Sun™ 3270 Pathway User's Guide

Release 2.0

Sun Microsystems, Inc.
www.sun.com

Part No. 816-5340-11
November 2003, Revision A

Submit comments about this document at: http://www.sun.com/hwdocs/feedback

Contents

Figures

Tables

Preface

This document describes how to use the Sun™ 3270 Pathway software suite. To fully use the Sun 3270 Pathway Bean, you must have a thorough knowledge of Java™ programming techniques. Non-programmers can successfully use the Pathway Terminal, recorder, and router by following the instructions in this document.

How This Book Is Organized

Chapter 1 provides an introduction to the Sun 3270 Pathway software.

Chapter 2 describes the software required to run the product.

Chapter 3 describes how to install the Sun 3270 Pathway software.

Chapter 4 describes how to use the Sun 3270 Pathway Bean with the Java Development Kit (JDK™) and in an integrated development environment (IDE).

Chapter 5 provides a brief description of each of the sample programs.

Chapter 6 describes the Sun 3270 Pathway Bean programmable interface.

Chapter 7 introduces the Sun 3270 Pathway Recorder.

Chapter 8 describes the Sun 3270 Pathway Recorder in detail.

Chapter 9 describes the Store List Wizard.

Chapter 10 describes the TCP routing software.

Chapter 11 describes the Pathway Terminal in detail.

Using UNIX Commands

This document does not contain information on basic UNIX® commands and procedures such as shutting down the system, booting the system, and configuring devices. See the following for this information:

- Software documentation that you received with your system
- Solaris™ operating environment documentation, which is at

  http://docs.sun.com

Typographic Conventions

Typeface[1]	Meaning	Examples
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your .login file. Use ls -a to list all files. % You have mail.
AaBbCc123	What you type, when contrasted with on-screen computer output	% **su** Password:
AaBbCc123	Book titles, new words or terms, words to be emphasized. Replace command-line variables with real names or values.	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this. To delete a file, type rm *filename*.

1 The settings on your browser might differ from these settings.

Shell Prompts

Shell	Prompt
C shell	*machine-name*%
C shell superuser	*machine-name*#
Bourne shell and Korn shell	$
Bourne shell and Korn shell superuser	#

Accessing Sun Documentation

You can view, print, or purchase a broad selection of Sun documentation, including localized versions, at:

`http://www.sun.com/documentation`

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in this document, go to:

`http://www.sun.com/service/contacting`

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. You can submit your comments by going to:

`http://www.sun.com/hwdocs/feedback`

Please include the title and part number of your document with your feedback:

Sun 3270 Pathway User's Guide, part number 816-5340-11

CHAPTER 1

Introduction to Sun 3270 Pathway

This chapter contains the following topics:

Sun 3270 Pathway Overview

Sun 3270 Pathway is the collective name for a suite of programs that can communicate to a 3270 end-system. The core of the components is the 3270 Pathway Bean, a Java component that enables a developer to:

- Develop an application that connects to a 3270 end-system
- Run an application on that end-system
- Extract data from that application.

Because the Sun 3270 Pathway Bean conforms to the Java Bean standards, you can use it to integrate it into many of the available Java integrated development environments (IDEs) that support Java Beans, such as Sun ONE Studio. You can also use a Java Development Kit (JDK) to produce applications that use Sun 3270 Pathway.

The following components are shipped with Sun 3270 Pathway:

- The Sun 3270 Pathway Bean (see Chapter 4)
- The 3270 Pathway Navigation Recorder tool (see Chapter 7)
- A TCP Router (see Chapter 10)
- The Pathway Terminal (see Chapter 11)

Supported Transport Protocols

Sun 3270 Pathway supports the Telnet 3270 (TN3270) transport protocol. This means that your 3270 host must be available through a TN3270 server so that the Sun 3270 Pathway Bean can connect to it.

The Sun 3270 Pathway supports both types of TN3270:

- Basic TN3270
- TN3270E (conforms to RFC 1647 and RFC 2355)

Sun 3270 Pathway is designed as a TN3270 client and primarily deals with 3270 datastreams. However, it can also deal with non-3270 datastreams during the following circumstances:

- When using the TN3270E protocol, the server is permitted to send raw Telnet data; therefore, the terminal must act in a non-3270 manner.
- When a server has a non-3270 routing front-end program, which sends and receives non-3270 datastreams.

When Sun 3270 Pathway is dealing with these non-3270 datastreams, it is deemed a network virtual terminal (NVT). While the datastreams are completely hidden from the programmer, some methods are not available when in NVT mode.

The majority of applications do not need to allow for this mode of operation because they communicate exclusively in 3270 mode.

Caution – Data flows from the terminal to the server (and vice versa) are in plain text over a TCP/IP connection. This means that all data (including passwords) flow in an insecure manner.

CHAPTER 2

Required Software

This chapter contains the following topics:

Development Environment

Sun 3270 Pathway is written entirely in the Java programming language. To develop an application that uses it, you must have a development environment that supports Java version 1.4.

Obtaining a Java Runtime Environment (JRE)

A Java runtime environment (JRE) version of at least 1.4 is required to run any of the Sun 3270 Pathway products.

JRE software and JDKs for all platforms can be obtained from `http://java.sun.com`.

The Solaris 9 operating system and later releases of the Solaris 8 OS have a bundled 1.4 JDK.

For Microsoft Windows users, a 1.4 JRE is supplied with the Sun 3270 Pathway software.

Obtaining a Java-enabled Web Browser

To run a Java applet that uses Sun 3270 Pathway, a web browser that supports a Java virtual machine (JVM™)[1] of at least version 1.4 is required. The JVM is part of the JRE and JDK and can be obtained from `http://java.sun.com`.

TCP Router

If you plan to deploy Java applets that use Sun 3270 Pathway, you might encounter the following problem due to a Java security restriction:

> A Java applet can only open a socket connection, in this case, a TN3270 connection to the web server machine from which that applet was served. Because the web server machine might not be the machine on which your TN3270 server resides, you might not be able to deploy the applets successfully.

To address this problem, you can run the TCP Router software on your web server machine to route the socket connection from the web server machine to your TN3270 host. The TCP Router is distributed with Sun 3270 Pathway. See Chapter 10.

1. The terms “Java Virtual Machine” and “JVM” mean a virtual machine for the Java platform.

CHAPTER 3

Installation

This chapter contains the following topics:

Installation Locations

Install Sun 3270 Pathway on your development machine. Later, if you need to deploy a Java applet to use Sun 3270 Pathway, you can also install it on the web server machine from which that applet is served.

After you install Sun 3270 Pathway, the following structure is created:

```
/3270_Pathway2.0.0
/3270_Pathway2.0.0/bin
/3270_Pathway2.0.0/doc
/3270_Pathway2.0.0/doc/com
/3270_Pathway2.0.0/doc/com/sun
/3270_Pathway2.0.0/doc/com/sun/emp
/3270_Pathway2.0.0/doc/com/sun/emp/pathway
/3270_Pathway2.0.0/doc/com/sun/emp/pathway/bean
/3270_Pathway2.0.0/etc
/3270_Pathway2.0.0/JRE  (Only with the InstallShield packaged products)
/3270_Pathway2.0.0/lib
/3270_Pathway2.0.0/samples
/3270_Pathway2.0.0/samples/bean
/3270_Pathway2.0.0/samples/recorder
```

This document refers to the top level directory as *INSTROOT*, which is `/3270_Pathway2.0.0` in this example.

The Java Archive (JAR) files that contain the Sun 3270 Pathway products are shipped in the *INSTROOT*`/lib` subdirectory:

INSTROOT`/lib/pathway_bean.jar` contains the Sun 3270 Pathway Bean.

INSTROOT`/lib/pathway_recorder.jar` contains the Sun 3270 Pathway Navigation Recorder.

INSTROOT`/lib/pathway_router.jar` contains the Sun 3270 Pathway Router.

INSTROOT`/lib/pathway_terminal.jar` contains the Pathway Terminal.

Caution – Do not unpack the JAR files. They work as single files and do not require unpacking.

The HTML API documentation for programming the Sun 3270 Pathway Bean is located in the following directory:

INSTROOT`/doc/index.html`

Installing Sun 3270 Pathway

This section describes how to install Sun 3270 Pathway on UNIX and Microsoft Windows platforms.

If you use Microsoft Windows, Sun 3270 Pathway is available as an InstallShield executable. This executable contains a 1.4 version of the JRE along with Sun 3270 Pathway.

The InstallShield installation process creates a program group and other associated entries on your Start menu.

Note – The InstallShield version of Sun 3270 Pathway does not contain a Java compiler. If you want Sun 3270 Pathway to compile the source code classes it creates, you must ensure that the Java compiler, `javac`, is in your path before you start Sun 3270 Pathway.

You can obtain the `javac` compiler by downloading the JDK from `http://java.sun.com`. If you are using an IDE, a copy of `javac` might already exist on your system.

▼ To Install Sun 3270 Pathway on Solaris and UNIX Systems

1. **Locate the `*.zip` file that contains the Sun 3270 Pathway software.**
2. **Uncompress the archive under an appropriate directory.**

   Refer to the `ZIP`(1) man page for information about command options.

▼ To Install Sun 3270 Pathway on Microsoft Windows Systems

1. **Locate the `*.exe` file that contains the Sun 3270 Pathway software.**
2. **Start the installation using either method:**
   - Double-click the `.exe` file.
   - On the desktop, select Start → Run, type the name of the Sun 3270 Pathway executable, and click OK.
3. **Follow the on-screen directions to complete the installation, making sure to select the desired installation location.**

   After the installation is finished, you can use the Sun 3270 Pathway software.

CHAPTER 4

Using the Sun 3270 Pathway Bean

This chapter explains how to use the Sun 3270 Pathway Bean in your environment of choice. The Sun 3270 Pathway Bean is located in the JAR file `pathway_bean.jar`, which is located in the *INSTROOT*`/lib` directory. This chapter contains the following topics:

Using Sun 3270 Pathway Bean With the JDK

If you want to use the JDK to write programs that use the Sun 3270 Pathway Bean, set your `CLASSPATH` environment variable to reference the Pathway Bean JAR.

For the UNIX Korn shell:

```
$ export CLASSPATH=$CLASSPATH:INSTROOT/lib/pathway_bean.jar
```

For Microsoft Windows:

```
C:\> set CLASSPATH=%CLASSPATH%;INSTROOT\lib\pathway_bean.jar
```

In both cases, replace *INSTROOT* with the name of the top-level directory where you installed Sun 3270 Pathway.

Using Sun 3270 Pathway Bean in an IDE

Because the Sun 3270 Pathway Bean is a Java Bean, it can be used in any environment where a standard Java Bean can be used; for example, Sun ONE Studio.

To use the bean in your environment, refer to the documentation for your IDE for information about adding a Java Bean.

CHAPTER 5

Sun 3270 Pathway Sample Programs

The sample programs included with Sun 3270 Pathway illustrate how to use it. This chapter provides a brief description of these programs.

The Sun 3270 Pathway Bean samples described in "Sun 3270 Pathway Bean Samples" on page 11 are located in the directory:

INSTROOT`/samples/bean`

The Pathway Recorder sample described in "Pathway Recorder Sample" on page 12 is located in the directory:

INSTROOT`/samples/recorder`

For more information about how to invoke these samples, refer to the comments included in the source files. For information about the `ACCT` transaction, refer to the *Sun Mainframe Transaction Processing Software Installation Guide*.

Sun 3270 Pathway Bean Samples

`Sample1.java`

`Sample1.java` shows how to:

- Create an instance of a Terminal object
- Make it visible in a window
- Connect it to a TN3270 server

After it is visible and connected to the end-system, the user can interact with Sun 3270 Pathway.

Sample1Applet.java

`Sample1Applet.java` is a version of `Sample1.java` that runs as a Java applet. A Java security restriction prevents a Java applet from connecting to any host other than the one from which it is served. For information about how to work around this problem, see Chapter 10.

Sample2.java

The `Sample2.java` program is the same as `Sample1.java`, but also listens for and prints the events raised by the Terminal object. The event listening is done in the `Sample2EventListener.java` class.

Sample3.java

`Sample3.java` is a more sophisticated example in which a nonvisible Terminal is used to extract information from a 3270 application running in a Sun Mainframe Transaction Processing (Sun MTP) region.

To run this sample requires a running region with the `ACCT` transaction available.

This sample takes a host, port, and surname as parameters and returns the numbers of all the accounts available to the `ACCT` transaction that have that surname.

Pathway Recorder Sample

ACCTSample.java

`ACCTSample.java` is an example demonstrating how to invoke a Pathway Recorder-generated class `ACCTBean.java`.

To run this sample requires a running region with the `ACCT` transaction available.

It performs a similar task to `Sample3.java`, but is using a generated bean to perform the navigation on the Terminal. This sample takes host, port, and surname parameters and returns the first account available to the `ACCT` transaction that has that surname.

CHAPTER 6

Sun 3270 Pathway Bean Programmable Interface

The Sun 3270 Pathway Bean supports a programmable interface that enables developers to have direct access to manipulate and customize the 3270 emulator for their programs.

This chapter contains the following topics:

Available Classes

Within the `com.sun.emp.pathway.bean` package there are three classes:

TABLE 6-1 Pathway Bean Classes

Class	Description

`Terminal`	The actual 3270 emulator. This is a subclass of `javax.swing.JComponent` and can be made visible and used in the same way as other Java Foundation Classes (JFC/Swing) components. However, it is not necessary to display a Terminal. It is fully functional whether it is visible or not.
	When a Terminal is visible, it displays a customizable representation of the 3270 screen. When not visible, there is a slight performance advantage.
	This class generates two event types:
	`java.beans.PropertyChangeEvent`
	`com.sun.emp.pathway.bean.TerminalEvent`

`TerminalEvent`	One of the event types generated by the `Terminal` class.
`TerminalField`	The representation of a 3270 field.

There are two Java interfaces:

TABLE 6-2 Pathway Bean Interfaces

Interface	Description
`TerminalListener`	An interface that classes should implement when listening for `TerminalEvent` events.
`TerminalCondition`	An abstract condition used in waiting for a terminal.

There is one exception:

TABLE 6-3 Pathway Bean Exception

Exception	Description
`TerminalConditionException`	An exception thrown as a result of a runtime exception being thrown by a `TerminalCondition`.

Most programming is done directly to the `Terminal` class. For a complete description of the available classes and methods, refer to the HTML API documentation that is located at:

INSTROOT`/doc/index.html`

Instantiating a Terminal Object

The Java Bean specification states that all beans should have a zero parameter constructor. The primary method of constructing a Terminal object uses the zero parameter constructor.

The `Terminal` class has two constructors, as shown in the following table:

Constructor	Description
`public Terminal()`	This zero parameter constructor is the primary means for creating a Terminal object. It sets all its properties to default values.
`public Terminal (Terminal terminalSource)`	This constructor enables an application to have two views of the same underlying TN3270 connection. After the new object is created, both the source and created objects share the underlying transportation mechanisms and can be thought of as representing the same connection. The `public Terminal (Terminal terminalSource)` constructor enables the application to present the 3270 screen to the user in different ways. All network-type properties of the emulator are shared between the two Terminal objects. For example, if you call `setTN3270Host("fred")` on one, the result of calling `getTN3270Host()` on the other is `"fred"`. All presentation-type properties of the emulator are separate for the two Terminal objects. For example, if you call the `setFont(f)` on one, the other emulator is not affected. A Terminal that was created using this constructor is considered to be entirely symmetric with the `terminalSource` Terminal from which it was constructed, that is, equal siblings rather than parent and child.

Connecting a Terminal

When you create an instance of a Terminal, it is not connected to any host system. You use the `connect()` method with a valid set of values assigned to the following properties to connect a Terminal to a TN3270 host.

Property	Description
`host`	The host to which to connect this emulator. Default is `localhost`. Set the property using the `setTN3270Host(String host)` method.
`port`	The TCP/IP port to which to connect. Default is `2001`. Set the property using the `setTN3270Port(int portnumber)` method.
`model`	The terminal model to emulate. Default is `Model 2-E`. Set the property using the `setModel(int model)` method.

The `connect()` method is an asynchronous call. Attempting to call it when the connection state is anything other than disconnected results in the generation of an `IllegalStateException`.

The following properties also affect the connection process:

Property	Description
`TN3270EAllowed`	If true, the Terminal uses the TN3270E protocol if the host asks it to do so. This is the default behavior. If false, the Terminal acts as if it does not support the TN3270E protocol. Set the property using the `setTN3270EAllowed(boolean allowed)` method.
`preferredNetname`	If this property has a non-empty value, the Terminal attempts to connect to the TN3270E system as the specified netname. Default is empty. Set the property using the `setPreferredNetname(String netname)` method.

The following example shows an easy way to create and start a 3270 emulator:

```
Terminal term = new Terminal();
term.setTN3270Host("www.myhost.com");
term.setTN3270Port(9993);
term.connect();
```

There are special considerations when connecting in an applet environment. Normally, the TN3270 host must be the same as that from which the HTML of the containing web page was served. To obtain this information, you can use the `getDocumentBase()` method in the `java.awt.Applet`; for example:

```
term.setTN3270Host(getDocumentBase().getHost());
```

This technique is illustrated in the `Sample1Applet.java` program.

Determining When a Terminal is Connected

A Terminal has four connection states:

- Connected
- Connecting
- Disconnecting
- Disconnected

You can think of it as a single property that has one of four values, or you can view it as four separate boolean properties:

- Connected / not connected
- Connecting / not connecting
- Disconnecting /not disconnecting
- Disconnected /not disconnected

Both views are supported by the properties of Terminal:

The `getConnectionState()` method returns an `int` representing one of the four connection states.

The `isConnected()`, `isConnecting()`, `isDisconnected()`, and `isDisconnecting()` methods return a boolean equal to `true` or `false` depending on the connection state of the 3270 emulator.

These five methods correspond to five of the Java Bean properties of the `Terminal` class. For a complete list of all the `Terminal` class properties, refer to the Javadoc™ supplied with the Sun 3270 Pathway product.

The five connection-related properties are *bound* properties, which means that the `Terminal` class raises `java.beans.PropertyChangeEvent` events when any of these properties change. For additional information about property change events and how to listen for them, refer to your JDK API documentation. Also examine the source code of the `Sample2.java` program.

Determining When the Keyboard is Locked

The `keyboardLockState` property determines if the keyboard is locked. You can use this property to determine when the host system has finished processing the last instruction and has returned control back to the user. Use the `isKeyboardLocked()` method to determine the keyboard lock state. The `Sample3.java` program illustrates the use of this property.

