Novell exteNd Composer™ 3270 Connect

5.0

www.novell.com

USER'S GUIDE

Novell®

Novell, Inc.

1800 South Novell Place

Provo, UT 85606

www.novell.com

exteNd Composer 3270 Connect ***User's Guide***

January 2004

Contents

About This Guide

Purpose

The guide describes how to use exteNd Composer 3270 Connect, referred to as the 3270 Component Editor. The 3270 Component Editor is a separately-installed component editor in exteNd Composer.

Audience

The audience for the guide is developers and system integrators using exteNd Composer to create services and components which integrate 3270 applications.

Prerequisites

The guide assumes the reader is familiar with and has used exteNd Composer's development environment and deployment options. You must also have an understanding of the 3270 environment.

Additional documentation

For the complete set of Novell exteNd Composer documentation, see the Novell Documentation Web Site (**http://www.novell.com/documentation-index/index.jsp**).

Organization

The guide is organized as follows:

Chapter 1, *Welcome to exteNd Composer and 3270*, gives a definition and overview of the 3270 Component Editor.

Chapter 2, *Getting Started with the 3270 Component Editor*, describes the necessary preparations for creating a 3270 component.

Chapter 3, *Creating a 3270 Component*, describes the parts of the component editor.

Chapter 4, *Performing 3270 Actions*, describes how to use the basic 3270 actions, as well as the 3270 Multi Row Wizard.

Chapter 5, *Logon Components, Connections and Connection Pools*, describes how to create logon components, connections and connection pools.

Appendix A, *Testing*, describes environmental differences between animation testing and deployment testing.

Appendix B, *Java Code Pages*, provides reference information on character encoding conversions.

Appendix C, is a glossary.

Appendix D, *Reserved Words*, is a section of those words used only for the 3270 Connect.

Conventions Used in the Guide

The guide uses the following typographical conventions.

Bold typeface within instructions indicate action items, including:

- Menu selections
- Form selections
- Dialog box items

Sans-serif bold typeface is used for:

- Uniform Resource Identifiers
- File names
- Directories and partial pathnames

Italic typeface indicates:

- Variable information that you supply
- Technical terms used for the first time
- Title of other Novell publications

`Monospaced` typeface indicates:

- Method names
- Code examples
- System input
- Operating system objects

1 Welcome to exteNd Composer and 3270 User Interface

Before You Begin

Welcome to the *3270 Connect Guide*. This Guide is a companion to the *exteNd Composer User's Guide*, which details how to use all the features of Composer, except the Connect Component Editors. So, if you haven't looked at the User's Guide yet, please familiarize yourself with it before using this Guide.

exteNd Composer provides separate Component Editors for each Connector, like 3270. The special features of each component editor are described in separate Guides like this one.

If you have been using exteNd Composer, and are familiar with the core component editor, the XML Map Component Editor, then this Guide should get you started with the 3270 Component Editor.

Before you can begin working with the 3270 Connect, you must have installed it into your existing exteNd Composer. Likewise, before you can run any Services built with this Connector in the exteNd Composer Enterprise Server environment, you must have already installed the Server side software for this Connector into exteNd Composer Enterprise Server.

NOTE: To be successful with this Component Editor, you must be familiar with the IBM 3270 environment and the applications that you want to XML-enable.

About exteNd Composer Connectors

exteNd Composer is built upon a simple hub and spoke architecture. The hub is a robust XML transformation engine that accepts requests via XML documents, performs transformation processes on those documents and interfaces with XML-enabled applications, and returns an XML response document. The spokes, or Connectors, are plug-in modules that "XML-enable" sources of data that are not XML aware, bringing their data into the hub for processing as XML. These data sources can be anything from legacy COBOL/applications to Message Queues to HTML pages.

exteNd Composer Connectors can be categorized by the integration strategy each one employs to XML enable an information source. The integration strategies are a reflection of the major divisions used in modern systems designs for Internet- based computing architectures. Depending on your B2B needs and the architecture of your legacy applications, exteNd Composer can integrate your business systems at the User Interface, Program Logic, or Data levels.

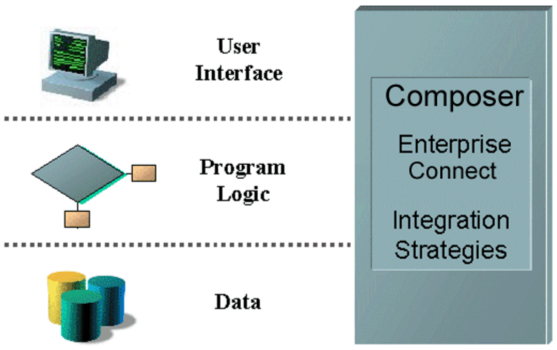


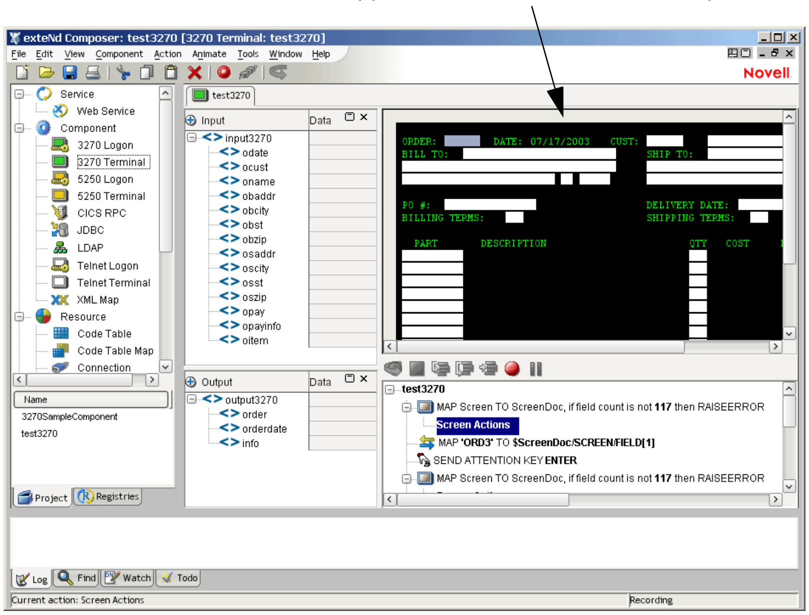
Figure 1-1

What is the 3270 (TDS) Connect?

The 3270 Connect XML-enables IBM compatible mainframe legacy system data using the User Interface integration strategy by hooking into the Terminal Data Stream (TDS). The term 3270 is commonly used to refer to the generic "dumb terminal" types used to connect to IBM mainframe systems. When connecting to an IBM Mainframe, the 3270 TDS uses IBM’s EBCDIC character-encoding scheme. The 3270 TDS, which was developed in the 1960s, emerged as that generation’s standard, and persists today. The 3270 TDS allows users to interact with legacy applications through the use of attention keys (e.g., Enter and PF Keys) that are interpreted by the application running on the mainframe to perform the appropriate actions. This interaction, through a dumb terminal, means that all the data is processed information from the mainframe computer. 3270 terminal emulation software can be used to make a microcomputer or PC act as if it were a 3270-type terminal while it is communicating with a mainframe.

