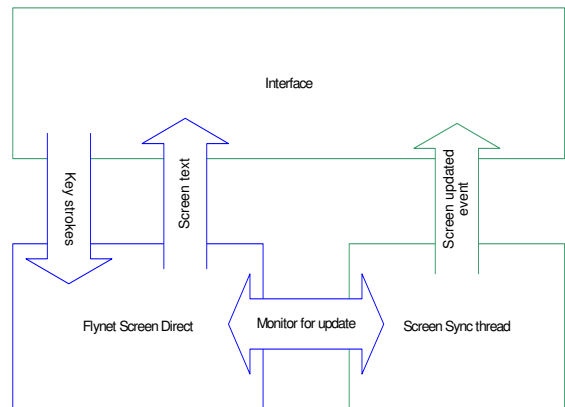
A timer in the Interface checks network connectivity every pre-defined number of milliseconds, displaying an “out of range message” if the network drops out. During this drop out, the ScreenSync thread is paused, and the Interface will accept no more key presses.

As an additional safety check, the network status is ascertained on every key press. If the network has dropped, it is buffered until the network returns. Thankfully there is minimal overhead for this check.

As the print string is sent via the normal screen, just with special VT escape codes, the screen update event would be fired if a print job was sent. The screen-redraw routine checks to see if there is a print string waiting, and sends it to the printer.

The Pricing application was only designed to work with one type of printer, the serial printer. The Pricing screen straight after sign on asks the user to scan a barcode on the edge of their printer. This allows the Pricing application to format the print string correctly for the serial printer. An interception of this scan allows us to check whether or not the printer is in fact the old serial printer or new IP based printer.

When the printer is scanned, the Interface checks to see if it's an IP address or straightforward string of characters. If it is an IP address, it knows that the new IP printer is being used, and it needs to reformat the print string and send to the IP driver. If not, it can send the print string unchanged to the serial port.



User acceptance

Tesco's user acceptance group evaluated Flynet's solution against the others on offer. The Leader of the Acceptance Group had explained that the most important criteria were that the interface had to be easy to read on the small screens, it needed to keep use of the Pocket Windows stylus to an absolute minimum (for speed and ease of use on the shop floor) and the final solution had to be able to cope seamlessly with drop-outs in the RF network connection. By following the approach described above Flynet was able to demonstrate superiority in these areas – which is why the Acceptance Group favoured this particular solution over the standard terminal emulation offerings.

Working alternatives

Another equally valid and working methodology was built with Tesco; however speed of rollout favoured the FSD solution above.

Flynet Screen Connector, the server based version of FSD used on the pocket devices, was run on a central server. This model moved VT emulation, printing and event handling off the device to a central point.

The Pocket Windows devices ran a thin interface that connected to the central server. Two interface flavours were built, one in C# and the other for web browsers. The XML web service style connection to the central server was not so sensitive to network drop out like VT over telnet.

The solution

Tesco ended up with a tailored VT terminal emulation solution. The Interface brought together the various components that were required. The working environment and device-specific drivers would have made implementing a solution with a traditional VT thick terminal emulation next to impossible.

Flynet Screen Direct offered an interface that removed all the complexity of VT emulation and replaced it with a simple model that was rapidly customised to fit the situation exactly.