`keyboardLockState` is also a bound property; therefore, `java.beans.PropertyChangeEvent` events are raised whenever the keyboard lock state changes.

Disconnecting a Terminal

A Terminal can become disconnected in one of the following ways:

- A timeout. A timeout occurs when there has been no network activity for a period of time. When this timeout occurs, the Terminal automatically disconnects itself. You can control this timeout period programmatically using the `setNetworkInactivityTimeout()` method.
- The TN3270 host can terminate the connection. For example, when a user of a terminal connected to Sun MTP invokes the `CSSF LOGOFF` transaction.
- The program issues a call to the `disconnect()` method of a Terminal.

When you are finished with a Terminal, you should disconnect it from the host system either by issuing a `disconnect()` call or by logging off the host system. This minimizes resource usage on the host system.

When a Terminal is connected, a number of Java threads are active, raising events and maintaining the network connection. This means that a connected Terminal is not eligible for garbage collection by the Java runtime environment. Therefore, you must ensure that a Terminal is disconnected before it is dereferenced, to enable the cleanup of the resources associated with the Terminal. Perform a call to the `dispose()` method of the Terminal when the program has finished using it to ensure an efficient cleanup of its resource.

Terminal Wait Methods

The Terminal has several methods that enable the application to wait until something significant has happened:

```
waitUntilDisconnected()
waitUntilConnected()
waitUntilKeyboardUnlocked()
waitHeuristic()
waitCondition(TerminalCondition terminalCondition)
waitForReadableString(String readableString, int offset)
```

In NVT mode, the keyboard is always unlocked. Therefore, the `waitUntilKeyboardUnlocked()` method always returns immediately.

Obtaining Information From a Terminal

A 3270 terminal can have two types of display: formatted (the screen consists of 3270 fields) and unformatted. This enables you to obtain information from a Terminal on a field basis or a screen buffer basis.

You can interrogate a Terminal object using the `isFormatted()` method to determine if the current screen is formatted or unformatted. When the terminal is in NVT mode, there are never any fields; therefore, the `isFormatted()` method returns `false`.

Obtaining Field Information

For a formatted display, each 3270 field is represented by a TerminalField object. You can obtain a field from a Terminal by calling one of the following methods:

- The `getFields()` method returns all the fields from a 3270 emulator screen in a Vector variable.
- The `findField()` method returns the TerminalField containing the specified screen location.

After a TerminalField is obtained, you can interrogate it using one of the methods described in the Javadoc API of TerminalField. For example, you can obtain the text from a TerminalField using the `getText()` method.

You can programmatically modify an unprotected TerminalField using the `setText()` and `setData()` methods. These methods alter the contents of the field the same way a terminal user can alter it. An attempt to modify a protected field results in an `IllegalStateException`.

Note – You can only obtain fields on formatted 3270 screens. Attempting to perform `getFields()` on an unformatted screen results in an empty vector. Attempting to perform `findField()` on an unformatted screen returns `null`.

Obtaining Screen Buffer Information

There are three buffers that describe the screen contents:

- Display: Holds characters on the screen.
- Extended attribute: Holds extended attributes of the screen positions.
- Color: Holds extended colors of the screen positions.

Performing the appropriate `get` method for each returns an array of characters. Because they are copies of the buffers, modifications have no effect on the 3270 terminal.

You can enter data on a formatted or unformatted screen using the `typeChar()` method of the `Terminal` class. You can obtain buffers from both formatted and unformatted screens.

You can also obtain an area of the screen using the `getReadableString()` method. This method returns an area of the screen as it appears to the user, that is, with attribute bytes and nulls returned as spaces.

When the terminal is in NVT mode, the color and extended attribute buffers have no meaning and are filled with default values.

Screen Change Notification

When the host sends a data stream to a Terminal, it can cause the display to change in a manner asynchronous to an application program using that Terminal.

To tell an application program about an occurrence, the `Terminal` class raises `com.sun.emp.pathway.bean.TerminalEvent` events. An application program listening for these events must implement the `com.sun.emp.pathway.bean.TerminalListener` interface. These events are generated regardless of the terminal mode.

The `TerminalListener` interface currently has a single method, `hostChangedScreen()`, which is called when the screen is changed by a data stream from the host system, indicating that new data was received from the system.

You can register and deregister listeners of this type using the Terminal methods `addTerminalListener()` and `removeTerminalListener()`.

This event is not implemented as a `java.beans.PropertyChangeEvent` because it does not have an associated value that changes.

Supplying Data to a Terminal

A user can manipulate a visible Terminal by clicking it to give it focus, then using the keyboard to enter data.

There is also a set of Terminal methods to perform the following functions:

- Pressing specific 3270 keys
- Manipulating the cursor
- Typing characters to the 3270 emulator

When a terminal is in NVT mode, a restricted set of methods are available. Calling methods that have a pure 3270 meaning when in NVT mode generates an `IllegalStateException`. Refer to the Javadoc™ supplied with Sun 3270 Pathway for a list of Terminal methods.

The Visible Terminal

When the Terminal is made visible, you can see that it is made up of two areas: the emulator area and the status bar at the bottom. The status bar can be hidden by calling the method `setStatusBarShowing(false)`.

The Status Bar

The status bar displays information about the Terminal and is divided into sections.

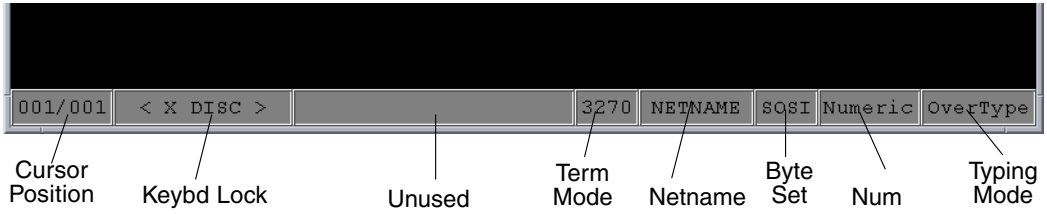

FIGURE 6-1 Status Bar

The labels in FIGURE 6-1 indicate each section of the status bar and are described in the following table:

Label	Description
Cursor Position	Indicates the row and column numbers of the cursor position.
Keybd Lock	Indicates if the keyboard is locked.
Unused	Indicates that this section is not used.
Term Mode	Indicates the terminal mode. Displays `3270`, `NVT`, or `SUSP`.
Netname	indicates the netname of the terminal, if there is one.
Byte Set	The byte set of the field that the cursor is currently on: either `SBCS` (pure single byte), `SOSI` (a combination of single and double byte), or `DBCS` (pure double byte). These values are displayed only when the host is connected using a double-byte codepage. If the terminal is connected using a single-byte codepage, this area is blank.
Num	Indicates whether the field is numeric. Displays `Numeric` if the current field is numeric, or else it is blank.
Typing Mode	Indicates the typing mode. This displays either `Insert` or `OverType`.

Keyboard Handling

The Terminal provides a default mapping of keystrokes to functionality within the Terminal.

- If you do not want the handling to occur, you must subclass Terminal and override the `protected void processKeyEvent(KeyEvent terminal)` method.

- If you want to remap the keyboard to function mapping, you should manipulate the `javax.swing.InputMap` of the Terminal to make the associations between the keys and the functions you require.

Automatic Font Resizing

By default, the Terminal is set to automatically attempt to set the font size of the emulator so the whole of the emulator fills the container it is placed on. If the container changes size, the font size might increase or decrease accordingly. To turn off this behavior, call the method `setAutoFontResizingEnabled(false)`.

Current and Maximum Display Sizes

Terminal has the following properties in the property list:

Current Display Property	Maximum Display Size Property
`rows`	`maximumRows`
`columns`	`maximumColumns`
`displaySize`	`maximumDisplaySize`

Model 3, 4, and 5 terminals allow the host system application to set the display size of the data on the screen when it sends a datastream to the Terminal. If the host system sends data instructions to use the alternate screen size, the current values equal the respective maximum values. If using the non-alternate screen size, the current values are set to the default values of a 3270 model 2 terminal, 80 rows, 24 columns, 1920 characters for the `displaySize`.

The following table defines the maximum values for each terminal model.

TABLE 6-4 Maximum Display Values for Terminals

Model	maximumRows	maximumColumns	maximumDisplaySize
3	32	80	2560
4	43	80	3440
5	27	132	3564

CHAPTER 7

Introduction to Sun 3270 Pathway Recorder

This chapter describes how to use the Sun 3270 Pathway Recorder to completely encapsulate access to a 3270 end-system within an opaque Java object. It contains the following topics:

Recorder Concepts

Sun 3270 Pathway Recorder is a tool that enables a developer to create Java source and object code that corresponds to a fixed navigation through a portion of a 3270 application. It records the keystrokes that a user performs and produces Java code that recreates these actions. If the remote system sends non-3270 data to the recorder, the connection is terminated. The recorder does not permit the recording of non-3270 sessions.

This example uses the Sun MTP sample transaction `ACCT`, which is a 3270 application that enables a 3270 user to determine the account number of a person when that person's surname and first name are supplied. This transaction displays these 3270 screens.

The user starts the transaction by typing `ACCT` on a blank transaction screen and pressing Enter. See FIGURE 7-1.

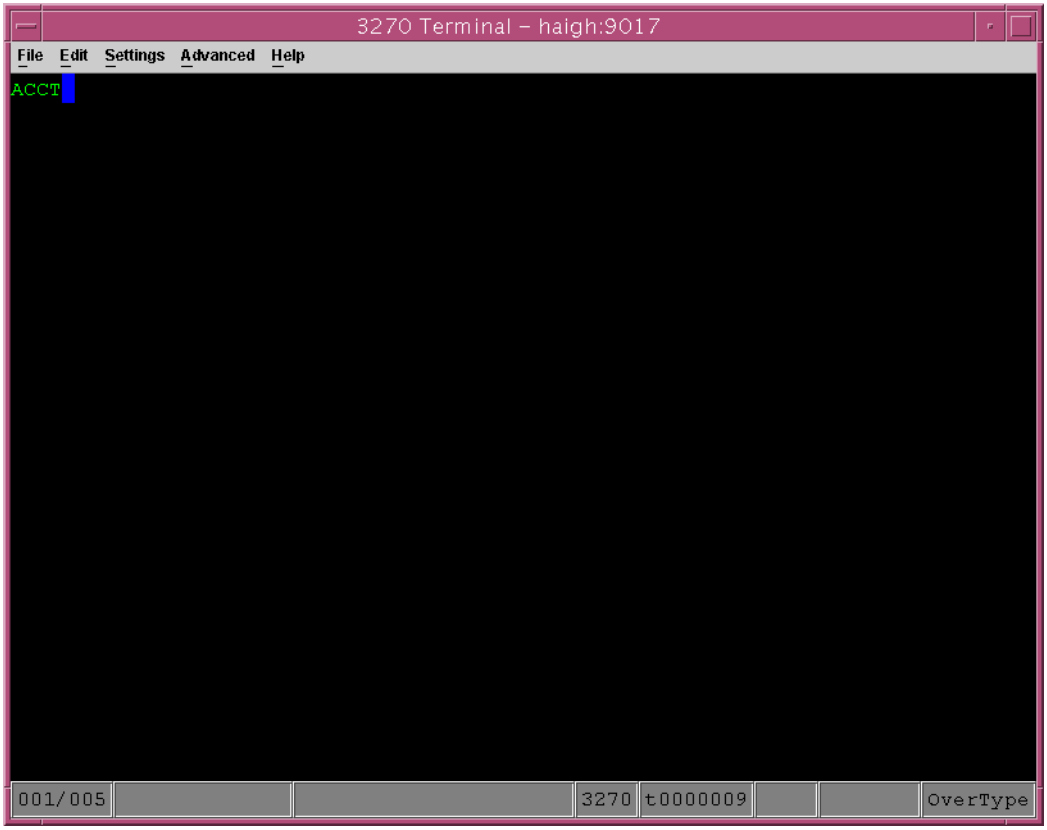

FIGURE 7-1 ACCT Transaction Screen

When the ACCT main screen is displayed, the user types the surname and first name of the person being queried and presses Enter. See FIGURE 7-2.

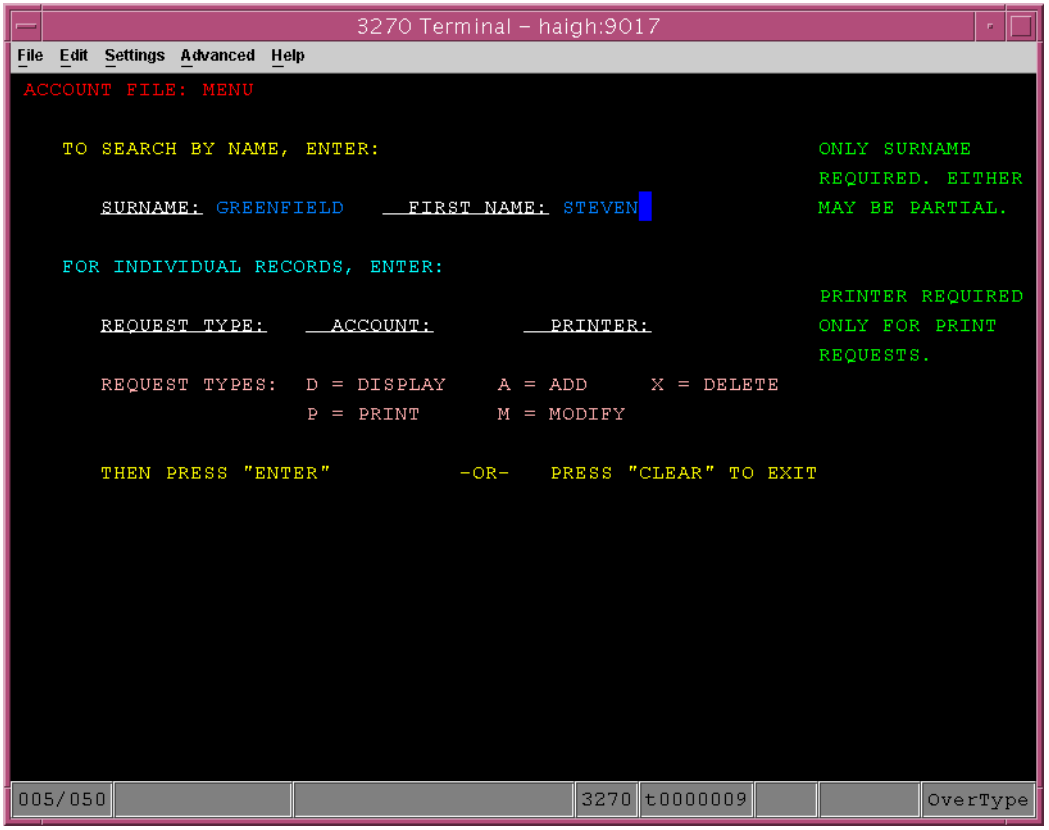

FIGURE 7-2 ACCT Transaction Main Menu

FIGURE 7-3 shows the details for the queried person.

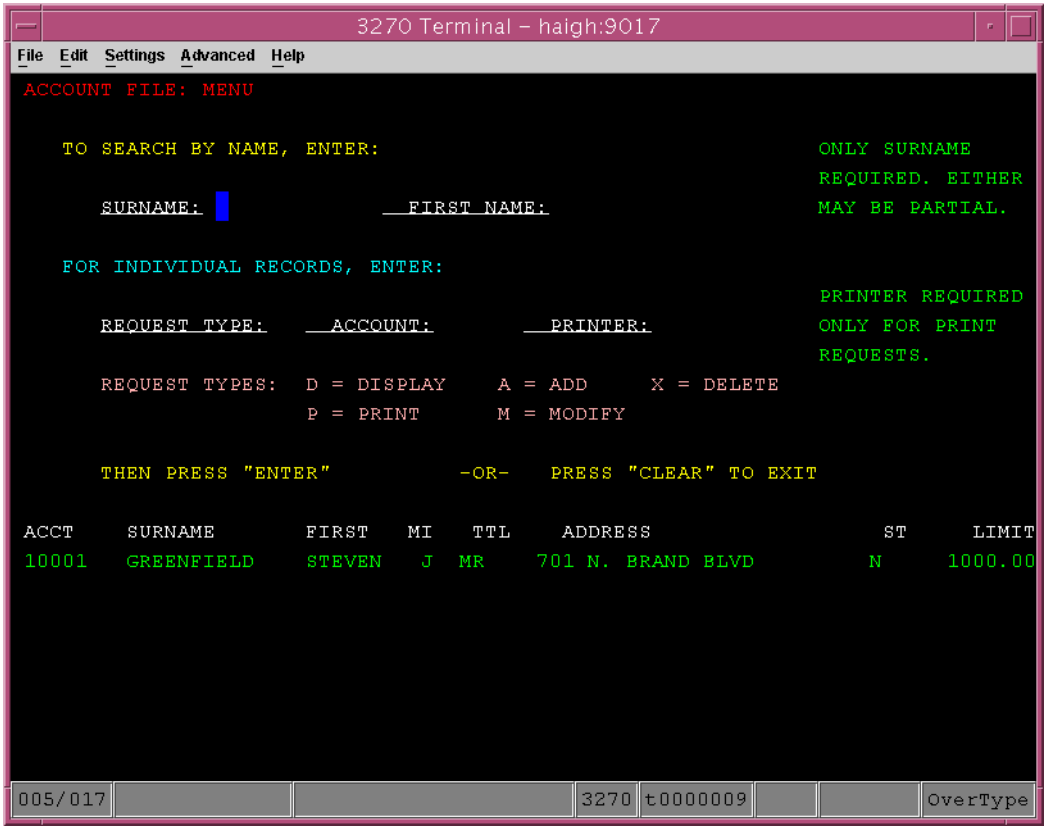

FIGURE 7-3 ACCT Search Results

The user ends the transaction by pressing Clear in the 3270 emulator screen.

When programming in Java, create a single Java object to which the input parameters (`setSurname` and `setFirstName`) are initially supplied, and from which to obtain the output parameter (`getAccountNumber`). For example:

```
AccountQueryObject aqo = new AccountQueryObject();
aqo.setSurname("Smith");
aqo.setFirstName("John");
aqo.performWork();
String s = aqo.getAccountNumber();
```

where the `aqo.performWork()` call instructs the `AccountQueryObject` to obtain the account number of the specified user.

Using the Sun 3270 Pathway Bean, you can implement classes such as `AccountQueryObject` by making calls to the Sun 3270 Pathway Bean as follows:

```
term.typeString("ACCT");
term.pressEnter();
```

However, this is a tedious task for a large 3270 application and is error-prone. Sun 3270 Pathway Recorder enables you to easily create objects, such as `AccountQueryObject`, by interacting with a 3270 emulator.