Using the 3270 Connect, you can make legacy applications and their business logic available to the internet, extranet, or intranet processes. You can navigate through an application as if you were at a terminal session, use XML request documents to drive the inquiries and updates into the screens rather than keying, use the messages returned from applications screens to make the same decisions as if you were at a terminal, and move the data and responses into XML documents that can be returned to the requestor or continue to be processed. The 3270 screens appear in the Native Environment pane of the 3270 Component Editor.

3270 Screens appear in the Native Environment pane



```
ORDER:        DATE: 07/17/2003   CUST:
BILL TO:                               SHIP TO:


PO #:                                  DELIVERY DATE:
BILLING TERMS:                         SHIPPING TERMS:

  PART        DESCRIPTION                    QTY   COST
```

test3270
- MAP Screen TO ScreenDoc, if field count is not **117** then RAISEERROR
  - Screen Actions
- MAP **'ORD3'** TO **$ScreenDoc/SCREEN/FIELD[1]**
- SEND ATTENTION KEY **ENTER**
- MAP Screen TO ScreenDoc, if field count is not **117** then RAISEERROR

About exteNd Composer’s 3270 Component

Much like the XML Map component, the 3270 component is designed to map, transform, and transfer data between two different XML templates (i.e., request and response XML documents). However, it is specialized to make a connection (either TN3270 or EPI) to a mainframe application, process the data using elements from a DOM, and then map the results to an output DOM. You can then act upon the output DOM in any way that makes sense for your integration application. In essence, you’re able to capture data from, or push data to, a legacy system without ever having to alter the legacy system itself.

A 3270 component can perform simple data manipulations, such as mapping and transferring data from an XML document into a mainframe transaction, or perform “screen scraping” of a 3270 transaction, putting the data into an XML document. It can also perform sophisticated operations, such as mapping and manipulating multi-row and multi-screen transactions. The 3270 component has all the functionality of the XML Map component and can process XSL, send mail, and post and receive XML documents using the HTTP protocol.

The following illustration shows how a 3270 component uses a TN3270 or EPI connection to interact with data on the mainframe.

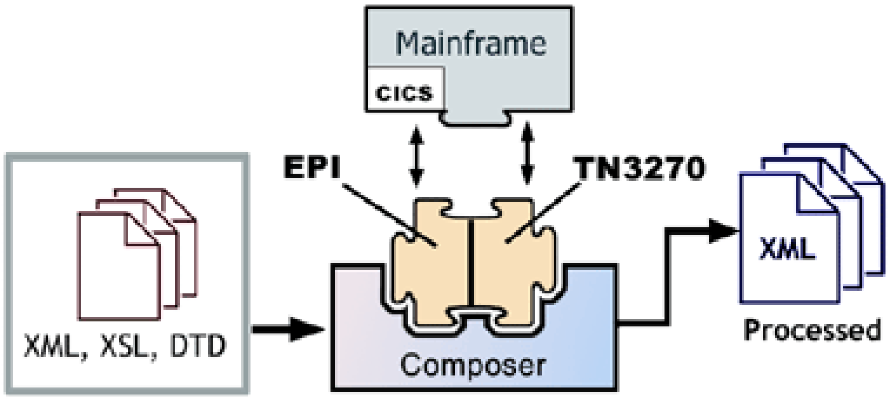


Figure 1-2

What Applications Can You Build Using the 3270 User Interface Component Editor?

The 3270 User Interface Component Editor allows you to extend any XML integration you are building to include any of your business applications that support 3270-based terminal interactions (See *exteNd Composer User's Guide* for more information.) For example, you may have an application that retrieve a product's description, picture, price, and inventory from regularly updated databases and displays it on a Web browser. By using the 3270 Component Editor, you can now get the current product information from the operational systems and the static information (e.g., the picture) from the database and merge the information from these separate information sources before displaying it to a user. This provides the same current information to both your internal and external users.

Where to Go for More Help

Perhaps the best way to understand the 3270 Connect in Composer is to see it in action. If you would like to take a Tutorial for the 3270 Connect that provides access to legacy transactions in a test environment, please contact Novell Technical Support for the connection information, tutorial instructions and any other supporting materials.

Additionally, you will find information on the Novell website: **www.Novell.com**.

2 Getting Started with the 3270 Component Editor

The Sample Transactions

For demonstration purposes, three transactions are used throughout this document in the samples presented: PART, GORD, and MENU. These transactions represent typical transactions used by operators. The PART transaction represents a scenario in which an operator uses a SKU number to drive an inquiry to a database. The GORD transaction represents a scenario in which an order for an item or several items is placed. And the MENU transaction represents a scenario in which an operator navigates through a menu-driven application to get to a particular screen. The PART, GORD, and MENU transactions are used to show you how to build Composer serviceComposer services that do the same things as the real life scenarios.

Steps Commonly Used to Create a 3270 Component

There are many ways to go about creating 3270 components; however, the most commonly used steps in creating a simple 3270 component are as follows:

1. Create XML Template(s) for the transaction.
2. Create a Connection Resource.
3. Create a component.
4. Enter Record mode and navigate to transaction.
5. Drag and drop input document data into screen.
6. Process the transaction from the keyboard action.
7. Drag and drop screen results into output document.
8. Stop recording.

Creating a 3270 Connection Resource

Before you can create a 3270 Component, you need to create a Connection Resource to access the mainframe transaction.

About Connection Resources

When you create a Connection Resource for the 3270 Component, you have two choices: a TN3270 Connection or an EPI 3270 Connection. You can use the TN3270 Connection when you want to connect to an IBM mainframe environment, regardless of the application or Transaction Processing Monitor you are using. The EPI 3270 Connection is available when you want to connect directly to a CICS application region, that supports the External Presentation Interface (EPI) API. This connection allows you to utilize TCP/IP to access the EPI gateway on a CICS host.