Starting the Sun 3270 Pathway Recorder

The Sun 3270 Pathway Recorder is installed as part of the Sun 3270 Pathway installation. Follow the procedures in this section to start it on various platforms.

Starting the Recorder on Solaris and UNIX Systems

On Solaris or UNIX systems, there are two ways to start the Recorder.

▼ To Start the Recorder Using the Shell Script

1. **Ensure that the required level of JRE is on your path.**

   See "Obtaining a Java Runtime Environment (JRE)" on page 3.

2. **Run the recorder shell script located in the `bin` directory of the product installation.**

▼ To Start the Recorder Using the JAR File

1. **Ensure that the required level of JRE is on your path.**

   See "Obtaining a Java Runtime Environment (JRE)" on page 3.

2. **Run the executable JAR file located in the `lib` directory of the product installation.**

   Change to the `lib` directory and type the following command:

```
$ java -jar pathway_recorder.jar
```

Starting the Recorder on Microsoft Windows Systems

There are three ways of starting the Recorder.

▼ To Start the Recorder Using the Start Menu

1. **Click the Start button.**
2. **Select Start → Programs → Sun 3270 Pathway 2.0.0 → Recorder.**

▼ To Start the Recorder Using the Recorder Executable

- **Run the `pathway_recorder.exe` program located in the `bin` directory of the product installation.**

▼ To Start the Recorder Using the JAR File

- **Run the executable JAR file located in the `lib` directory of the product installation.**

  Change to the `lib` directory and type the following command:

```
C:\> java -jar pathway_recorder.jar
```

Creating the Navigation Class

After starting the recorder, you can create a navigation class, which is a Java class that drives a Sun 3270 Pathway Bean. For this example, access to a Sun MTP TN3270 end-system with the `ACCT` transaction available is desirable. If you do not have access, the principles still apply. The following sections create a class named `AccountQueryObject` using the recorder. This obtains a user's account number when the user's first name and surname are supplied.

▼ To Make a Connection and Display the Recorder Main Screen

1. **Start the recorder.**

   See "Starting the Sun 3270 Pathway Recorder" on page 31.

2. **On the recorder main screen, select File → New, which displays the New Navigation Class window.**

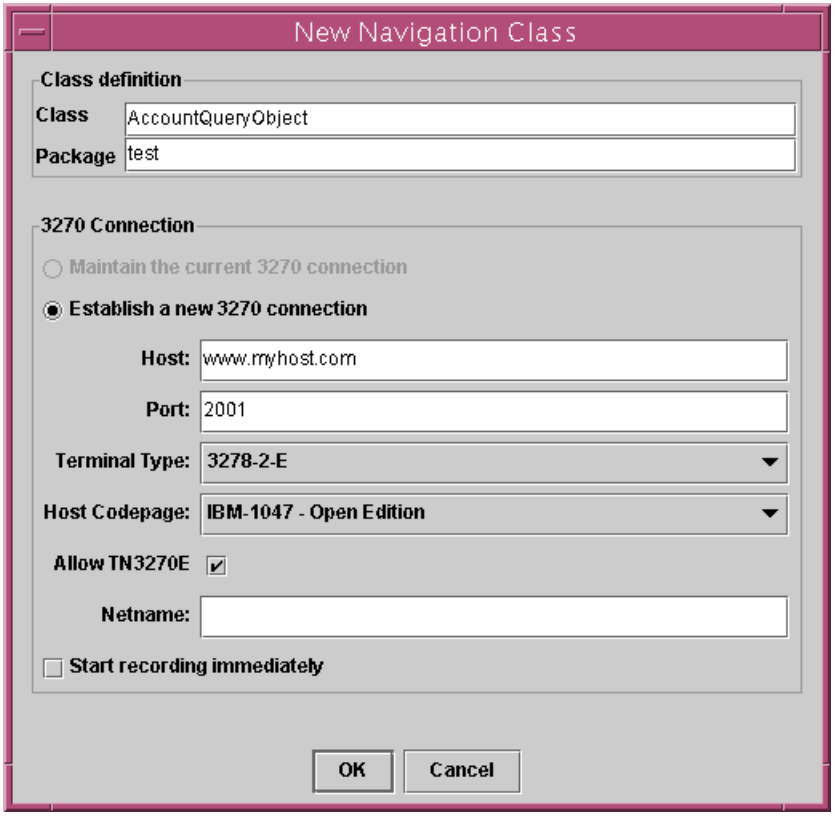

FIGURE 7-4 New Navigation Class Window

3. **Type the name of the Java class, for example,** `AccountQueryObject`**.**

4. **Type the Java package name for this class, for example,** `test`**.**

5. **Type the Host and Port, and select the Terminal Type for the TN3270 end-system, for example:**

```
www.myhost.com
2001
3278-2-E
```

   You also have the option to specify if you want to Allow TN3270E and to specify the netname for the terminal.

6. **Click OK.**

The following figure shows the Sun 3270 Pathway Recorder main screen.

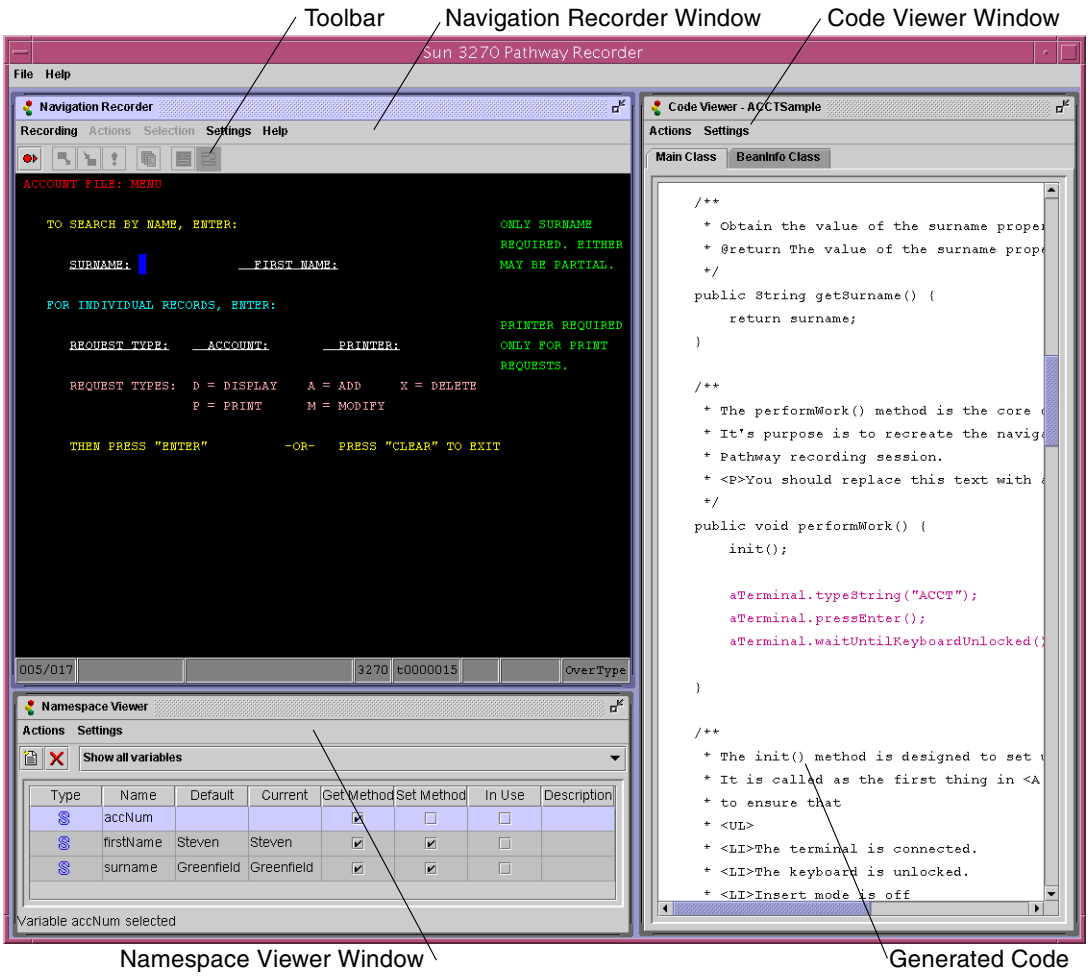

FIGURE 7-5 Sun 3270 Pathway Recorder Main Screen—Example

The sections of this window are as follows:

- The Navigation Recorder window is a 3270 emulator.
- The Toolbar provides controls to start/stop recording and to enter/retrieve variables to/from the emulator.
- The Namespace Viewer window displays all currently defined variables.
- The Code Viewer window shows the Sun 3270 Pathway navigation class being generated.

- The Generated code is the Java-compatible code that calls the Sun 3270 Pathway Bean. It appears dynamically while the user interacts with the Navigation Recorder window.

Starting to Record

When the Navigation Recorder is in recording mode, all interaction with the 3270 emulator causes the generation of code in the Code Viewer window, which calls the Sun 3270 Pathway Bean to perform the recorded actions.

▼ To Start Recording

1. **On the Navigation Recorder window, press Clear to ensure that the emulator displays a blank transaction screen.**
2. **Start recording by selecting either the Start option from the Recording menu of the Navigation Recorder window or by pressing the toolbar button.**
3. **On the 3270 emulator, type the transaction ID `ACCT` and press Enter.**

   The Code Viewer window generates the code as you type.

```
term.typeString("ACCT")
term.pressEnter();
term.waitUntilKeyboardUnlocked();
```

The lines of code in the Code Viewer window are grouped together into *interactions*. Interactions are the actions taken by the terminal user between each occurrence of pressing a 3270 Action Identifier (AID) key. An interaction includes a line of code that waits for the 3270 host to respond to the AID key.

Creating an Input Variable

After you type `ACCT` and press Enter, the window shown in FIGURE 7-2 is displayed in the Navigation Recorder window. Type the surname and first name of the person whose account details you want to query. The surname and first name are values that are supplied to the generated Sun 3270 Pathway navigation class at runtime; in other words, they are input parameters. You define an input parameter by creating a variable in the Namespace Viewer window.

▼ To Create an Input Variable

1. **Select the Create option from the Actions menu or use the Create A New Variable button on the toolbar**
2. **When the Create a variable window is displayed, make the following entries:**
   a. **Type the variable name; for example,** `surname`**.**
   b. **In the Description field, type a brief description of the variable.**
   c. **In the Type area, select** `String`**.**
   d. **In the Current Value field, type a value; for example,** `Greenfield`**.**

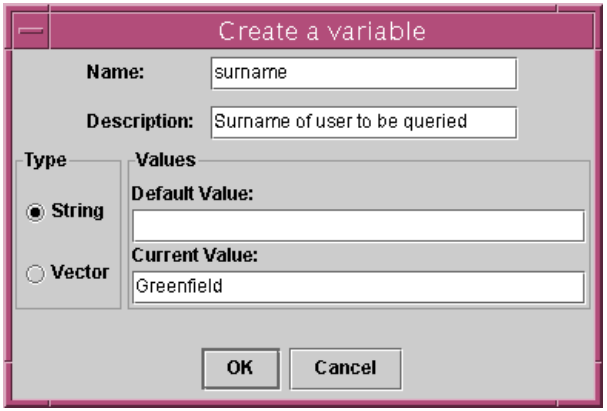

FIGURE 7-6 Create a Variable

3. **Click OK.**
   In the Namespace Viewer window an entry for the `surname` variable is displayed.

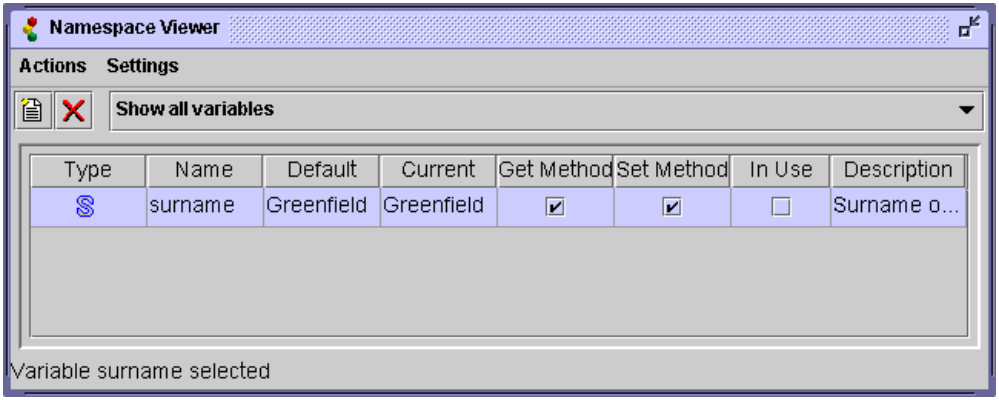

FIGURE 7-7 Namespace Viewer Displaying a Variable

In the Code Viewer window the following lines of code are generated:

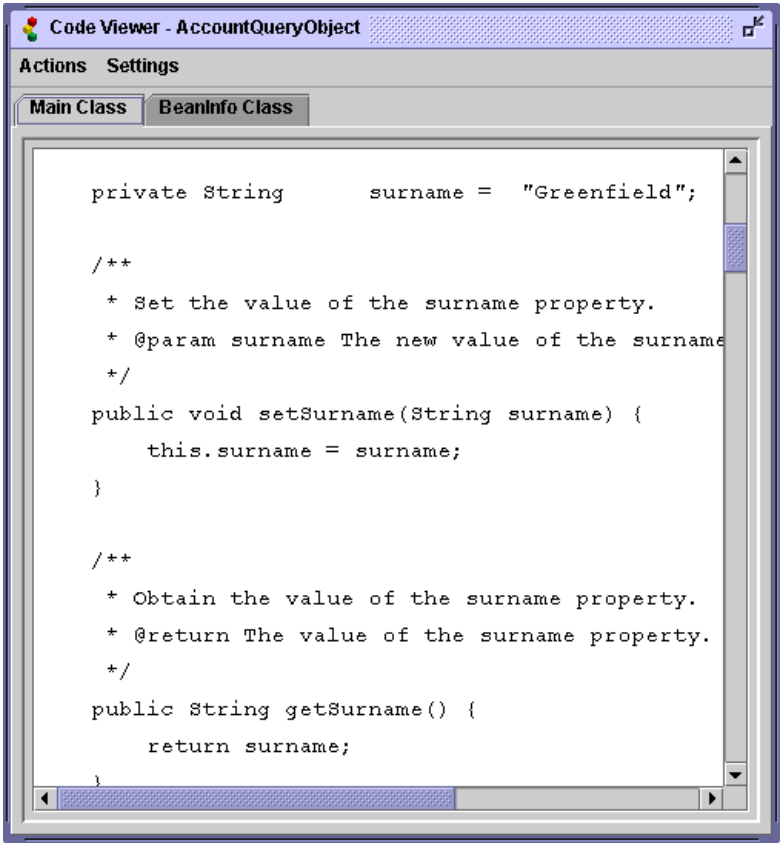

FIGURE 7-8 Code Viewer Window

This code shows the declaration of a String variable named `surname` and a pair of accessor methods named `setSurname` and `getSurname`, which you can use to access this variable from outside the generated class.

Note – The `surname` variable is initialized to a null string because an empty default value was supplied for the variable in the creation dialog. The current value supplied in the creation dialog (Greenfield), does not appear in the generated code, but is applied for the purposes of the recording session.

After an input parameter is defined, the recorder can enter it at the 3270 emulator.

▼ To Select the Variable

1. **On the Actions menu select the Type a variable option or use the toolbar button.**

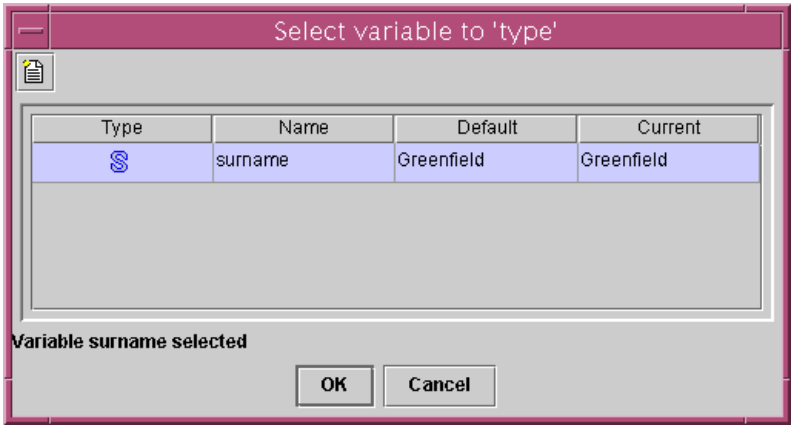

FIGURE 7-9 Select Variable to Type

2. **Select the** `surname` **variable.**
3. **Click OK.**

   Two things happen:

   1. The current value of the `surname` variable (for example, Greenfield) is entered into the 3270 emulator as if a user had typed the string.
   2. The generated code is displayed in the Code Viewer window as:

      ```
      term.typeString(surname);
      ```

4. **You can now press the Tab key in the 3270 emulator to move the cursor to the next field where you type the First Name.**

   This generates another line of code:

   ```
   term.tab();
   ```

5. **Repeat the process of creating and entering another input parameter named `firstName` with a value of Steven.**

   After you complete this entry and press Enter, the Code Viewer window shows the following lines of code for the entire interaction:

```
term.typeString(surname);
term.tab();
term.typeString(firstName);
term.pressEnter();
term.waitUntilKeyboardUnlocked();
```

Creating an Output Variable

The 3270 emulator now shows the account number for Steven Greenfield, which are the five characters beginning in row 17, column 2. This is the information that you want to extract. To make this information available to code that calls the navigation class at runtime, you must define a variable.

▼ To Create an Output Variable

1. **Select the Create option from the Actions menu or use the Create A New Variable button on the toolbar**
2. **When the Create a variable window is displayed, make the following entries:**
   a. **Type the variable name; for example, `accNum`.**
   b. **In the Description field, type a brief description of the variable.**
   c. **In the Type area, select `String`.**

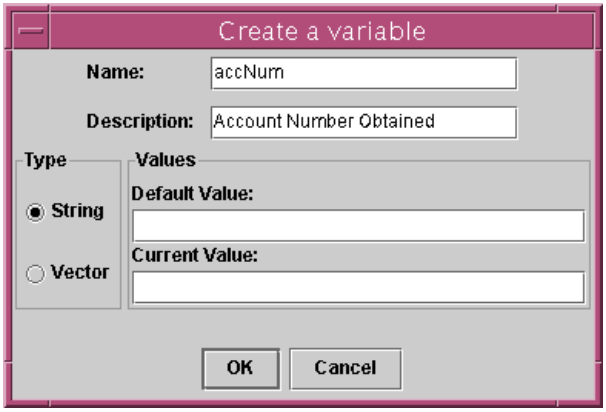

FIGURE 7-10 Create an Output Variable

3. **Click OK.**

Note – Because you are going to use this variable as an output variable, it is not necessary to specify a default or current value.