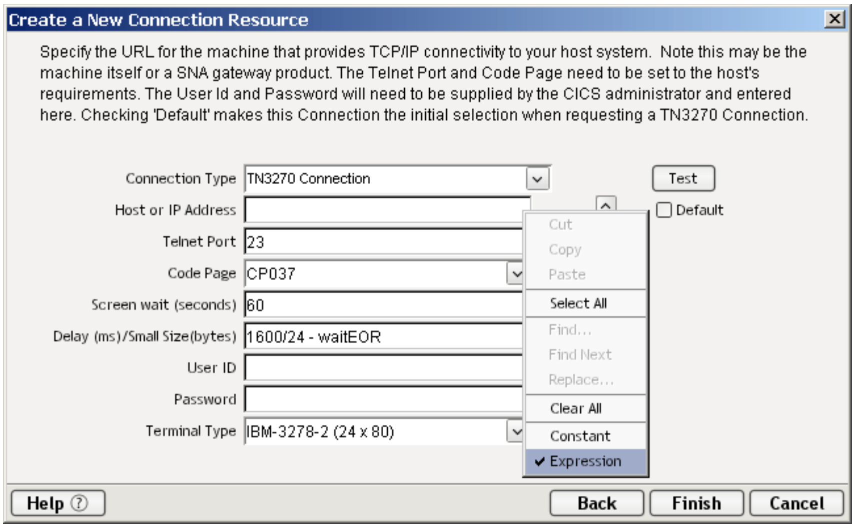
About Constant and Expression Driven Connections

You can specify Connection parameter values in one of two ways: as Constants or as Expressions. A constant based parameter uses the value you type in the Connection dialog every time the Connection is used. An expression-based parameter allows you to set the value using a programmatic expression, which can result in a different value each time the connection is used at runtime. This allows the Connection's behavior to be flexible and vary based on runtime conditions each time it is used.

For instance, one very simple use of an expression driven parameter in a EPI 3270 Connection would be to define the User ID and Password as PROJECT Variables (e.g. PROJECT.XPATH("USERCONFIG/MyDeployUser"). This way when you deploy the project, you can update the PROJECT Variables in the Deployment Wizard to values appropriate for the final deployment environment. At the other extreme, you could have a custom script that queries a Java business object in the Application Server to determine what User ID and Password to use.

➢ **To switch a parameter from Constant driven to Expression driven:**

1. Click the RMB in the parameter field you are interested in changing.
2. Select **Expression** from the context menu and the editor button will appear or become enabled.
3. Click on the button and then create an expression that evaluates to a valid parameter value at runtime.


Create a New Connection Resource

Specify the URL for the machine that provides TCP/IP connectivity to your host system. Note this may be the machine itself or a SNA gateway product. The Telnet Port and Code Page need to be set to the host's requirements. The User Id and Password will need to be supplied by the CICS administrator and entered here. Checking 'Default' makes this Connection the initial selection when requesting a TN3270 Connection.

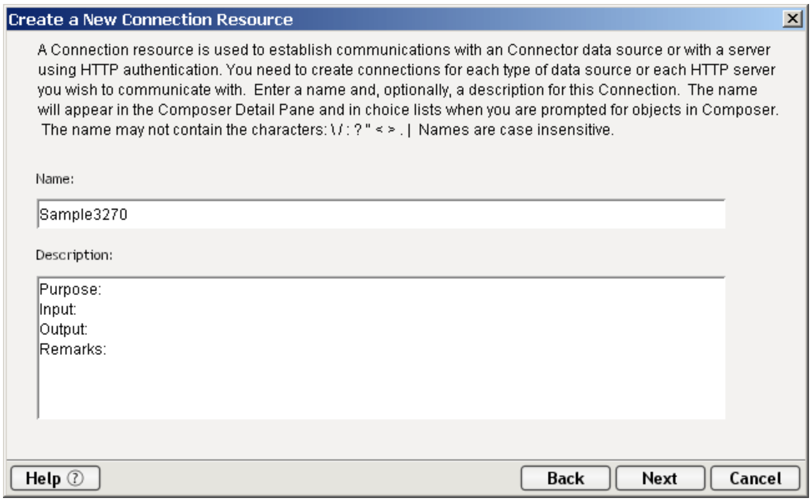
Connection Type	TN3270 Connection
Host or IP Address	
Telnet Port	23
Code Page	CP037
Screen wait (seconds)	60
Delay (ms)/Small Size(bytes)	1600/24 - waitEOR
User ID	
Password	
Terminal Type	IBM-3278-2 (24 x 80)

➢ **To create a 3270 Connection Resource:**

1 From the Composer **File** menu, select **New**, then **xObject,** then open the **Resource** tab and select **Connection**.

NOTE: Alternatively, you can highlight **Connection** in the Composer window category pane, right click your mouse button, then select **New**.

The Create a New Connection Resource Wizard appears.


Create a New Connection Resource

A Connection resource is used to establish communications with an Connector data source or with a server using HTTP authentication. You need to create connections for each type of data source or each HTTP server you wish to communicate with. Enter a name and, optionally, a description for this Connection. The name will appear in the Composer Detail Pane and in choice lists when you are prompted for objects in Composer. The name may not contain the characters: \ / : ? " < > . | Names are case insensitive.

Name:

Sample3270

Description:

Purpose:
Input:
Output:
Remarks:

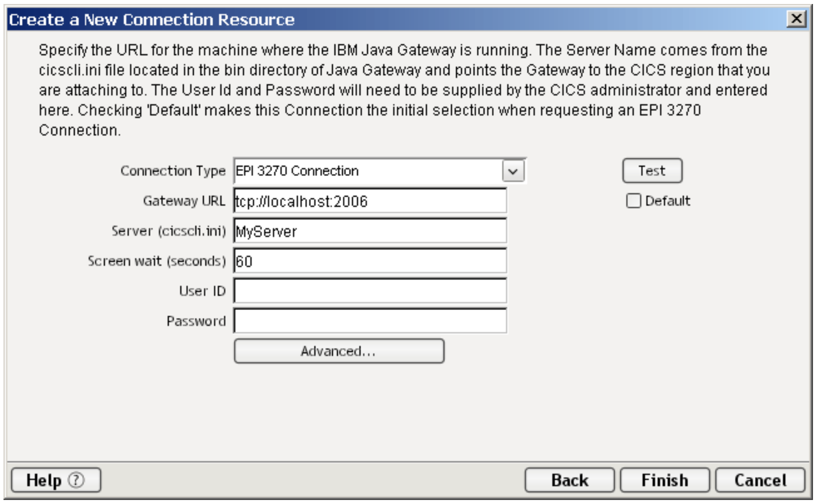
2 Type a **Name** for the connection object.

3 Optionally, type **Description** text.

4 Click **Next**.

➤ To select an EPI 3270 Connection:

1 Select **EPI 3270 Connection** if you are connecting to a CICS application through the CICS Java Gateway.



Create a New Connection Resource

Specify the URL for the machine where the IBM Java Gateway is running. The Server Name comes from the cicscli.ini file located in the bin directory of Java Gateway and points the Gateway to the CICS region that you are attaching to. The User Id and Password will need to be supplied by the CICS administrator and entered here. Checking 'Default' makes this Connection the initial selection when requesting an EPI 3270 Connection.

Connection Type	EPI 3270 Connection
Gateway URL	tcp://localhost:2006
Server (cicscli.ini)	MyServer
Screen wait (seconds)	60
User ID	
Password	

2 In the **Gateway URI** field, enter the physical address of the machine to which you are connecting. In the example above, *tcp://localhost.2006* is an alias for a physical machine where *tcp://* is the protocol, *localhost* is the name of the machine, and *2006* is the default gateway to CICS.

NOTE: The **Gateway URI** field is case sensitive.

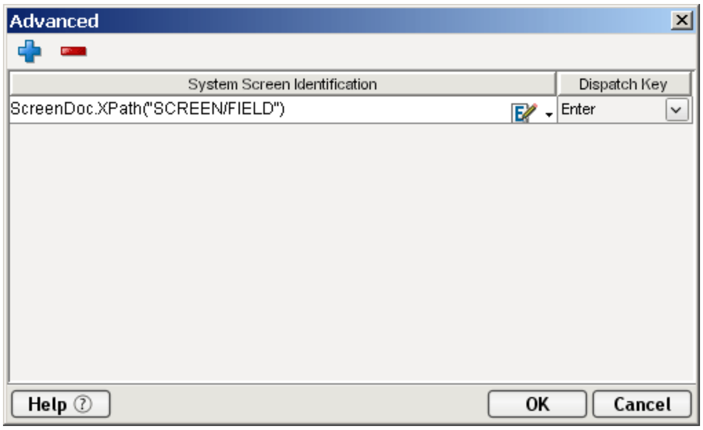
3 In the **Server** field, enter the region alias or server name. Your system administrator will provide you with this name, which is defined in a separate client initialization file.

4 The **Screen wait (seconds)** field, displays the amount of time in seconds that a 3270 Terminal component will wait for the arrival of the next screen in the Map Screen Action pane.

5 Enter a **UserID** and **Password**. They are simply defined here (password is encrypted). The user will have access to UserID and Password from ECMAScript, allowing them to map UserID and Password as a variable into the screen. This way, no one ever sees the passwords.

6 Select the **Default** check box if you'd like the settings you've entered to become the default settings for subsequent 3270 Connections.

7 Click on **Advanced** to display the System Handling dialog.



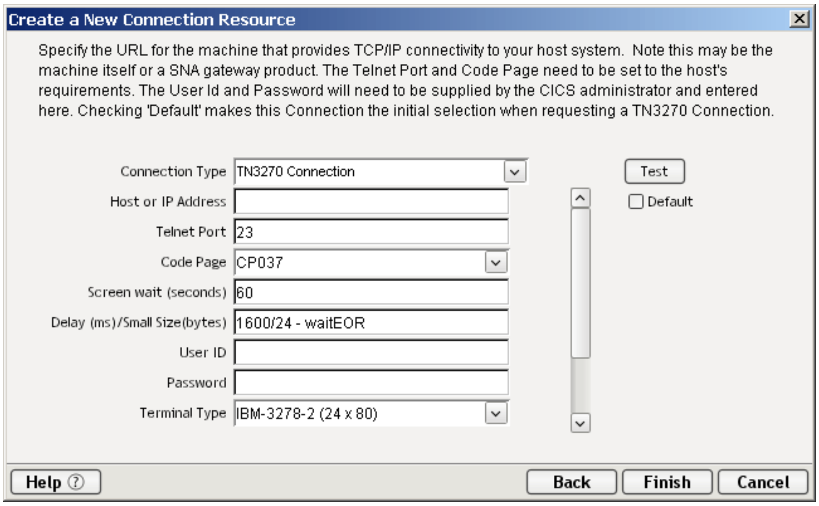
System Screen Identification	Dispatch Key
ScreenDoc.XPath("SCREEN/FIELD")	Enter

For more details on adding screens and AID keys to handle them, refer to the Screen Handling procedures mentioned in “Handling System Messages” below.

8 Click **Finish**. The newly created resource connection object appears in the Composer Connection Resource detail pane.

➢ **To select a TN3270 Connection:**

1 Select a **TN3270 Connection** type if you are using an application other than CICS.

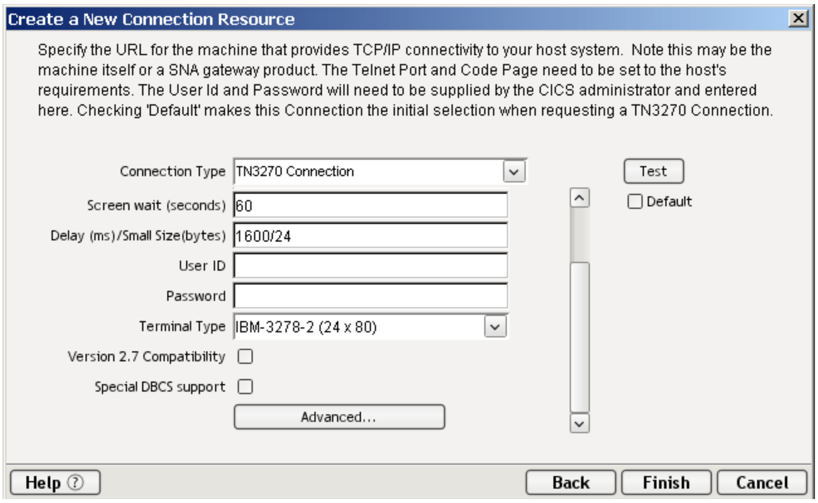


Specify the URL for the machine that provides TCP/IP connectivity to your host system. Note this may be the machine itself or a SNA gateway product. The Telnet Port and Code Page need to be set to the host's requirements. The User Id and Password will need to be supplied by the CICS administrator and entered here. Checking 'Default' makes this Connection the initial selection when requesting a TN3270 Connection.

Connection Type	TN3270 Connection
Host or IP Address	
Telnet Port	23
Code Page	CP037
Screen wait (seconds)	60
Delay (ms)/Small Size(bytes)	1600/24 - waitEOR
User ID	
Password	
Terminal Type	IBM-3278-2 (24 x 80)

☐ Default

2. In the **Host or IP Address** field, enter the physical address or alias for the machine to which you are connecting. Your system administrator will provide you with this information, which is defined in a separate host file.
3. In the **Telnet Port** field, enter the number of the port. The default port number is 23.
4. In the **Code Page** field, specify a code page (See “About Code Page Support” on page 25).
5. The **Screen wait (seconds)** field, displays the amount of time in seconds that a 3270 Terminal component will wait for the arrival of the next screen in the Map Screen Action pane.
6. In the **Delay (ms)/Small Size (bytes)** field, enter an appropriate setting. This setting helps solve network performance issues by waiting extra time **(ms)** for a complete packet to arrive when the packet received is smaller than **(bytes)**. In the above example, the network is instructed to wait an additional **1600** ms for a complete packet to arrive when the packet received is smaller than **24** bytes. An additional parameter, **-waitEOR**, allows a ‘screen complete’ state to be determined by waiting for a packet containing EOR (End of Record).
7. Enter a **UserID** and **Password**. These are not actually submitted to the host during the establishment of a connection. They are simply defined here (password is encrypted). The user will have access to UserID and Password from ECMAScript, allowing them to map UserID and Password as a variable into the screen. This way, no one ever sees the passwords.
8. In the **Terminal Type** field, lists the various types of terminals supported by 3270 components, including different screen sizes (i.e. 24x80 and 27x32). Select from the drop down list box the type of terminal you are using.
9. Select the **Terminal Type** from the dropdown list.