After you create the variable, you can select it.

▼ To Select the Variable

1. **Make sure you are in Flow Selection mode.**

   On the Navigation Recorder window, select Selection → Flow Selection, or click the Flow Selection toolbar button.

2. **Using a click-and-drag action, select the five characters from the 3270 emulator screen where the Account Number appears.**

   This action does not generate any code.

3. **After the area is highlighted, you can select the Store Highlighted Area option from the Actions menu or use the corresponding toolbar button.**

   The dialog box shown in FIGURE 7-11 is displayed.

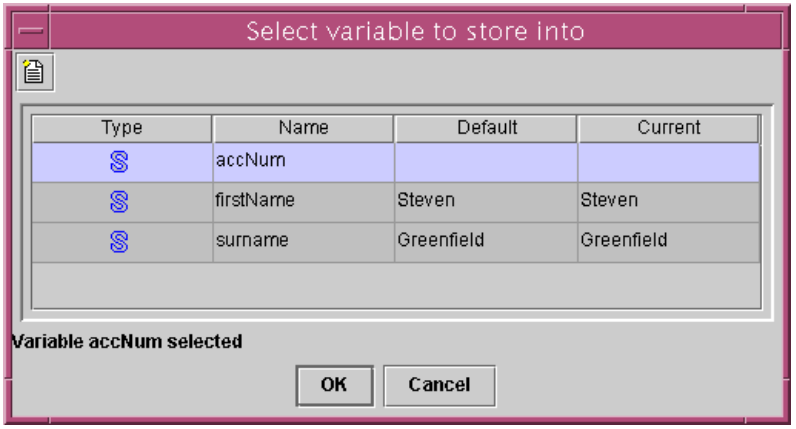

FIGURE 7-11 Select Variable to Store

4. **Select the `accNum` variable that you just created.**
5. **Click OK.**
6. **The following code is generated in the Code Viewer window:**

   ```
   accNum = term.getReadableString(1281,5);
   ```

   The generated code extracts the five-digit account code from offset 1281 of the 3270 emulator's display buffer using the `getReadableString()` method of the Sun 3270 Pathway Bean. This means that any character on the 3270 emulator that looks like a space character is returned as a space, even if it is a null or attribute byte. For additional information about `getReadableString()`, refer to the Sun 3270 Pathway Bean API definition.

Completing the Recording

This section describes how to complete the recording process and save the navigation class.

▼ To Finish the Recording Process

1. **Press Clear on the 3270 emulator screen to exit the `ACCT` transaction.**
2. **Type the `CSSF LOGOFF` transaction and press Enter on the 3270 emulator to exit the terminal session.**
3. **Select the Stop option from the Recording menu of the Navigation Recorder window or use the corresponding toolbar button.**

▼ To Save the Navigation Class

1. **On the recorder main window, select File → Save.**
2. **In the dialog box, verify the class and package names.**

   If they are not correct, type the correct names in the Class and Package fields.
3. **Select a directory into which to save the output files.**
4. **Click Save.**

   See "To Save Navigation Classes to the Disk" on page 47 for more information about the dialog box.

Generated Output Files

Based on the previous example, the following output files are generated for a class named `AccountQueryObject`:

```
AccountQueryObject.java
AccountQueryObjectBeanInfo.java
AccountQueryObject.ManifestStub
AccountQueryObject.jar
```

`AccountQueryObject.java` is the source code of the Sun 3270 Pathway-generated class. You can use this source file in the development environment of your choice (along with the Sun 3270 Pathway Bean) to build your own Java applications.

`AccountQueryObjectBeanInfo.java` and `AccountQueryObject.MaifestStub` are files required to compile `AccountQueryObject.java` and turn it into a Java Bean.

A built version of this Java Bean is generated as `AccountQueryObject.jar`. You can include this precompiled Java Bean into your development environment instead of the Java source code.

Calling the Navigation Class

The following example shows how to call a Sun 3270 Pathway navigation class from Java code.

```
AccountQueryObject aqo = new AccountQueryObject();
aqo.setSurname("Smith");
aqo.setFirstName("John");
aqo.performWork();
String s = aqo.getAccountNumber();
```

Examine the source code for the navigation class to see how it works. A sample of a Sun 3270 Pathway-generated bean and a segment of code to call it is in the *INSTROOT*`/samples/recorder` directory.

CHAPTER 8

3270 Pathway Recorder in Detail

This chapter provides detailed information about the 3270 Pathway Recorder. It contains the following topics:

Sun 3270 Pathway Navigation Class

This section describes how to create and save a new Sun 3270 Pathway navigation class.

▼ To Create a Navigation Class

1. **Start the recorder.**

   See "Starting the Sun 3270 Pathway Recorder" on page 31.

2. **Select File → New on the main recorder window to display the New Navigation Class window.**

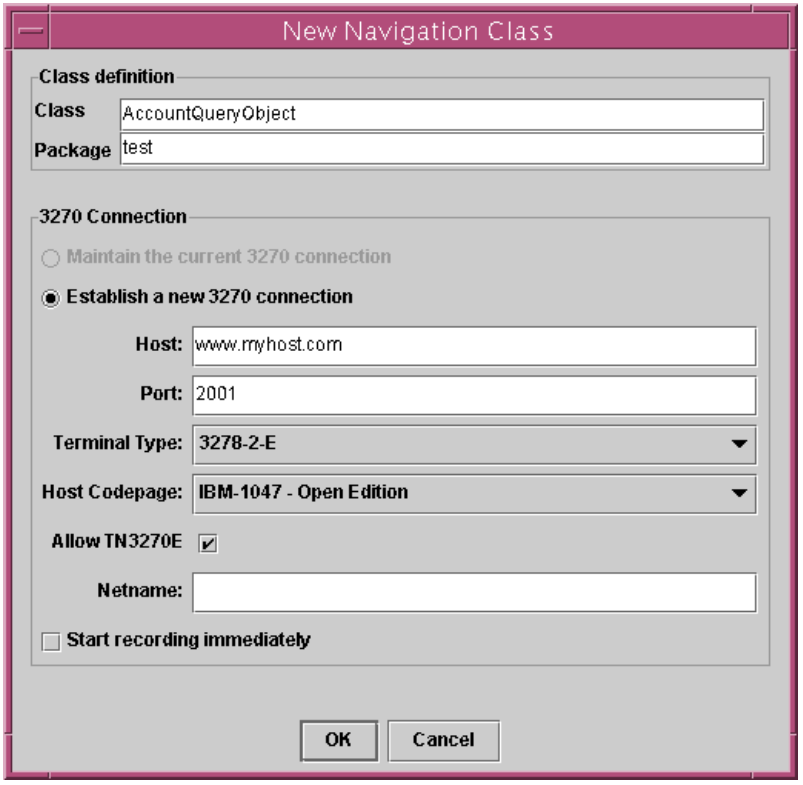

FIGURE 8-1 New Navigation Class Window

3. **In the dialog box, specify the following information:**

   a. **In the Class field, type the name of the navigation class to create, for example** `AccountQueryObject`**.**

      Follow Java naming conventions and start the name with an uppercase letter. The dialog allows you to type only characters that are valid for a Java class name. For example, you cannot use an exclamation mark (!).

   b. **In the Package field, type the package name.**

      The dialog attempts to prevent you from typing an invalid package name. It is a good idea to specify a package name because some IDEs refuse to import a source file (or Java Bean) that is not part of a package.

c. **3270 connection details:**

When you start the recorder, you must specify the TCP/IP host name and port of the TN3270 server to which to connect. You can also choose one of the following terminal models:

- 3278 Model 2
- 3278 Model 2E

d. **If you want the recording process to begin as soon as the terminal is connected, select the Start recording immediately checkbox.**

When recording is turned on, any user interaction with the 3270 emulator generates code in the Code Viewer window, which represents the user interaction.

e. **If you decide to create a new Sun 3270 Pathway navigation class when a 3270 connection is already established, you can choose to maintain that connection by selecting the Maintain 3270 Connection button.**

This is useful when creating multiple navigation classes to perform separate fragments of your 3270 application, which you can subsequently invoke sequentially.

4. **Click OK.**

▼ To Save Navigation Classes to the Disk

1. **On the recorder main screen, select File → Save.**
2. **When the dialog box shown in FIGURE 8-2 is displayed, type the class and package names in the Class definition area.**

You can change class and package names that are displayed in the dialog box.

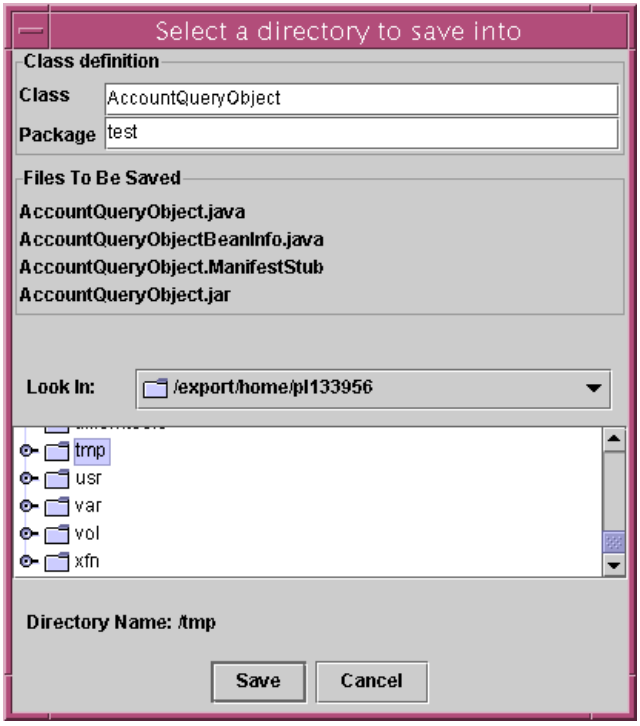

FIGURE 8-2 Saving a Navigation Class

3. **Select the output directory.**

4. **Click the Save button.**

   This initiates the saving and building process and generates the following:

   - Main class file
   - `BeanInfo` class file ("Structure of the `BeanInfo` Class" on page 59)
   - Manifest stub

   The navigation class files are compiled. The compiled class files and the manifest stub are used to create a Java Bean JAR containing the navigation class. A progress bar displays during the save process. If an error occurs at any stage, a window displays the error details.

   When this process completes, all the files are ready for use, either as stand-alone classes or to be imported into IDEs.

Navigation Recorder Window

The Navigation Recorder window contains a 3270 emulator, menu bar, and toolbar.

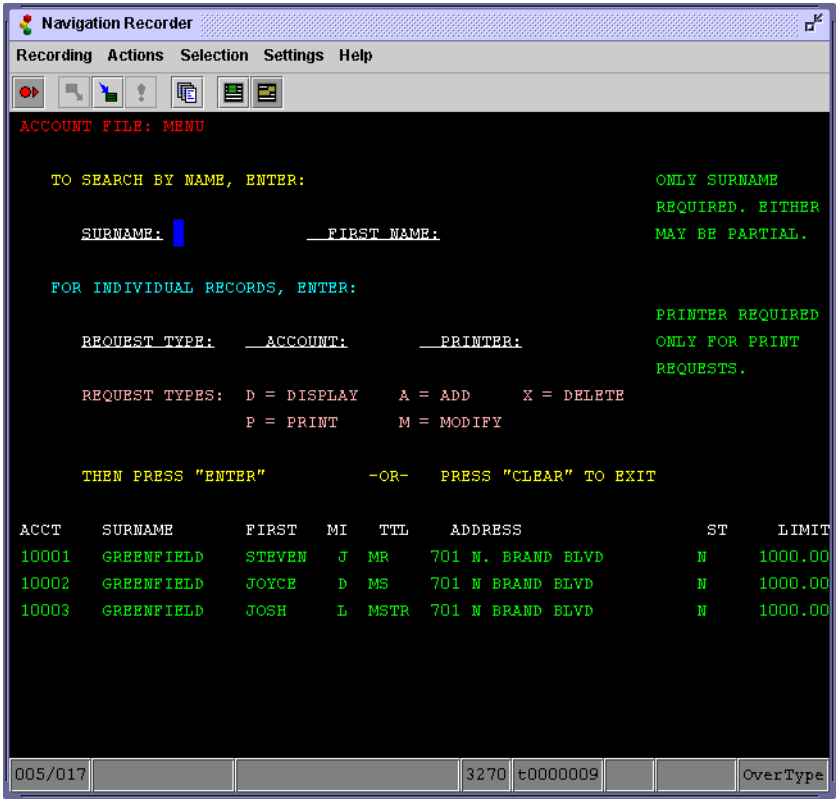

FIGURE 8-3 Navigation Recorder Window

The 3270 emulator enables you to interface with the host system. The toolbar, located at the top of this window, permits easy access to the functions:

- Start/Stop recording
- Store a Highlighted area
- Type a variable
- Assert Highlighted Area
- Store List Wizard
- Set Selection mode to Block
- Set Selection mode to Flow

These functions are also available from the menu bar of this window. The menu bar also enables you to change the font of the emulator by selecting Settings → Font.

Starting and Stopping Recording

The most important control on the Navigation Recorder window is the recording control. When recording is turned on, any user interaction with the 3270 emulator generates code in the Code Viewer window, which represents the user interaction.

▼ To Start and Stop Recording

- **Use either method:**
  - Select the Start or Stop options from the Recording menu.
  - Click the Recording button on the toolbar.

Supplying a Variable

When recording is turned on, you can type the contents of a variable into the 3270 emulator at the current cursor position by selecting the Type a variable option from the Actions menu or using the corresponding toolbar button on the Navigation Recorder.

Selecting this option displays a dialog box from which you can choose the variable whose contents you want typed into the 3270 emulator. You can only type String type variables into the 3270 emulator screen; the dialog box only shows these variables. After you choose a String variable and click OK, the current value of the variable is entered into the 3270 emulator as if a user had typed it. This operation might not succeed if, for example, there is insufficient space in the 3270 field or if the cursor of the 3270 emulator is on a protected area of the screen.

▼ To Create a New Variable

- **Click the Create button on the Navigation Recorder toolbar.**

See "Variable Types" on page 55.

Double-Byte Character Set (DBCS) Issues

When typing the contents of a String variable into the 3270 emulator screen, the string must contain data that is compatible with the screen field into which it is being entered. The following table describes each screen field type.

Field	Description
Pure single-byte	The string can only contain Unicode characters that translate to single-byte code points using the current DBCS code page.
Pure double-byte	The string can only contain Unicode characters that translate to double-byte code points using the current DBCS code page.
SOSI (shift-out shift-in)	The string can contain Unicode characters that translate to either single- or double-byte code points using the current DBCS code page. However, the string cannot contain SO or SI characters because they are generated automatically where required. Therefore, you must ensure that there is sufficient space in the entry field for both the characters and any automatically generated SO or SI characters.

Highlighting an Area of the Screen

Several operations on the 3270 emulator, such as storing an area of the screen into a variable or performing an `assert` operation, require you to highlight an area of the screen. There are two selection modes that you can use to highlight an area of the screen:

Flow Selection Mode. Using this mode, all 3270 characters in the display buffer are highlighted from the chosen start location to the chosen end location.

Block Selection Mode. Using this mode, a block or rectangle of characters are highlighted where the chosen start and end locations define opposite corners of the block.

▼ To Highlight an Area of the Screen

1. **Choose a selection mode from the appropriate menu item or use the appropriate toolbar button on the Navigation Recorder window.**
2. **Place the mouse at the start of the area to highlight. Click and hold down the (left) mouse button.**

   This activates the recorder functions that operate on a highlighted area.

3. **Drag the mouse to the end of the area to highlight.**

   An outline is displayed showing the highlighted area.

4. **Release the mouse button.**

   The background color of the characters changes to indicate the selection.

   Highlighting an area does not generate any code in the Code Viewer window.

Storing an Area of the Screen

You can store an area of the 3270 emulator screen in a variable, but first you must highlight the area of the screen that you want to store. See "Highlighting an Area of the Screen" on page 51.

▼ To Store the Highlighted Area of the Screen

1. **Select the Store Highlighted Area option from the Actions menu or use the toolbar button from the Navigation Recorder window.**

2. **In the dialog box, choose a variable in which to store the highlighted area.**

   Depending on the selection mode you used to select the highlighted area, you must choose a specific type of variable:

   If you are in Flow Selection mode, you must choose a String variable. The contents of the variable are replaced with the contents of the highlighted area.

   If you are in Block Selection mode, you must choose a Vector variable. The contents of the variable are replaced with the contents of the highlighted area. Each row of the block selection becomes an element of the Vector variable.

   When DBCS data from a shift-out shift-in (SOSI) field is stored, any SO and SI characters present in the highlighted area are not stored in the selected variable because they are not required in a Unicode string. This allows you to reenter an area of the screen, which was stored in a variable this way, using the Type a variable control at a later time.

Creating an `assert`

After you highlight an area of the screen, you can perform an `assert` operation for that area.

- **Select the Assert Highlighted Area option of the Actions menu or use the corresponding toolbar button from the Navigation Recorder window.**

  This generates a line of code in the Code Viewer window:

```
assert("ENTER COMMAND",261);
```

  When this code is executed, it confirms (asserts) that these characters are present at the specified offset of the 3270 emulator display buffer. If this is not the case, the `assert()` method generated in the output class throws a Java `IllegalStateException`.

  This mechanism enables a Sun 3270 Pathway Bean to confirm that it is on the expected screen of an application and to notify its caller if this is not true.

  SO or SI characters within an area being asserted are not placed in the `assert` string because they are not required.

Starting the Store List Wizard

The Store List Wizard enables you to create Java code to save related data from multiple 3270 screens into a single variable, for example, a list. The Store List Wizard saves you the effort of traversing each screen of the list and copying the data manually. See Chapter 9 for information about using the Store List Wizard.

▼ To Change the Font

- **Use either method:**
  - Select Settings → Font on the Navigation Recorder window, and select a font style that you prefer.
  - Follow these steps:

  a. Move the cursor over the 3270 emulator.

  b. Click the right mouse button to display the pop-up menu.

  c. Select Font, and select a font style that you prefer.

Namespace Viewer Window

The Namespace Viewer window (FIGURE 7-7) enables you to view and manipulate variables. The primary purpose of the variables is to act as input and output areas for data supplied to and from the generated Java class at execution time.