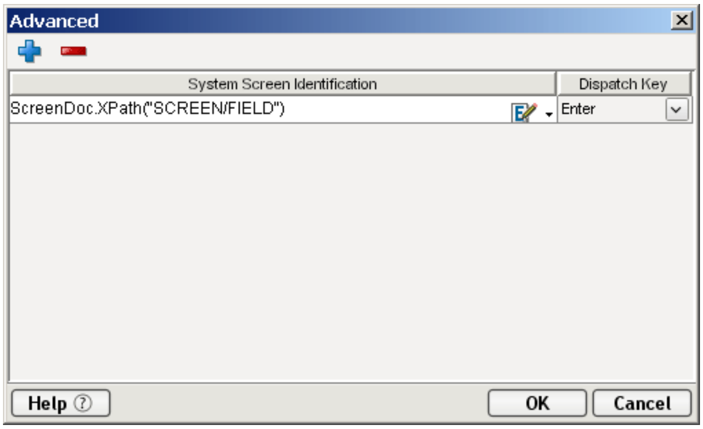

Create a New Connection Resource

Specify the URL for the machine that provides TCP/IP connectivity to your host system. Note this may be the machine itself or a SNA gateway product. The Telnet Port and Code Page need to be set to the host's requirements. The User Id and Password will need to be supplied by the CICS administrator and entered here. Checking 'Default' makes this Connection the initial selection when requesting a TN3270 Connection.

Connection Type	TN3270 Connection
Screen wait (seconds)	60
Delay (ms)/Small Size(bytes)	1600/24
User ID	
Password	
Terminal Type	IBM-3278-2 (24 x 80)
Version 2.7 Compatibility	☐
Special DBCS support	☐

10 Click in the checkbox to enable **Version 2.7 Compatability**.

11 Click on **Advanced** to display the System Handling dialog.


Advanced

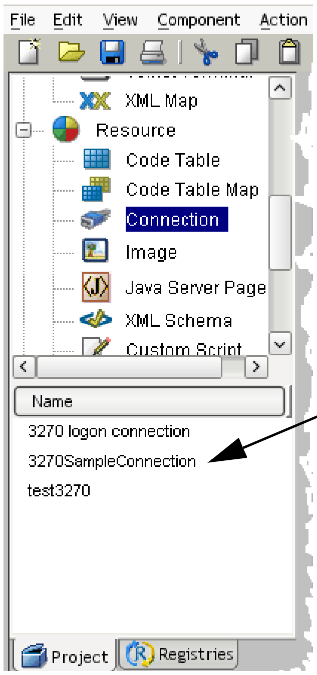
System Screen Identification	Dispatch Key
ScreenDoc.XPath("SCREEN/FIELD")	Enter

For more details on adding screens and AID keys to handle them, refer to the Screen Handling procedures mentioned in “Handling System Messages” below.

12 Select the **Default** check box if you’d like the settings you’ve entered to become the default settings for subsequent TN3270 Connections.

13 Click on the **Test** Button to see if your connection works.

14 Click **Finish**. The newly created resource connection object appears in the Composer Connection Resource detail pane.



3270 logon connection
3270SampleConnection
test3270

About Code Page Support

Code Page support in exteNd Composer Connection Resources allows you to specify which Character Encoding scheme to use when translating characters sent between exteNd Composer and other host systems. exteNd Composer data uses UniCode character encoding (the Java and XML standard). Existing legacy and other host systems use a variety of character encoding schemes (i.e., Code Pages) specific for their language or usage. A mechanism is needed to translate the character encoding between these systems if they are to communicate with one another. This is handled in exteNd Composer by specifying the Code Page used by a host system in the Connection Resource used to access that system. For more information on encoding, see Appendix B, “Java Code Pages”.

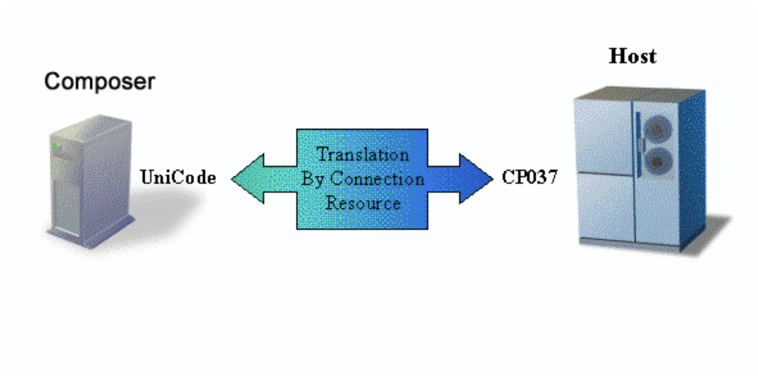


Figure 2-1

Handling System Messages

A special feature of 3270 Connections is the ability to handle unpredictable message screens received during a terminal session that would be problematic to a 3270 Component during its execution. One example of this type of message screen is when the system administrator broadcasts notices to terminals warning them of system shutdowns or other events. The arrival of this screen is both unpredictable and unwanted during the normal execution of a 3270 Component interacting with a system transaction. These system messages are usually dispatched by a user hitting an AID key after having read them.

Both types of 3270 Connections (TN3270 and EPI 3270) allow you to define special screen handlers to respond to screens you wish to exclude from component processing. You can also define multiple screen handlers for a single connection. By defining screen handlers in the Connection Resource, you can define your handlers once and multiple components can take advantage of them by using the same connection.

NOTE: Connection screen handlers are not restricted to messages from the System Administrator. You can define a screen handler to respond to any screen received in the terminal data stream.

➢ **General steps to creating a Screen Handler**

1. Create a working 3270 Component that uses a 3270 Connection Resource.
2. Reproduce and capture the screen you wish to handle in a ScreenDoc DOM.
3. Select a field/data combination on the screen to uniquely identify this screen to the screen handler.
4. Add the screen handler to the Connection Resource by supplying a screen identity expression and an AID key for dispatching the screen.

➢ **Specific steps to creating a Screen Handler**

1. Create your 3270 Component as you normally would (i.e. create a Connection and other Resources as necessary and then build the component)
2. If your ScreenDoc DOM is not already visible, select **View/Window Layout** from the menu bar, followed by clicking the **XML Layout** tab. Then move ScreenDoc to the Visible list and click **OK**.
3. Save the component but leave it open. Capture the particular screen you wish to handle as follows.

   NOTE: This will take some coordination with your system operator/administrator or whoever is responsible for generating the message screen you wish to handle.

a. Set a breakpoint on an action in the first Map Screen after the map Screen containing your Log in actions. (If this is a Logon Component, then set the breakpoint on the first action in the KeepAlive or Logout section.)