Any variables defined to the recorder in the Namespace Viewer window also are displayed as instance variables in the generated Java code, which you can see in the Code Viewer window (FIGURE 7-8). Also, each variable can have a pair of `get()` and `set()` accessor methods. For example, for a variable called `var1`, a pair of methods is created by default in your navigation class called `getVar1` and `setVar1`.

Namespace Viewer Window Contents

On the Namespace Viewer window menu you can choose the type of variables to view:

- Both String and Vector variables
- String variables only
- Vector variables only

The Namespace Viewer window contains a list of the variables of the chosen type with information about each.

TABLE 8-1 Variable Type Information

Variable Information	Description
Type	S or V. Vector variables that currently contain elements also have a + symbol on the type. By clicking this symbol you can view the current values of each element of the vector.
Name	Name of the variable.
Default	Default value of all Vector variables is shown as `{empty}`. You can alter the default value of a String variable by double-clicking the left mouse button in this area.
Current	Current value of the variable.
Get Method	If the check box is checked, a `get()` method is generated for the variable.

TABLE 8-1 Variable Type Information *(Continued)*

Variable Information	Description
Set Method	If the checkbox is checked, a `set()` method is generated for the variable.
In Use	If this checkbox is checked, the variable is being used and you cannot delete it. You cannot alter the value of this check box.
Description	Description of the variable. You can change the description by double-clicking the left mouse button in this area.

Variable Types

There are two variable types in the Namespace Viewer window:

String. A Java String object. String variables are typically used to contain either input or output parameters for the navigation class.

Vector. A Java Vector object where each element of the vector is a Java String. Vector variables are only used as output parameters. For example, they can be used to contain the result of storing a Block Selection or to contain a list generated by the Store List Wizard.

Creating and Deleting Variables

This section describes how to create and delete variables.

Note – You can only delete a variable when it is not in use. There is no generated code other than the `get()` and `set()` methods that access the variable.

▼ To Create a Variable

1. **Select the Create option from the Actions menu of the Namespace Viewer window or use the corresponding toolbar button.**
2. **On the Create a variable window (FIGURE 7-10), supply the following information:**
   a. **In the Variable Name field, type the name using the Java convention of starting variable names with a lowercase letter.**

      For example, for a variable named `var1`, the accessor methods generated are called `getVar1` and `setVar1` (with an uppercase V). This follows the standards of the Java Bean naming pattern conventions. The recorder attempts to prevent you from typing any characters that are not valid in a Java variable name.

b. In the Description field, type a brief description of the variable.

When your Sun 3270 Pathway class is saved as a Java Bean, this description is stored in the generated `BeanInfo` file (see "To Save Navigation Classes to the Disk" on page 47). An IDE makes this information available to a developer using the generated Sun 3270 Pathway Bean.

c. In the Variable Type field, select either String or Vector, depending on the type of variable that you want to create.

d. In the Default/Current Value field, type an appropriate value.

This field applies to String variables only. Because Vector variables are used only as output variables, there is no mechanism to set a default or current value.

The Default value is the value to which the variable is initialized in the generated class. For example, a String variable called `var1` with a default value of `fred` generates the line:

```
private String var1 = "fred";
```

The Current value defines the initial value of the variable for the purposes of the recording session only. This value does not appear in the generated code. For example, this allows a recorder user to define a variable called password, set the current value of password, and successfully navigate through a 3270 application that requires that password.

▼ To Delete a Variable

1. **Select the variable in the Namespace Viewer window.**
2. **Select the Actions → Delete option or press the Delete button on the toolbar of the Namespace Viewer window.**

   A confirmation dialog box is displayed.
3. **Click OK to delete the variable.**

Code Viewer Window

The Code Viewer window (FIGURE 7-8) dynamically displays the code generated by the recorder. Using the Code Viewer window you can view either the main class, which is generated by recorder, or the `BeanInfo` class ("Structure of the `BeanInfo` Class" on page 59) that corresponds to the main class. The code visible in these two classes is what is saved to disk when you select the Save option from the File menu on the main recorder window. See "To Save Navigation Classes to the Disk" on page 47.

Structure of the Main Navigation Class

When recording is on, interaction with the 3270 emulator in the Navigation Recorder window immediately generates code in the Code Viewer window. This section describes the important portions of the Sun 3270 Pathway-generated main class.

Instance Variables

Several private instance variables are defined in your main Sun 3270 Pathway navigation class as:

```
private Terminal term;
```

This is the instance of Sun 3270 Pathway Bean that this navigation class manipulates. This variable has no default value and is initialized to `null`.

The following line of code identifies the default TN3270 host name to which this navigation class connects, unless otherwise instructed:

```
private String tn3270Host = "locis.loc.gov";
```

The following line of code identifies the name of the default TN3270 port to which this navigation class connects, unless otherwise instructed:

```
private int tn3270Port = 23;
```

Each of these instance variables also has a corresponding `get()` and `set()` method pair for access.

`init()` Method

The `init()` method is a private method that is called at the beginning of the `performWork()` method. It is the job of the `init()` method to ensure that the instance variable `term` is a valid reference to a connected Terminal before returning control to the `performWork()` method.

The `init()` method uses the following logic:

1. Ensure that the `term` instance variable is not null. If `term` is null, create a new Terminal and set its host and port to be equal to the `tn3270Host` and `tn3270Port` instance variables.
2. When `term` is in the process of disconnecting, allow it to finish.
3. If `term` is disconnected, issue a connect request.
4. Wait for `term` to become connected.
5. Wait for the keyboard to become unlocked.

This enables a user to call a Sun 3270 Pathway navigation class in several ways. In the following examples, `NavClass1` and `NavClass2` are separate navigation classes.

Example 1: Create a navigation class that creates and uses its own private Terminal, connecting it to the default host and port:

```
NavClass1 nav1 = new NavClass1();
nav1.performWork();
```

Example 2: Create a navigation class that creates and uses its own private Terminal, connecting it to host `myhost` and port `9993`:

```
NavClass1 nav1 = new NavClass1();
nav1.setTN3270Host("myhost");
nav1.setTN3270Port(9993);
nav1.performWork();
```

Supply the first navigation class with a predefined Terminal, then call that navigation class. Next, invoke a second navigation class in the same way. The Terminal `term` can remain connected between the calls to the two navigation classes, enabling the classes to be chained together sequentially.

```
Terminal term = new Terminal();
term.setTN3270Host("myhost");
term.setTN3270Port(9993);
term.connect();
NavClass1 nav1 = new NavClass1();
nav1.setTerminal(term);
nav1.performWork();
NavClass2 nav2 = new NavClass2();
nav2.setTerminal(term);
nav2.performWork();
```

`performWork()` Method

A user of the main class should call the `performWork()` method to have it perform its navigational work. The `performWork()` method always starts with a call to the `init()` method.

Following the call to `init()`, there are lines of code that represent the recorded actions. These lines are grouped together and are referred to as interactions. Each interaction represents an instance of a user interacting with the 3270 emulator, then

pressing an AID key (for example, Enter, Clear, PF, PA), which causes the keyboard to become locked pending a response from the 3270 host. The interaction typically ends with a line of code that waits for the host to respond.

Example: If the user types `ACCT` and presses Enter, the generated interaction is:

```
term.typeString("ACCT");
term.pressEnter();
term.waitUntilkeyboardUnlocked();
```

The code is grouped into these interactions for clarity and readability and to allow the most recent interaction to be deleted by selecting the Delete option from the Actions menu of the Code Viewer window.

The most recent interaction is highlighted in a different color, drawing attention to the lines of code being generated.

By default, the recorder attempts to position the generated code in the Code Viewer window so that the most recent interaction is kept visible. The generated code consists primarily of calls to the Terminal bean that this navigation class will use.

Structure of the `BeanInfo` Class

A `BeanInfo` class is a Java class whose sole purpose is to describe a Java Bean. Because the recorder generates code that can be a Java Bean, it also generates a `BeanInfo` class.

`BeanInfo` classes are a standard part of Java Bean definition. The `BeanInfo` class generated by the recorder is an implementation that provides a full description of the Bean as created by the recorder.

All the instance variables defined in the generated navigation class are declared as properties in the `BeanInfo` class. Depending on whether there are `get()` and `set()` methods for the variables, normal, read-only, or write-only properties are generated.

Most recorder users do not need to understand the structure of this class.

Deleting the Most Recent Interaction

When using the recorder, you can delete the most recent interaction.

Caution – Deleting interactions may make your generated class inconsistent with the 3270 application that you were recording. Therefore, use this function with care. It is your responsibility to ensure that the generated class is correct if you choose to delete interactions.

▼ To Delete the Most Recent Interaction

1. **Verify the interaction is currently highlighted in red.**
2. **In the Code Viewer window, select Actions → Delete from the menu.**

   A confirmation dialog box is displayed.
3. **Click OK to delete the interaction.**

Code Viewer Window Settings

You can manipulate the behavior of the Code Viewer window using the following options on the Settings menu:

- The Font option to specify the font in which the Window is displayed. See "To Change the Font" on page 53.
- The Follow Main Code option to have the Code Viewer window always display the most recent interaction.
- The Code generation option to influence how the code is generated.

The Code generation option displays a dialog that allows you to set the following:

- The Java class and package names of the generated class.
- The number of spaces to use to prefix indented lines of code (indent size).
- Whether to display opening curly braces ({) on a new line or at the end of the previous line.
- How to handle cursor movement, for example, by a call to move the cursor to an absolute buffer offset, or by one or more calls to `term.cursorRight()`, `term.cursorDown()`.
- Whether Javadoc comments are generated for the class.
- How to handle waiting. See "Wait Strategies" on page 61.

Wait Strategies

This section describes the differences between a well-behaved and poorly-behaved 3270 application, and the waiting strategies used by the recorder.

A well-behaved 3270 application sends a screen of information to a user on a 3270 terminal as follows:

1. One or more datastreams are sent from the host to the terminal to build the screen that is presented to the user.
2. The last datastream sent to the terminal instructs the terminal to unlock the keyboard, allowing the user to interact with the terminal.

A poorly-behaved 3270 application sends a screen of information to a user on a 3270 terminal as follows:

1. One or more datastreams are sent from the host to the terminal to build the screen that is presented to the user.
2. A datastream *prior* to the last datastream sent to the terminal unlocks the keyboard.

In this case, it is usually only the speed with which the subsequent data streams are sent after the keyboard is unlocked that prevents the user from interacting with the terminal before the last data stream arrives. An application behaving in this manner is in error because a user should not be allowed to interact with a terminal until the host has finished sending data to that terminal.

Before using the recorder, correct any 3270 applications that are poorly-behaved. If you fail to do this, the recorder uses its wait strategies to cope with poorly-behaved applications.

The strategies used to insert `wait` commands into the generated code are based on two methods available in the `Terminal` class. You can find a complete description of these methods in the Javadoc for the Pathway Terminal.

The method typically used with a well-behaved screen is:

```
waitUntilKeyboardUnlocked()
```

The method typically used with a poorly-behaved screen is:

```
waitHeuristic(int t)
```

The recorder has three strategies for inserting `wait` commands into the generated code. The following table describes each.

TABLE 8-2 Wait Strategies

Wait Strategy	Description
Intelligent waiting	Inserts calls to the `waitUntilKeyboardUnlocked()` method when the recorder observes that the screen is well-behaved. This is the default strategy. If the recorder detects that a screen is poorly-behaved, a call to `waitHeuristic(int t)` is generated instead. The interval used as the parameter *t* for `waitHeuristic()` is determined by observing the data streams received at recording time, and making a reasonable guess at an interval based on 2 x maximum observed interval between sends for the screen after the keyboard is freed.
Intelligent waiting with override	Similar to intelligent waiting. The only difference is that instead of using a reasonable guess for the `waitHeuristic()` method, a fixed configurable value is used. You can modify the value using the Code Generation dialog. This is a useful strategy if the application is less well-behaved at deployment time than it was at recording time.
Fixed heuristic waiting	Generates `waitHeuristic()` calls for all screens. The time parameter for this call is a fixed configurable value. You can modify the value using the Code Generation dialog. This is a useful strategy if the application is as well-behaved at deployment time as it was at recording time.

Normally, the default strategy is adequate. However, you may require the other strategies for some types of application. You can change the strategy using the Code Generation dialog.

Note – Using the `waitHeuristic()` method has an associated performance penalty because, for each interaction with the host, an extra delay of the time period *t* is incurred.

CHAPTER 9

Store List Wizard

A wizard is an application that makes it easier to perform a complicated task. The Store List Wizard enables you to create Java code that saves related data from multiple 3270 screens into a single variable. The Store List Wizard saves you from having to access each screen and manually copy the data into a list.

The length of a list can vary depending on runtime factors. Because it is not possible to guarantee the length of the list at runtime, a function that can traverse a list and determine when the end of the list has been reached is needed. The Store List Wizard provides this functionality.

This chapter contains the following topics:

How the Store List Wizard Works

When you start the Store List Wizard, a dialog guides you through a series of pages. Typically, the wizard requests some information on each page. You supply the required information and click the Next button to go to the next page of the wizard.

The Store List Wizard requires answers to the following questions:

- Where are you going to store the data?
- Where does the data appear on the 3270 emulator?
- How is the next page of data accessed?
- What determines the end of the data?

After the wizard has obtained enough information, it can create Java source code to perform the list retrieval. This source code is displayed in the Code Viewer window of the Navigation Recorder.

Using the Store List Wizard

▼ To Start the Store List Wizard

1. **Make sure the 3270 Pathway Recorder is in recording mode.**
2. **Make sure you are on the first page of your list.**

   If not, close the Store List Wizard and in the 3270 emulator screen navigate to the start of the list.
3. **Use either method to start the wizard:**
   - On the Navigation Recorder window, select Actions → Store List Wizard.
   - Click the Store List Wizard toolbar button on the Navigation Recorder window.

   When you are recording the traversal of a list, it is a good idea to use a list that spans more than one 3270 emulator screen. The Store List Wizard needs to record the key strokes necessary to make the host application proceed to its next screen.

The Store List Wizard Interface

The Store List Wizard is displayed as a dialog within the 3270 Pathway Recorder. FIGURE 9-1 is an example of the basic structure, which is a page at the top of the window with a set of controls beneath it.

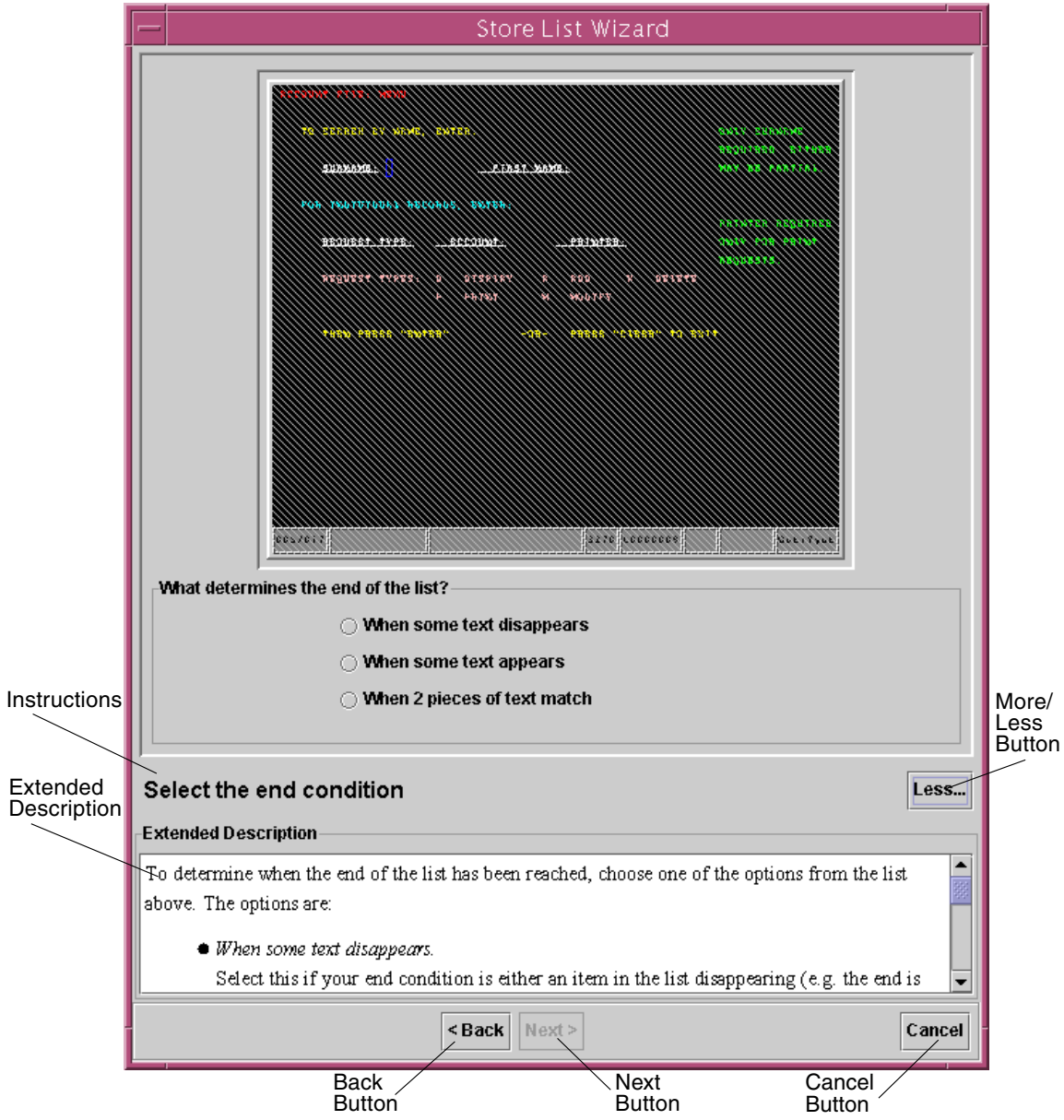

FIGURE 9-1 Store List Wizard Window

The labels in FIGURE 9-1 identify the following items on the Store List Wizard window:

Label	Store List Wizard Screen Area
Instructions	Indicates the instruction for this page.
More/Less Button	When More is clicked, the Extended Description is displayed. When Less is clicked, the Extended Description is removed from view. Clicking toggles between states.
Extended Description	Provides instructions on how to complete this page of the wizard.
Back Button	Returns to the previous page of the Store List Wizard. This enables you to check or correct information.
Next Button	Is active when the wizard has obtained enough information for that page. The Next button is replaced by the Finish button on the last page of the wizard. When the Finish button is clicked, the Store List Wizard closes and the Java source code is generated.
Cancel Button	Exits the Store List Wizard without generating any code.

Store List Wizard Usage Example

This section contains an example of a route through the Store List Wizard, which uses the Sun MTP transaction `CTBL`. If you have the Sun MTP software installed, you can follow this example by displaying the Program Control Table (PCT) before starting the Store List Wizard.

1. **Start the Sun MTP region and make a connection.**
2. **Type the `CTBL` transaction to display the Table Manager menu.**
3. **Press PF4 to display the Standard Tables menu.**
4. **Press PF6 to display the PCT.**
5. **Start the Store List Wizard as described in "Starting the Store List Wizard" on page 53.**

   When the Store List Wizard starts, a Welcome screen is displayed, which gives some general information about the Store List Wizard. No information is required from the user at this point.
6. **Click the Next button to move to the next page.**

Using the Select the Storage Vector Page

The Select the storage vector page is used to define the Vector variable in which to store the data. The page has two areas, which are identified with labels in FIGURE 9-2.

Label	Screen Area Description
List	Area in the middle of the screen that lists all the Vector variables that were created in this navigation. If none were created, the list is empty.
Create Button	Creates a new Vector variable.

▼ To Select a Storage Vector

- **Select the correct Vector variable in the list and click Next.**

FIGURE 9-2 Select the Storage Vector

Using the Select the Area of the List Page

The list area is defined as the area on the 3270 emulator screen where the data for the list appears. When the Store List Wizard navigates to a new 3270 emulator screen, the data defined by the list area is stored in the storage vector. Each row of the list area is stored as a separate element within the storage vector.

▼ To Select an Area of the List

1. **Use the mouse to highlight the list area on the 3270 emulator.**

   When a selection is made, the Selected Area portion of the window indicates the area of the list that was highlighted, and the Next button becomes active (see FIGURE 9-3).

2. **When you are satisfied that you have selected the correct area click Next.**

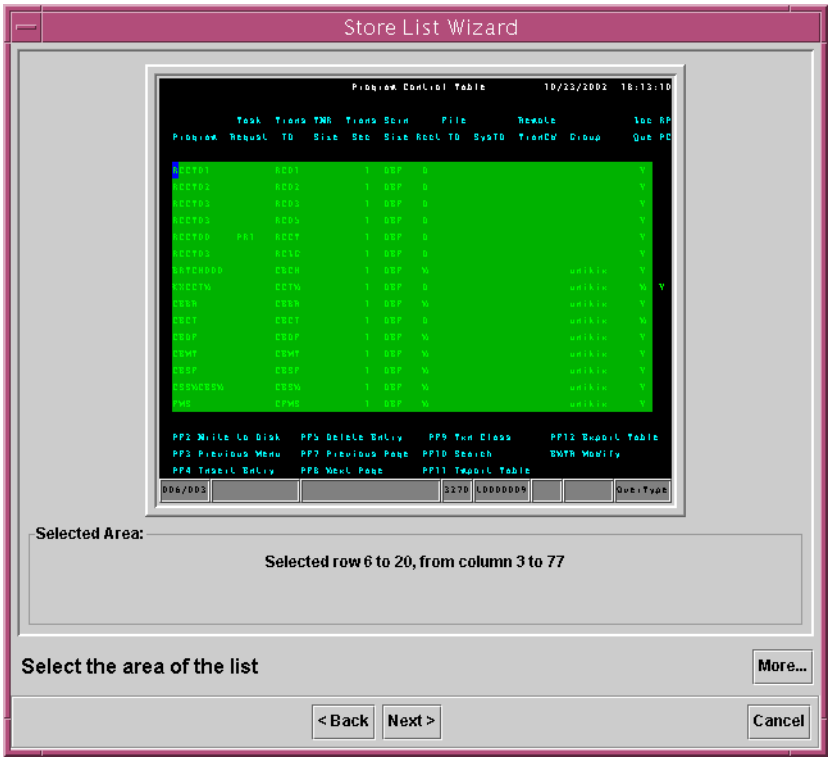

FIGURE 9-3 Select the Area of the List

Using the Select the End Condition Page

There are three conditions you can use to determine the end of the list:

- When some text disappears

  Use this condition when a message is displayed on the screen informing you that the end of the list has not been reached, or when the next item in the list is different from the previous item.

- When some text appears

  Use this condition when a message is displayed on the screen informing you that the end of the list has been reached, or when the next item in the list matches a specific piece of text.

- When two pieces of text match

  Use this condition for lists that display the current page number of the list and a page total, for example, Page 1 of 5.

FIGURE 9-4 shows a screen with the `When some text appears` option selected.

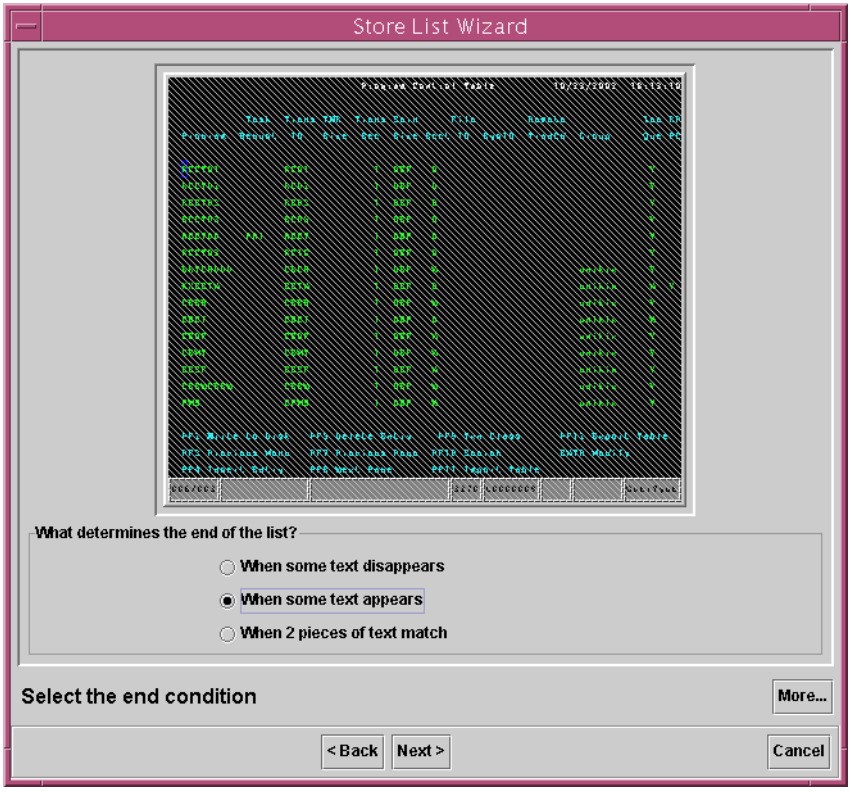

FIGURE 9-4 Select an End Condition

Using the Record the Advance Actions Page

You must record the page advance actions, even if you are already at the end of the list, so that the wizard knows how to advance at runtime if the list is more than one page.

On the 3270 emulator, record the actions required to advance the list by one screen. These actions are displayed in the Advance actions list below the 3270 emulator screen. After you press an AID key, the 3270 emulator does not accept any keyboard input. If you need to correct a mistake, click the Clear Recorded Actions button.

If your key strokes are not being recorded, ensure that the 3270 emulator screen has focus by clicking it with the mouse and trying your commands again.

Press the PF8 key to display the results shown in FIGURE 9-5.

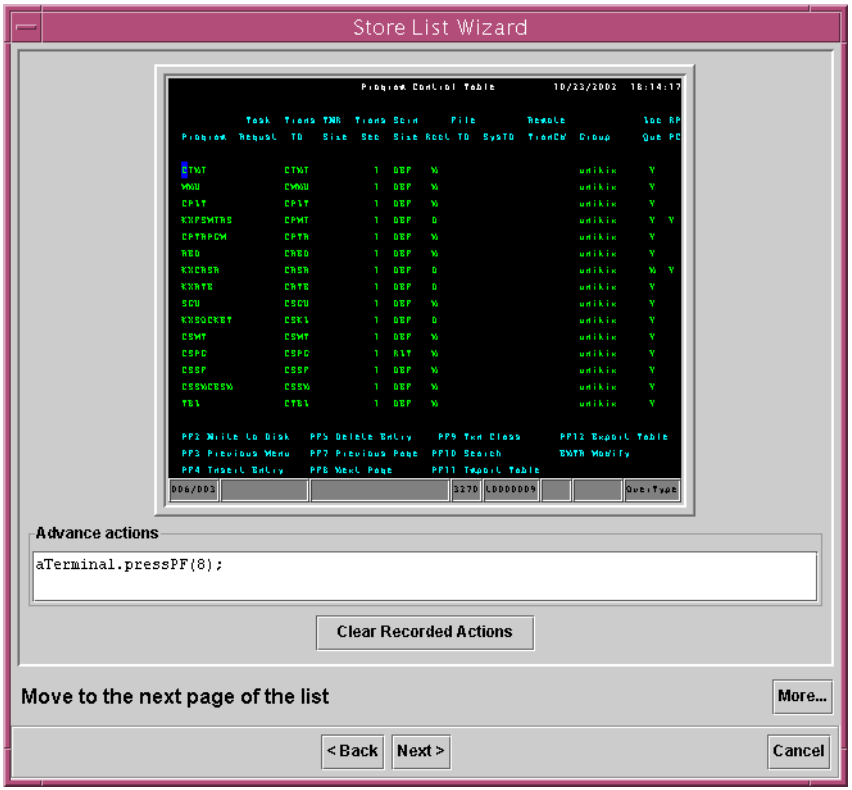

FIGURE 9-5 Move to the Next Page of the List

Using the Manual Move to the End of the List

Move the 3270 emulator to the end of the list by using the Advance 1 Page button or by directly manipulating the 3270 emulator. See FIGURE 9-6.

The Advance 1 Page button repeats the actions performed on the Record the advance actions page. You should use the Advance 1 Page button so that when you finish with the Store List Wizard, the storage vector contains the list data. No matter which method you use to arrive at the end of the list, the runtime storage of the list into the storage vector is not affected.

When you reach the screen that identifies the end of the list, advance to the next page of the Store List Wizard by clicking Next.

For this example, make sure that you continue to advance forward until the end of the list message is displayed.

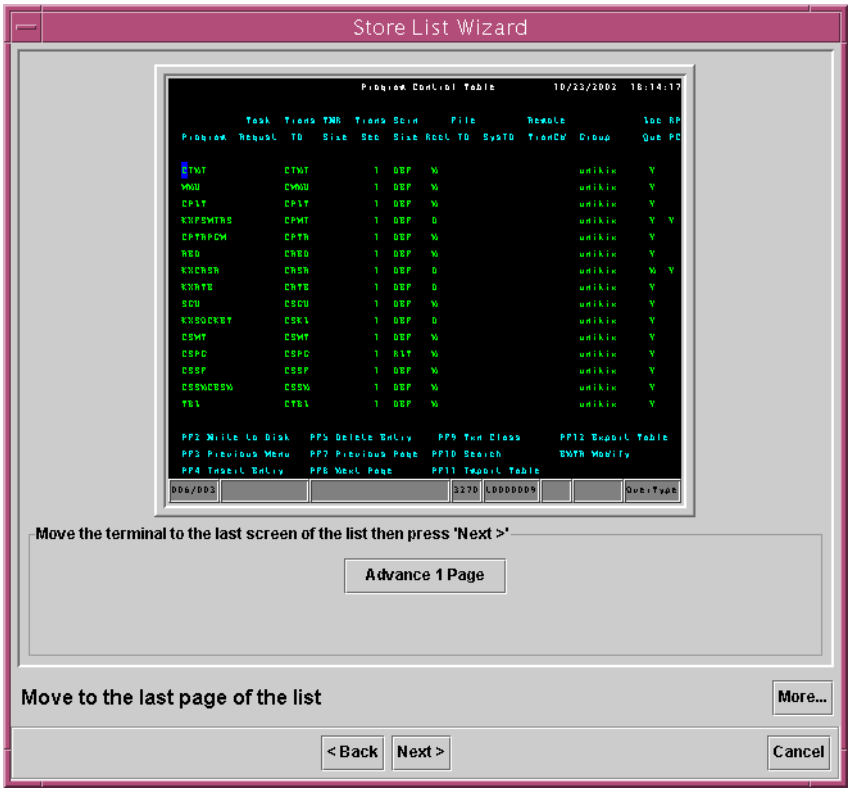

FIGURE 9-6 Move to the Last Page of the List

Using the Select the Text That has Appeared Page

Specify which text is to be displayed on the 3270 emulator when the end of the list is reached by highlighting the selection on the 3270 emulator using the mouse.

Select the position of the text using one of the following options:

- Always at fixed location

  When the 3270 emulator screen has the selected text at this position, the end of the list has been encountered.

- Anywhere in data area

  To activate this option, the selected text must be wholly contained within the list area and cannot span more than one 3270 emulator row. If the selected text is found at the appropriate column on any row within the data area, the end of the list has been encountered.

Certain applications only provide information stating that the end of the list has been reached after an attempt is made to move past the end of the list. In these cases, do not store the final screen, because it is a retransmission of the last page of the list. To ensure that the wizard stores the correct cases for these applications, make sure that the Record data from this emulator screen check box is *not* checked.

When you click Finish, the Store List Wizard closes and generates code in the Code Viewer window. For this example, ensure that the check box is not checked as shown in FIGURE 9-7.

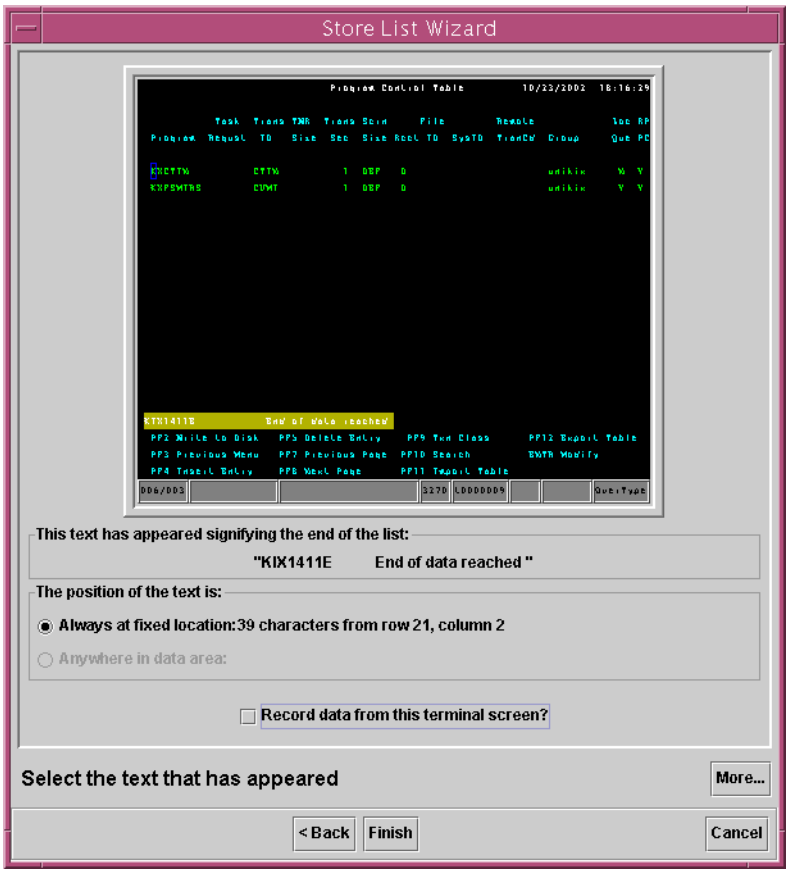

FIGURE 9-7 Select the Text That has Appeared

This concludes a typical walkthrough for the Store List Wizard. There are three other screens in the Store List Wizard that are described in the following sections.

Using the Select the Text That Will Disappear Page

To make text disappear when the end of the list is reached, you must select the position of the disappearing text using one of the following options:

- Always at fixed location

  When the 3270 emulator screen does not have the selected text at this position, the end of the list has been encountered.

- Anywhere in data area

  To activate this option, the selected text must be wholly contained within the list area and cannot span more than one 3270 emulator row. If the selected text is found at the appropriate column on any row within the data area, the end of the list has been encountered.

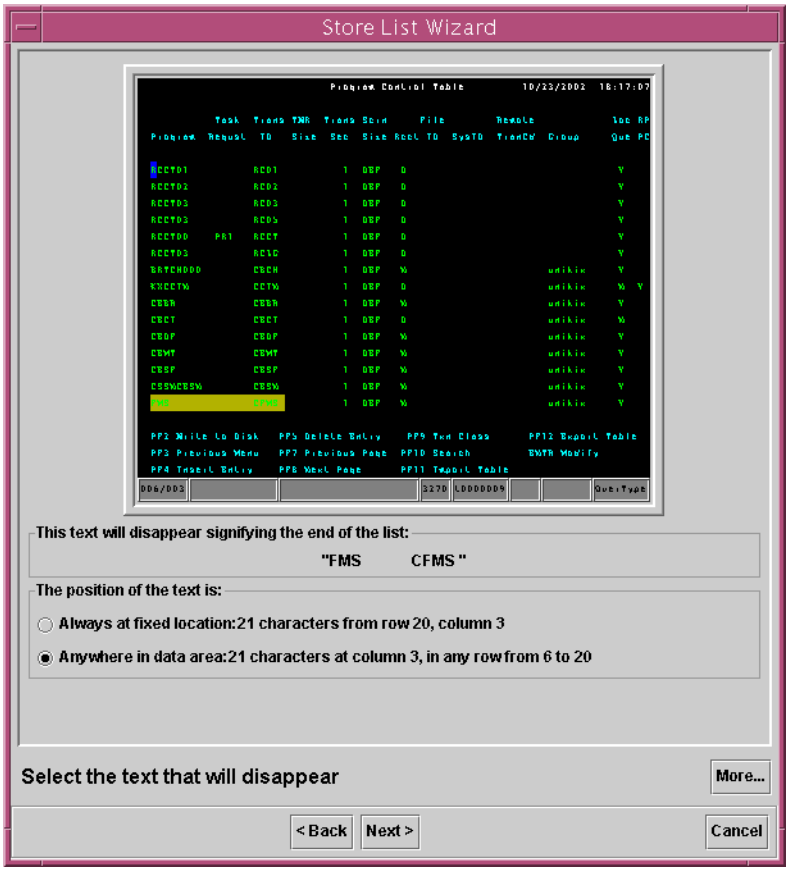

FIGURE 9-8 Select the Text That Will Disappear

Using the Move to the End of the List Page

Move the 3270 emulator to the end of the list by clicking the Advance 1 Page or the Auto Advance to End button. If the 3270 emulator is at the end of the list already, these two buttons are disabled.