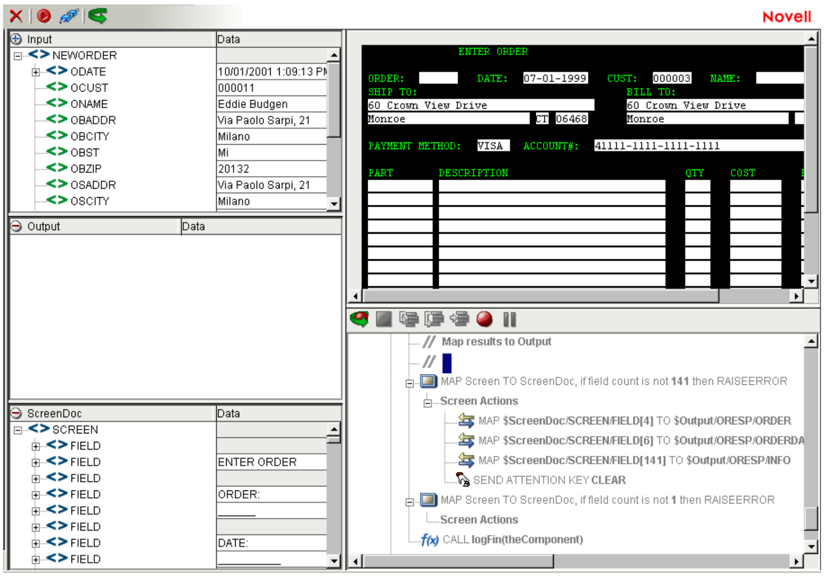
b. Animate the component by pressing **F5**.

c. Run to the Breakpoint by pressing **F9**.

d. Pause the Animation by pressing **F6**.

e. Initiate the system message.

f. Press **Enter** and the system message screen will appear in the Native Environment Pane and the XML representation of the message will appear in the Screen Doc pane.



Input	Data
NEWORDER	
ODATE	10/01/2001 1:09:13 PM
OCUST	000011
ONAME	Eddie Budgen
OBADDR	Via Paolo Sarpi, 21
OBCITY	Milano
OBST	Mi
OBZIP	20132
OSADDR	Via Paolo Sarpi, 21
OSCITY	Milano

Output	Data

ScreenDoc	Data
SCREEN	
FIELD	
FIELD	ENTER ORDER
FIELD	
FIELD	ORDER:
FIELD	________
FIELD	
FIELD	DATE:
FIELD	________

```
                     ENTER ORDER

ORDER:          DATE: 07-01-1999   CUST: 000003   NAME:
SHIP TO:                           BILL TO:
60 Crown View Drive                60 Crown View Drive
Monroe                    CT 06468 Monroe

PAYMENT METHOD: VISA  ACCOUNT#: 41111-1111-1111-1111

PART       DESCRIPTION                         QTY    COST
```

```
// Map results to Output
//
MAP Screen TO ScreenDoc, if field count is not 141 then RAISEERROR
   Screen Actions
      MAP $ScreenDoc/SCREEN/FIELD[4] TO $Output/ORESP/ORDER
      MAP $ScreenDoc/SCREEN/FIELD[6] TO $Output/ORESP/ORDERDA
      MAP $ScreenDoc/SCREEN/FIELD[141] TO $Output/ORESP/INFO
      SEND ATTENTION KEY CLEAR
MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
   Screen Actions
f(x) CALL logFin(theComponent)
```

4 Choose a test field whose data will serve to identify this screen as one you wish to handle. For instance, you may choose the screen title field whose data is “Display Messages.”

5 Hover your mouse pointer over the test field’s XML representation in the ScreenDoc and note its full XPath location (e.g. ScreenDoc/SCREEN/FIELD[5])

6 Stop Animation by pressing **Shift-F5**, sign off the host system, and close the component, but do not save it.

7 In the Composer window, select the Connection Resource for the component and open it.

8 On the **Connection Info** tab, press click on the **Advanced** button and the System Screen handling dialog appears.

9 Check to **Enable TN3270E** protocol support. If the checkbox is unchecked, disable and clear the Resource Name Text field.

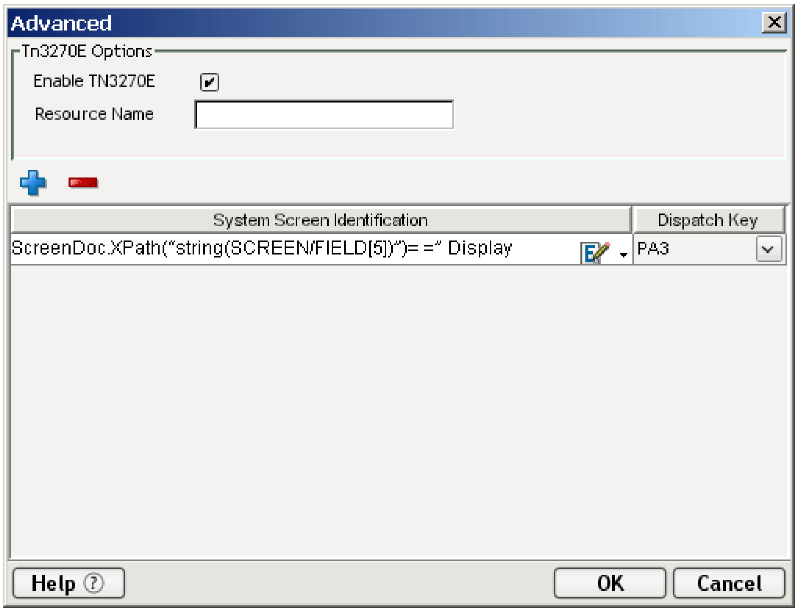
10 Enter the **Resource Name** used in the TN3270 Connect, leave blank if the Server is supplying the name. If you right-click on the Resource Name field, you can change the value from a constant to an Expression and use ECMAScript to fill in the value.

11 Click on the + icon to add a line to define Screen handler.

12 In the Expression field, type an expression comparing the test data against the test field's XPath (e.g. ScreenDoc.XPath("string(SCREEN/FIELD[5])")= =" Display Messages").

NOTE: the XPath string function is applied to make sure the comparison works properly.

13 In the **AID/DispatchKey** field, select a key to send in response to the screen.



Enable TN3270E ✓

Resource Name

System Screen Identification	Dispatch Key
ScreenDoc.XPath("string(SCREEN/FIELD[5])")= =" Display	PA3

14 Click on **OK** to save the Handler, and click on **OK** to save the connection.

15 Test your System Screen Handler by animating the component past the Logon Map Screen actions, setting a breakpoint, initiating the system message, and continuing your animation past an Enter Send Key action.