The Auto Advance to End button attempts to take you automatically to the end of the list. If it succeeds, the button becomes deselected and the Finish button becomes active. However, if it seems that the list is not going to reach the end, click the button again and use the Back button to return to previous pages to check that you have correctly defined the scope of the list.

Certain applications only provide information stating that the end of the list has been reached after an attempt is made to move past the end of the list. In these cases, it is not desirable to store the final screen because it is a retransmission of the last page of the list. To ensure that the wizard stores the correct cases for these applications, make sure that the Record data from this emulator screen check box is not checked. See FIGURE 9-9.

When you click Finish, the Store List Wizard closes and generates code in the Code Viewer window.

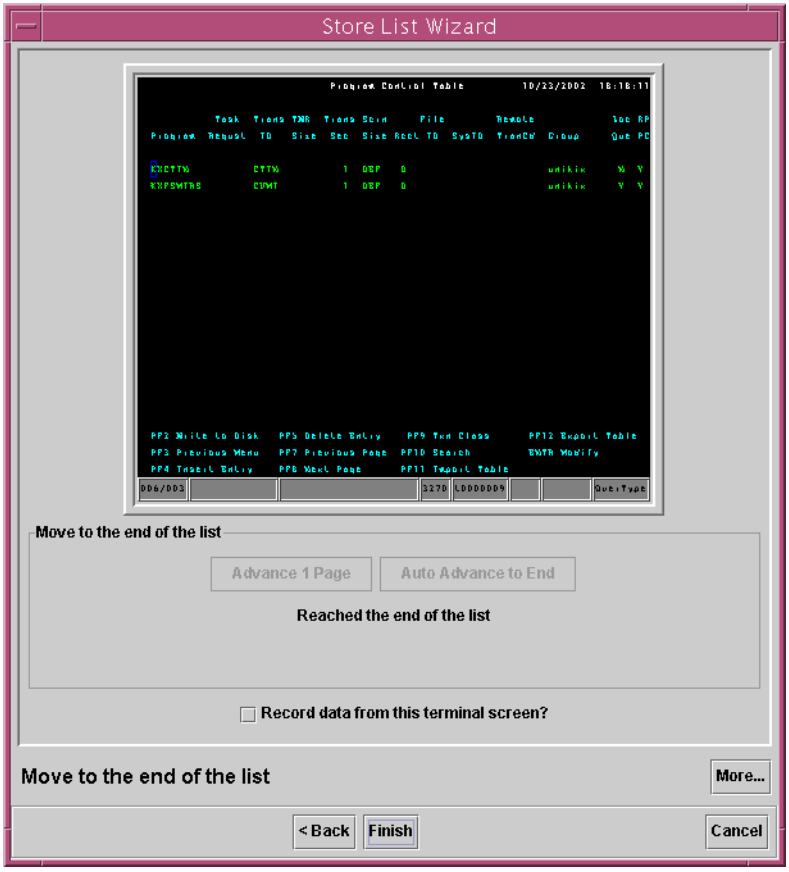

FIGURE 9-9 Move to the End of the List

Using the Select Matching Text Areas Page

Certain applications provide a message informing the user which page of the list is currently displayed and how many pages the list spans. With these applications, you know when you are on the last page because the two values match. To define this to the Store List Wizard, you must select two areas on the 3270 emulator screen as shown in the following figure.

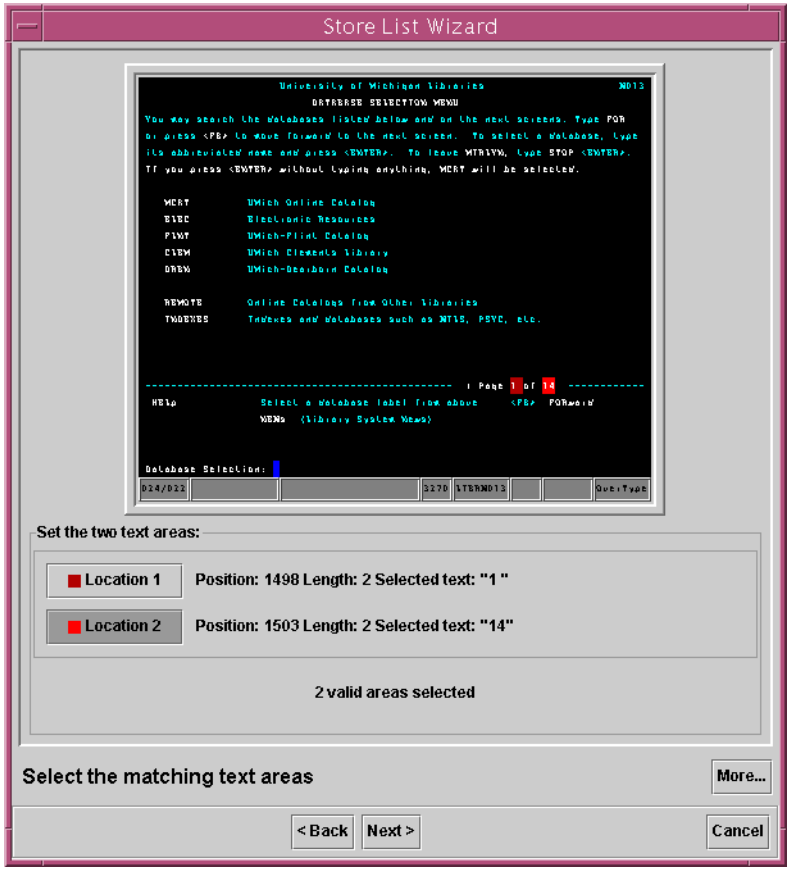

FIGURE 9-10 Select the Matching Text Areas

▼ To Select Matching Text Areas

1. **Click either the Location 1 or Location 2 button.**
2. **Select the area on the 3270 emulator screen.**
   The highlighted area matches the respective location color.
3. **Click the other location button.**
4. **Select the second area on the 3270 emulator screen.**
   The two text areas must be the same length and must not overlap.

CHAPTER 10

TCP Routing Program

This chapter describes the configuration, execution, and use of the TCP Routing program. It contains the following topics:

The Java security model prevents any applet from connecting to any host except the one from which it was originally downloaded. This means that if a Java applet must communicate to a TN3270 server, the TN3270 server must be located on the same host as the web server. To overcome this limitation, the TCP Routing program (router) can run on the web server system to route client requests to the destination TN3270 server. FIGURE 10-1 illustrates a configuration in which the TCP Routing program is used as an intermediary.

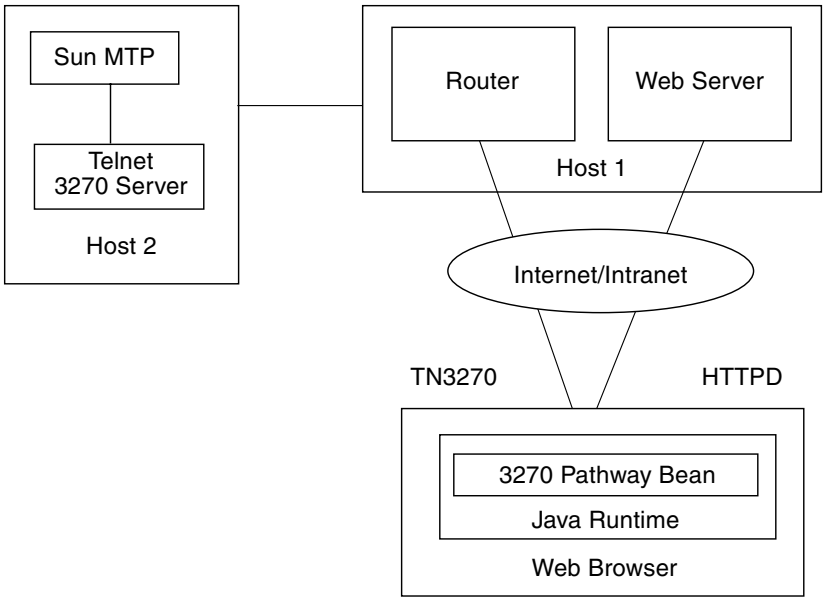

FIGURE 10-1 TCP Router Configuration—Example

Starting the TCP Router

The procedures in this section describe how to start the router on various platforms.

Starting the Router on Solaris and UNIX Systems

On Solaris or UNIX systems, there are two ways to start the router.

▼ To Start the Router Using the Shell Script

1. **Ensure that the required level of JRE is on your path.**

   See "Obtaining a Java Runtime Environment (JRE)" on page 3.

2. **Run the `pathway_router.sh` shell script located in the `bin` directory of the product installation.**

▼ To Start the Router Using the JAR File

1. **Ensure that the required level of JRE is on your path.**

   See "Obtaining a Java Runtime Environment (JRE)" on page 3.

2. **Run the executable JAR file located in the `lib` directory of the product installation.**

   Change to the `lib` directory and type the following command:

```
$ java -jar pathway_router.jar
```

Starting the Router on Microsoft Windows Systems

There are three ways to start the router.

▼ To Start the Router Using the Start Menu

1. **On the desktop, click the Start button.**
2. **Select Start → Programs → Sun 3270 Pathway 2.0.0 → Router.**

▼ To Start the Router Using the Router Executable

- **Run the `pathway_router.exe` program located in the `bin` directory of the product installation.**

▼ To Start the Router Using the JAR File

1. **Ensure that the required level of JRE is on your path.**

   See "Obtaining a Java Runtime Environment (JRE)" on page 3.

2. **Run the executable JAR file located in the `lib` directory of the product installation.**

   Change to the `lib` directory and type the following command:

```
C:\> java -jar pathway_router.jar
```

Command Line Parameters

You can invoke the TCP Router with a series of parameters on startup.

Format:

`pathway_router [-h` *Target-Host*`] [-p` *Target-Port*`] [-l` *Local-Port*`] [-g] [-t]`

where:

`-h` *Target-Host*	Name of the machine on which the TN3270 server and the Sun MTP region are executing.
`-p` *Target-Port*	Port number at which the TN3270 server is listening. The default is 2001.
`-l` *Local-Port*	Port number at which the router is to listen on the local machine. The default is 2001.
`-g`	Starts the router with the graphical user interface (GUI). See "Graphical User Interface (GUI)" on page 82.
`-t`	Starts the router with all connections having diagnostic tracing in the active state. See "Diagnostic Tracing" on page 84.

Graphical User Interface (GUI)

The router can be started with or without a graphical user interface (GUI). By default, the router starts with a GUI.

To start the router without a GUI, you must specify as a command-line parameter, the target host to connect to. Optionally, you can specify the target port and local port. If you do not specify these, the router uses the default values.

If the router is started without a GUI, it automatically starts routing to the specified target host. If you want the router to automatically start routing to a particular host *and* want a graphical display, you must specify the `-g` option as a command-line parameter.

The GUI enables you to see the current number of connections going through the router. It also enables you to enable or disable diagnostic tracing at any time.

Note – Starting the router with the GUI enabled adds some processor overhead.

▼ To Connect to a Host Using the GUI

1. **Type the following command to start the router with the GUI:**

```
$ java -jar pathway_router.jar
```

The connection dialog shown in FIGURE 10-2 is displayed.

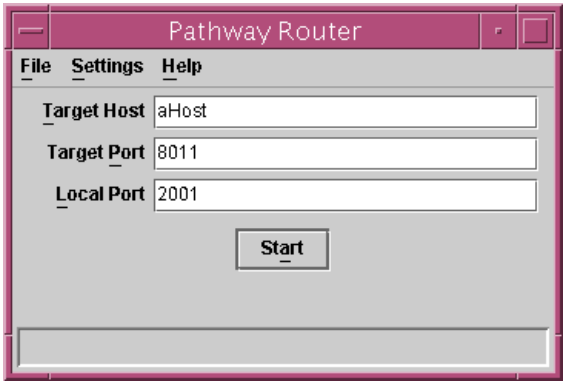

FIGURE 10-2 GUI Router Connection Dialog—Example

2. **Type values in the following fields: Target Host, Target Port, and Local Port.**
3. **Click the Start button, which attempts to connect to the specified host and disables the input fields as shown in the following figure.**

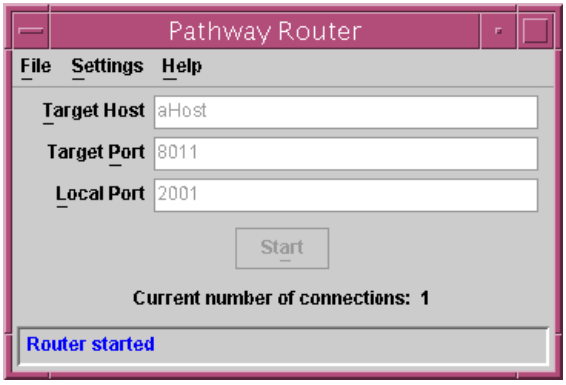

FIGURE 10-3 GUI Router Connected—Example

The current number of connections is shown below the Start button.

Menu Bar

The following table lists the menu bar items on the GUI connection dialog.

Menu Item	Function
File	Contains an Exit option that displays the shutdown window.
Settings • Trace New Items	Check box toggles a setting so that any new connections have diagnostic tracing set to on or off. See "Diagnostic Tracing" on page 84.
Help	Contains an About option that displays router product information.

Diagnostic Tracing

A diagnostic trace stores the data sent to and received from the target host. The data is stored in trace files in the current working directory on your disk. There is a file for each connection through the router and it contains both the data sent to and data sent from the target host. You can review this information to determine if the data was what you were expecting.

Note – Do not start diagnostic tracing unless you must review the data or are asked to obtain a diagnostic trace by your authorized Sun service provider.

When you use a diagnostic trace, be aware that:

- Trace files can quickly use large amounts of disk space. Start tracing only when you want to generate trace files.
- Generating diagnostic traces slightly degrades the performance of running systems.

Structure of a Diagnostic Trace File Name

The following table shows the structure of a diagnostic trace file name.

TABLE 10-1 Diagnostic Trace File Name Structure

Name Element	Example
The letters `pathway_router` followed by an underscore.	`pathway_router_`
The client host name, followed by an underscore.	`starlight_`
The client port number, followed by an underscore. Note that the client port is not the same as the local port.	`2001_`
The current millisecond time value.	`1035199509022`
The file name extension `.trace.txt`	`pathway_router_starlight_2001_1035199509022.trace.txt`

Contents of the Diagnostic Trace File

When a trace is started, the start time and date of that trace is written to the trace file. When there is a transfer of data, the following information is written to the file:

- The total number of bytes exchanged in the data transfer
- The hexadecimal, EBCDIC, and ASCII representations of the data transferred
- The stop time for the trace

The following figure shows an example of a trace file.

```
Start time => 2000/03/18-16:36:18
Number of bytes = 3 (0x3)
0000: fffc28                                [..(             ] [...             ]
Number of bytes = 3 (0x3)
0000: fffb18                                [...             ] [...             ]
Number of bytes = 18 (0x12)
0000: fffa1800 49424d2d 33323738 2d322d45 [....IBM-3278-2-E] [......(.........]
0010: fff0                                  [..              ] [.0              ]
Number of bytes = 3 (0x3)
0000: fffb00                                [...             ] [...             ]
Number of bytes = 3 (0x3)
0000: fffd00                                [...             ] [...             ]
Number of bytes = 3 (0x3)
0000: fffb19                                [...             ] [...             ]
Number of bytes = 3 (0x3)
0000: fffd19                                [...             ] [...             ]
Stop time  => 2000/03/18-16:36:25
```

CHAPTER 11

The Pathway Terminal

This chapter describes the Pathway Terminal, a 3270 emulator. It contains the following topics:

Note – For each Sun MTP region you wish to connect to with the Pathway Terminal, you must configure the `unikixtnemux` TN3270 server. Refer to the *Sun Mainframe Transaction Processing Software Configuration Guide* for instructions.

Starting the Pathway Terminal

The procedures in this section describe how to start the Pathway Terminal on various platforms.

Starting the Pathway Terminal on Solaris and UNIX Systems

On Solaris or UNIX systems, there are two ways to start the Pathway Terminal.

▼ To Start the Pathway Terminal Using the Shell Script

1. **Ensure that the required level of JRE is on your path.**

   See "Obtaining a Java Runtime Environment (JRE)" on page 3.

2. **Run the `pathway_terminal.sh` shell script located in the `bin` directory of the product installation.**

▼ To Start the Pathway Terminal Using the JAR File

1. **Ensure that the required level of JRE is on your path.**

   See "Obtaining a Java Runtime Environment (JRE)" on page 3.

2. **Run the executable JAR file located in the `lib` directory of the product installation.**

   Change to the `lib` directory and type the following command:

```
$ java -jar pathway_terminal.jar
```

Starting the Pathway Terminal on Microsoft Windows Systems

There are three ways to start the Pathway Terminal.

▼ To Start Pathway Terminal Using the Start Menu

1. **On the desktop, click the Start button.**
2. **Select Start → Programs → Sun 3270 Pathway 2.0.0 → 3270 Terminal.**

▼ To Start Pathway Terminal Using the Executable

- **Run the `pathway_terminal.exe` program located in the `bin` directory of the product installation.**

▼ To Start Pathway Terminal Using the JAR File

1. **Ensure that the required level of JRE is on your path.**

   See "Obtaining a Java Runtime Environment (JRE)" on page 3.

2. **Run the executable JAR file located in the `lib` directory of the product installation.**

   Change to the `lib` directory and type the following command:

```
C:\> java -jar pathway_terminal.jar
```

   With all of these methods, you can specify startup parameters. You can modify the properties of the Start button entry for the Pathway Terminal or the properties of a shortcut you created. At a command prompt, specify the options on the command line. "Command Line Parameters" on page 90 describes these parameters.

Command Line Parameters

You can start the Pathway Terminal with a series of parameters. Supplying a URL or host and port causes the emulator to automatically attempt a connection to the specified host.