Creating a Style Sheet Resource

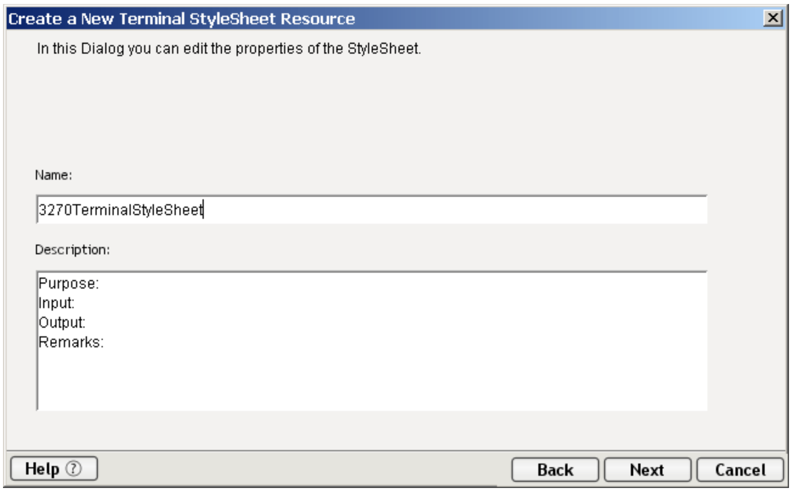
An additional resource associated with the 3270 Connector is the style sheet resource. This allows you to create a style sheet with which to display the emulation screen in the native environment pane.

➤ **To create a Style Sheet Resource:**

1. Select **File>New>xObject** from the Composer menu, then click on the **Resource** tab and select **Terminal Style Sheet**.

   NOTE: Alternatively, you may highlight **Terminal Style Sheet** in the **Resource** section of Composer's category pane, click your right mouse button, and select **New**.

   The Create a New Terminal Style Sheet wizard appears.



In this Dialog you can edit the properties of the StyleSheet.

Name:

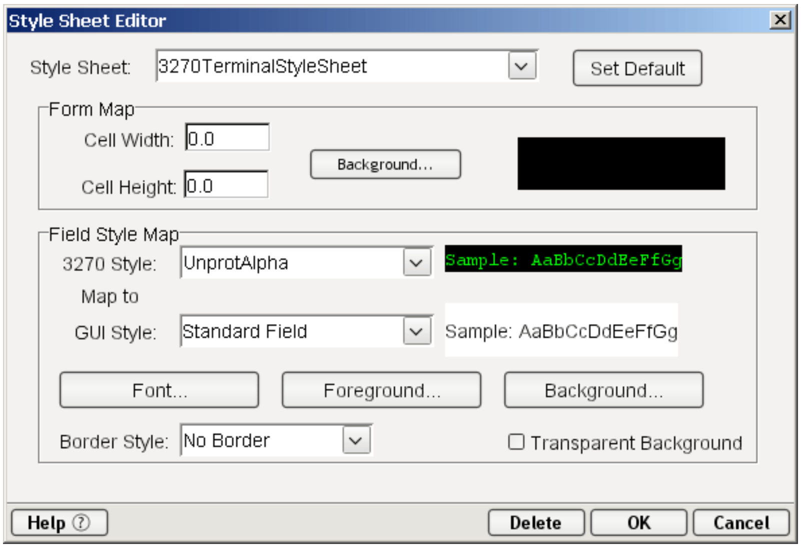
3270TerminalStyleSheet

Description:

Purpose:
Input:
Output:
Remarks:

2. Type a **Name** for the new style sheet. Optionally, you may type in Description text.

3 Click the **Next** button. The Style Sheet Editor window appears.



Style Sheet: 3270TerminalStyleSheet

Cell Width: 0.0

Cell Height: 0.0

3270 Style: UnprotAlpha

Sample: AaBbCcDdEeFfGg

GUI Style: Standard Field

Sample: AaBbCcDdEeFfGg

Border Style: No Border

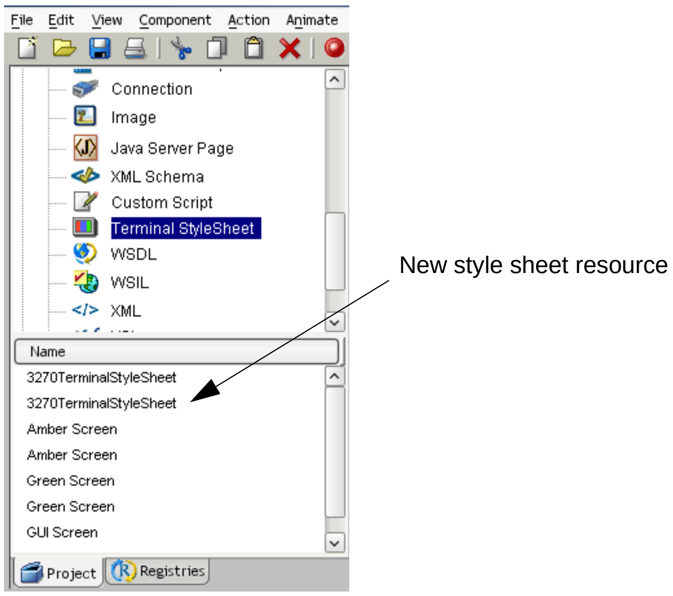
☐ Transparent Background

4 Use the Style Sheet Editor as described below to configure your style sheet:

- **Style Sheet** - Select a style sheet from this drop down list to change the appearance of the emulation screen in the native environment pane. This field initially contains the name you specified on the first page of the Terminal Style Sheet wizard. To create a new style sheet, type a name over one of the names in the list.
- **Set Default** - Select this button to make the currently selected style sheet the default for a component.
- **Form Map**
  Cell Width/Height - Modify these settings for drawing characters that may be truncated by changing font types.
  Background - Select this button to see background color options for the style sheet.

- **Field Style Map**
  3270 Style - This control lists the styles available from the TDS. You cannot edit these values. Select the style you wish to map to a new style you create.
  GUI Style - This control lists available styles you create. Type over an existing style to create a new one, then specify its **Font**, **Foreground**, and **Background** using the corresponding buttons.
  Border Style - Select one of three pre-defined borders from this drop down list. You cannot edit this control.
  Transparent Background - Select this check box if you want the GUI to have a transparent background.

5 Click **OK**. The newly created style sheet resource appears in Composer's detail pane.



Creating XML Templates for Your Component

In addition to a connection resource, a 3270 component may also require that you have already created XML templates so that you have sample documents for designing your component. See Chapter 5, *Creating XML Templates* in the *exteNd Composer User's Guide* for more information.

In many cases, your input documents will be designed to contain data that a terminal operator might type into the transaction interactively. Likewise, the output documents are designed to receive data returned to the screen as a result of the operator's input. For example, in a business scenario, a terminal operator may receive a phone request from a customer interested in the price or availability of an item. The customer gives the operator a SKU number, which he or she enters into the system. This represents a SKU number in an input document. Likewise, when the operator's terminal receives information back from the mainframe, based on the SKU, and the operator provides this information back to the customer, this represents the output document.