Format

`pathway_terminal [-h` *hostname* `[-p` *portnumber*`]] | [-u` *URL*`] [-m` *model*`] [-f` *font*`] [-n` *netname*`] [-t` *timeout-value*`] [-s` *print-style*`] [-c` *codepage*`] [-x] [-r [`*filename*`]]`

where:

`-h` *hostname*	Name of TN3270 server host.
`-p` *portnumber*	Port number of TN3270 server. This is the same as the port number specified as the `TNServer*ListenPort` in the `unikixrc.cfg` file.
`-u` *URL*	TN3270 host and port specified as a URL. For example: `tn3270://myserver.com:2002`
`-m` *model*	Terminal model; valid values are: 2, 2-E, 3, 3-E, 4, 4-E, 5, and 5-E. The default model is 2-E.
`-f` *font*	*font* is in the format *name.style.size*. For example: `-f Monospaced.PLAIN.12`.
`-n` *netname*	If you allow TN3270E, you can request a specific *netname* for this terminal.
`-t` *timeout-value*	Host inactivity *timeout-value* in seconds. The Pathway Terminal automatically disconnects from the host after a period of inactivity. Inactivity is determined from the last time any network activity between the host and the Pathway Terminal occurred. The default is not to time out, which is the same as specifying `-t 0`.
`-s` *print-style*	Style to use for printing. Valid values are: • `normal`: Print same as appears on screen • `whiteonblack`: Print in white on a black background • `blackonwhite`: Print in black on a white background • `coloronwhite`: Print in color on a white background The default is `normal`.

`-c` *codepage*	Indicates the codepage to use. One of the following: • IBM-037 (US English, Portuguese, Canadian French) • IBM-273 (German) • IBM-277 (Danish, Norwegian) • IBM-278 (Finnish, Swedish) • IBM-280 (Italian) • IBM-284 (Spanish) • IBM-285 (UK English) • IBM-297 (French) • IBM-500 (Belgian, Swiss German) • IBM-870 (Croatian, Czechoslovakian, Hungarian, Polish, Romanian, Serbian Latin, Slovak, Slovene) • IBM-875 (Greek) • IBM-930 (Japanese DBCS) • IBM-933 (Korean DBCS) • IBM-935 (Simplified Chinese DBCS) • IBM-937 (Traditional Chinese DBCS) • IBM-939 (Japanese DBCS) • IBM-1025 (Bulgarian, Macedonian, Serbian Cyrillic, Russian) • IBM-1026 (Turkish) • IBM-1047 (Open edition) • IBM-1140 (US English - Euro) • IBM-1141 (German - Euro) • IBM-1142 (Danish, Norwegian - Euro) • IBM-1143 (Finnish, Swedish - Euro) • IBM-1144 (Italian - Euro) • IBM-1145 (Spanish - Euro) • IBM-1146 (UK English - Euro) • IBM-1147 (French - Euro) • IBM-1148 (Multilingual - Euro) • IBM-1149 (Icelandic - Euro)
`-x`	Excludes the use of the TN3270E protocol when connecting to the remote system. Only TN3270 is used.
`-r` [*filename*]	Identifies the remote animation file to use. The file name is optional. If no file name is specified, the Pathway Terminal uses the file *INSTROOT*`/etc/animator_command`. Remote animation is supported only on Windows platforms.

3270 Emulator Window

The emulator window consists of three parts: the menu bar, the emulator itself, and a status bar.

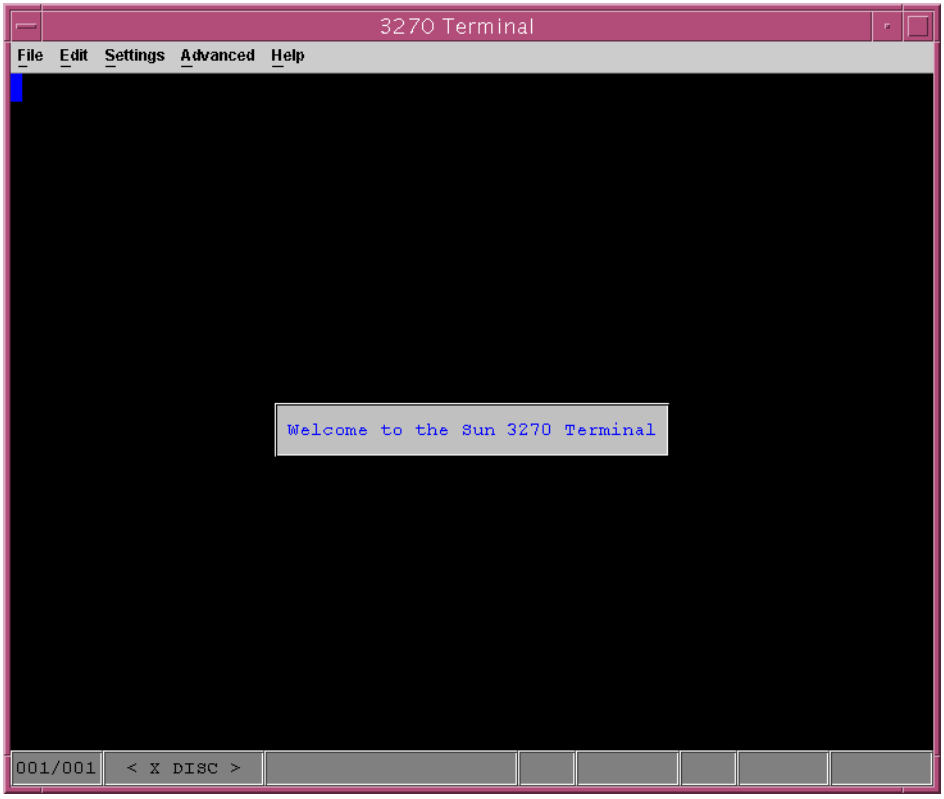

FIGURE 11-1 Disconnected Emulator

The Menu Bar

The emulator's menu bar has five items: File, Edit, Settings, Advanced, and Help.

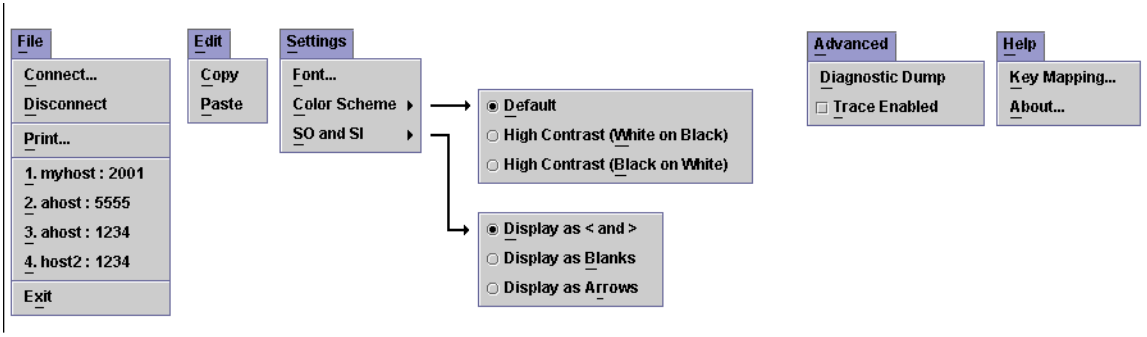

FIGURE 11-2 Menu Bar

The menu bar enables you to:

- Connect to a TN3270 system, if the emulator is in a disconnected state
- Disconnect from the currently connected system, if the emulator is in a connected state
- Print the current display
- Copy the screen to the system clipboard
- Paste the contents of the system clipboard to the emulator
- Change the font of the display
- Change the color settings of the emulator
- Change the way that SO and SI markers are displayed when running in DBCS mode
- Capture diagnostic information
- Display the current keyboard mapping

The File menu also displays the last four systems the emulator connected to. When the emulator is disconnected, clicking on one of the systems causes the emulator to attempt to connect to it.

The Emulator

When the emulator is connected, it responds to keyboard input like a standard 3270 terminal. For information on how 3270 keys are mapped to keys on your keyboard, see "Specifying Keyboard Mapping" on page 96.

By default, the terminal does not automatically disconnect (*timeout*) from a system that it is connected to. If you want it to automatically timeout, you must specify a terminal timeout value when starting the Pathway Terminal. See "Command Line Parameters" on page 90 for information about the timeout startup parameter.

The Status Bar

At the bottom of the emulator window is the status bar. See "The Status Bar" on page 24 for a description of the sections of the status bar.

Connecting to a 3270 System

If you specify the `-h` *hostname* or `-u` *URL* parameters when starting the Pathway Terminal, the emulator automatically attempts to connect to the specified host system. If you do not specify the `-h` or `-u` parameters, use the following procedure to connect to a host.

▼ To Connect to a Host

1. **Start the Pathway Terminal using one of the methods described in "Starting the Pathway Terminal" on page 88.**

   The terminal emulator window is displayed.

2. **Click File → Connect to display the Connect dialog box.**

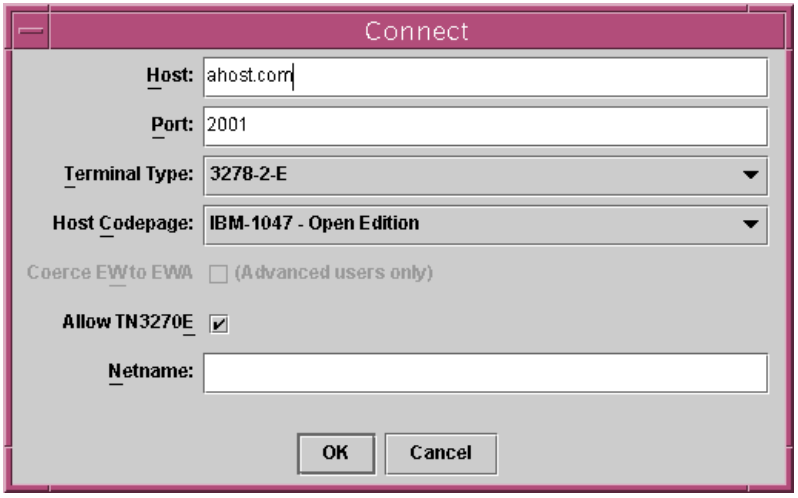

FIGURE 11-3 Emulator Connect Window

3. **Set the attributes by typing values in the following fields:**

   a. **In the Host field, type the host name.**

   b. **Type a port number in the Port field; the default is 2001.**

      This must be the same as the port number specified as the `TNServer*ListenPort` in the `unikixrc.cfg` file.

   c. **Select a terminal type for the Terminal Type field using the drop-down menu; the default is 3278-2-E.**

   d. **Select a codepage for the Host Codepage field by using the drop-down menu.**

      The default is IBM-1047 (Open Edition). The supported codepages are listed in "Command Line Parameters" on page 90.

   e. **The Coerce EW to EWA field is enabled only for Terminal types 3278-3, 3278-3-E, 3278-4, 3278-4-E, 3278-5, and 3278-5-E.**

      Some TN3270 servers do not generate correct 3270 datastreams for these terminal types. When you check this box, the Pathway Terminal displays the screens correctly.

Note – The coerce option is available only on the Connect window. There is no equivalent command-line parameter.

---

   f. **Check or deselect the Allow TN3270E check box, which allows the use of the TN3270E protocol when connecting to the remote system.**

g. Optionally, type a value in the Netname field.

If you checked Allow TN3270E, you can request a specific netname for the terminal.

4. Click Connect to start the connection process.

A message is displayed if the Pathway Terminal fails to connect to the specified host. If you connect successfully, the first screen of the 3270 host is displayed.

Note – The last four systems that were connected to, either successfully or unsuccessfully, are displayed on the Pathway Terminal's File menu. Clicking on one of these entries causes the Pathway Terminal to automatically attempt to connect to that host.

Copy and Paste

Use the copy action to copy the current emulator display to the clipboard of your operating system. To paste the contents of the system clipboard to the Pathway Terminal, use the paste action. The emulator cursor must be in a field that permits keyboard input to perform a successful paste operation.

By default, Ctrl-C can be used for the copy command and Ctrl-V for the paste command. These keystrokes can be remapped in the usual manner.

Specifying Keyboard Mapping

A real 3270 emulator has a number of special keys, such as Clear, and Reset, that are not provided on the standard PC keyboard. The Pathway Terminal maps certain keys and key combinations to the actions of these special keys. For more information about cursor control, see "Accessibility Features of the Pathway Terminal" on page 101.

▼ To View the Current Key Mapping

- **On the emulator window, select Help → Key Mapping.**

▼ To Change the Current Key Mapping

- **Edit the** `.pathway_terminal.keymap.properties` **file, which is located in your** `$HOME` **directory.**

  The following example shows some lines from this file:

```
pressed.BACK_SPACE=backspace
shift.pressed.TAB=backtab
pressed.ESCAPE=pressClear
pressed.ENTER=pressEnter
ctrl.pressed.ENTER=newline
```

  Each line in the `.pathway_terminal.keymap.properties` file maps one or more keystrokes to an action. Be careful when editing this file and be sure the keystrokes and action names are correct. If any actions are specified incorrectly, error messages are displayed when you start the Pathway Terminal.

  It is permissible to map multiple keys to one action. For example, to map Ctrl-Z and Shift-Enter to the newline action, type the following lines in the properties file:

```
CTRL.pressed.Z=newline
SHIFT.pressed.ENTER=newline
```

  Do not assign the same keystroke for two actions. For example the following lines, which attempt to assign Ctrl-Z to backspace and to newline, will not work:

```
CTRL.pressed.Z=backspace
CTRL.pressed.Z=newline
```

  You cannot change the key map while the Pathway Terminal is running. You must stop the Pathway Terminal, edit the file, and restart the Pathway Terminal.

▼ To Restore the Key Mapping to the Default Values

1. **Stop the Pathway Terminal.**
2. **Delete the** `.pathway_terminal.keymap.properties` **file.**
3. **Restart the Pathway Terminal.**

   The `.pathway_terminal.keymap.properties` file is automatically recreated.

Resizing the Pathway Terminal

When you resize the Pathway Terminal, it attempts to change the font size of the emulator, so that the emulator fits the specified size. This provides a simple way to make the emulator display more readable. However, it might result in an area smaller than the one you specified. If you are having difficulty achieving the results you want using this method, use the Settings → Font option to specify a particular size.

Diagnostic Information

There is a dump function within Pathway Terminal to enable you to obtain diagnostic information about the current state of the terminal.

Note – This diagnostic information is primarily for the use of support personnel and is unlikely to be of direct interest to most users.

The diagnostic dump command saves Pathway Terminal diagnostic information to a file. The file is saved to the directory from which the Pathway Terminal was started. When you issue a dump, a message displays the file and directory name. The dump saves general information about the emulator, as well as the current screen. If tracing is turned on, the file will also contain the data flows from the emulator to the host and from the host to the emulator. The trace information is from the time tracing was started until the time the dump was taken.

Using the Pathway Terminal in DBCS Mode

To use the Pathway Terminal in double-byte character set (DBCS) mode, choose one of the available DBCS codepages on the Connect window, or specify a DBCS codepage using the `-c` parameter of the `pathway_terminal` command.

Available 3270 Field Types

When running in DBCS mode, each 3270 field is one of the following types:

- A pure single-byte field
- A pure double-byte field
- A mixed single- and double-byte field known as a SOSI shift-out, shift-in) field

Restrictions on a Pure Single-byte Field

A pure single-byte field can only contain single-byte characters. Any attempt to type a character into a pure single-byte field that is not a single-byte character results in a terminal alarm.

Restrictions on a Pure Double-byte Field

A pure double-byte field can only contain double-byte characters. A double-byte character occupies two display positions and is twice the physical width of a single-byte character. Any attempt to type a character that is not a double-byte character into a pure double-byte field results in a terminal alarm.

Note – A character is considered double-byte if its representation in the selected host codepage is two bytes.

Restrictions on Mixed (SOSI) Fields

A SOSI field can contain both single- and double-byte characters. You must enclose all double-byte characters in a SOSI field within a pair of SOSI characters.

- SO and SI characters each occupy a display position.
- An SO character appears as a less than symbol (<).
- An SI character appears as a greater than symbol (>).
- SO and SI characters are inserted automatically as a terminal user types. The terminal user does not have to enter them.
- Any keyboard entry operation that would leave DBCS characters outside an enclosing SOSI pair is invalid and is rejected with a terminal alarm; for example, deleting an SI character that would leave a DBCS character immediately adjacent to a single-byte character set (SBCS) character.
- An attempt to type DBCS characters that does not leave room for the required SO and SI characters to be automatically inserted in the field is rejected with a terminal alarm; for example, typing a DBCS character in the last two display positions of a SOSI field.

Making an Association With an Internet Browser

Note – This section applies to users on Microsoft Windows systems.

You can configure some web browsers so that any links to TN3270 hosts are started with the Pathway Terminal. When you click on the link, the browser starts the Pathway Terminal and attempts to make a connection to the TN3270 system.

On your system, associate files of type URL:TN3270 Protocol with `pathway_terminal -u`. You must have the Pathway Terminal in your path or you must specify the complete path name where the `pathway_terminal` executable is installed. Refer to Microsoft Windows Help for instructions on associating files.

Accessibility Features of the Pathway Terminal

The Pathway Terminal software contains accessibility features, which make it usable by assistive technologies. Using the Pathway Terminal with the Sun MTP software makes Sun MTP accessible by assistive technologies.

The accessibility features of the Pathway Terminal are:

- Three color schemes:
  - Standard: The emulator displays colored text.
  - High contrast 1: The emulator text is white and the background is black.
  - High contrast 2: The emulator text is black and the background is white.

  Controls to set the color scheme, are located on the Settings menu. See FIGURE 11-2.
- Controls that move the cursor around the screen:
  - Move to first field: Moves the cursor to the first character position of the first field of the screen
  - Move to next field: Moves the cursor to the first character position of the next field relative to the field where the cursor is located
  - Move to previous field: Moves the cursor to the first character position of the previous field relative to the field where the cursor is located

  These controls accommodate those assistive technologies that require the cursor to be in a field for that field to be accessed.

  These movement controls differ from the Home, Tab, and Backtab controls, which only move the cursor to editable fields. These movement controls move the cursor to all fields.

  To view the current key mappings for these controls, follow the procedure in "To View the Current Key Mapping" on page 96. To define these movement controls, see "To Change the Current Key Mapping" on page 97.
- Support for the Java Accessibility API, which ensures that Pathway Terminal is accessible by assistive technologies that conform to the API. For example, a screen reader technology that conforms to the Java Accessibility API can read the contents of the emulator screen.

  The Java Access Bridge is required for an assistive technology to interact with a JVM. Before an assistive technology can be used with the Pathway Terminal, the Java Access Bridge must be installed into the JVM that is to be used to run the emulator. Note that the Windows version of Sun 3270 Pathway is distributed with its own JVM, which is used to run the emulator. You must ensure that this JVM has the Java Access Bridge applied to it to enable assistive technologies.

The Java Access Bridge can be obtained from:

http://java.sun.com/products/accessbridge

For instructions on how to install the Java Access Bridge, refer to its documentation.

Note – DBCS and SOSI fields might not report the correct information when an assistive technology inquires about the fields' contents.

Index

D

E

F

G

H

I

J

K

M

T

U

V

W