Also, if your component design calls for any other xObject resources such as custom scripts or code table maps, it is best to create these before creating the 3270 Component. For more information, see the *exteNd Composer User's Guide*.

3 Creating a 3270 Component

Before Creating a 3270 Component

As with all exteNd Composer components, the first step in creating a 3270 component is to specify the XML templates needed. For more information, see *Creating a New XML Template* in the *Composer User's Guide*.

Once you've specified the XML templates, you can create a component, using the template's sample documents to represent the inputs and outputs processed by your component.

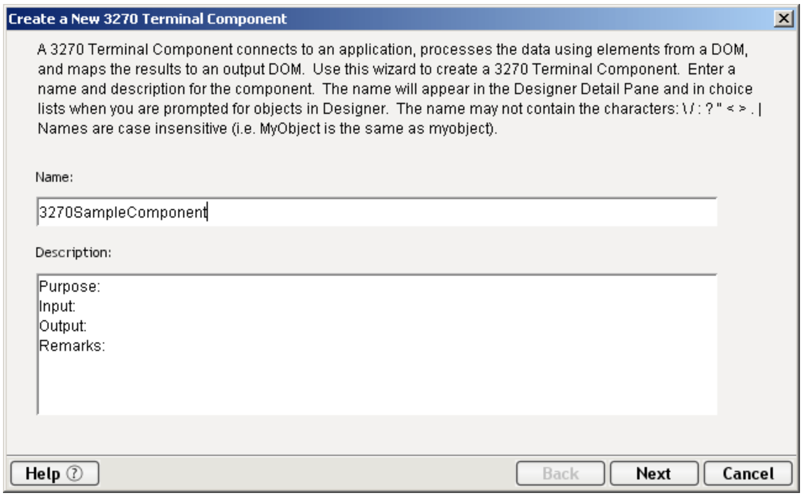
Also, as part of the process of creating a 3270 component, you must select a 3270 connection or you can create a new one. See "Creating a 3270 Connection Resource" on page 18.

➢ **To create a new 3270 Component:**

1 Select **File>New>xObject**. Select the **Component** tab and choose **3270 Terminal**.

NOTE: Alternatively, under **Component** in the Composer window category pane you can highlight **3270 Terminal**, click the right mouse button, then select **New**.

2 The Create a New 3270 Component Wizard appears.


Create a New 3270 Terminal Component

A 3270 Terminal Component connects to an application, processes the data using elements from a DOM, and maps the results to an output DOM. Use this wizard to create a 3270 Terminal Component. Enter a name and description for the component. The name will appear in the Designer Detail Pane and in choice lists when you are prompted for objects in Designer. The name may not contain the characters: \ / : ? " < > . | Names are case insensitive (i.e. MyObject is the same as myobject).

Name:

3270SampleComponent

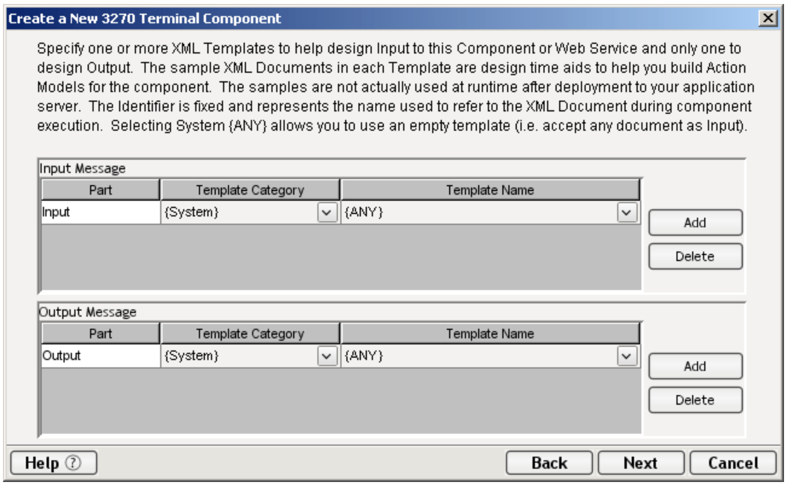
Description:

Purpose:
Input:
Output:
Remarks:

3 Enter a **Name** for the new 3270 Component.

4 Optionally, type **Description** text.

5 Click **Next**. The XML Input/Output Property Info panel of the New 3270 Component Wizard appears.


Create a New 3270 Terminal Component

Specify one or more XML Templates to help design Input to this Component or Web Service and only one to design Output. The sample XML Documents in each Template are design time aids to help you build Action Models for the component. The samples are not actually used at runtime after deployment to your application server. The Identifier is fixed and represents the name used to refer to the XML Document during component execution. Selecting System {ANY} allows you to use an empty template (i.e. accept any document as Input).

Input Message

Part	Template Category	Template Name
Input	{System}	{ANY}

Output Message

Part	Template Category	Template Name
Output	{System}	{ANY}

6 Specify the Input and Output templates as follows.

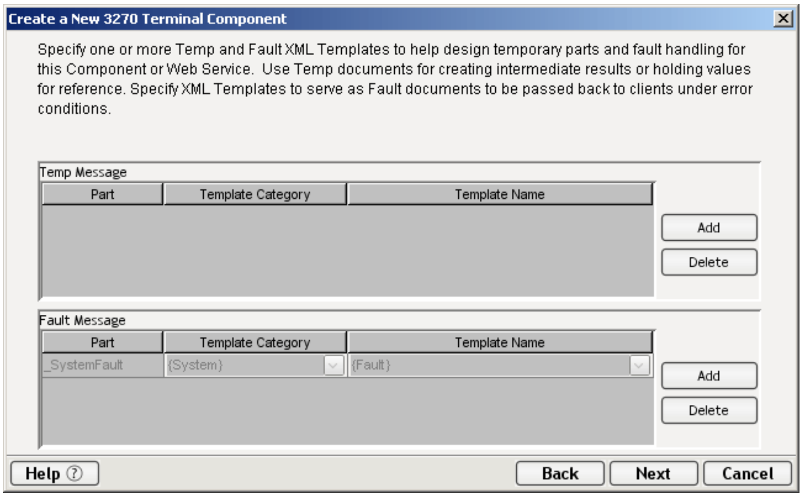
- Type in a name for the template under **Part** if you wish the name to appear in the DOM as something other than “Input”.
- Select a **Template Category** if it is different than the default category.

- Select a **Template Name** from the list of XML templates in the selected **Template Category**.
- To add additional input XML templates, click **Add** and choose a **Template Category** and **Template Name** for each.
- To remove an input XML template, select an entry and click **Delete**.

7 Select an XML template for use as an Output DOM using the same steps outlined above.

NOTE: You can specify an input or output XML template that contains no structure by selecting {System}{ANY} as the Input or Output template. For more information, see "Creating an Output DOM without Using a Template" in the User's Guide.

8 Click Next. The XML Temp/Fault Template Info panel of the New 3270 Component Wizard appears.


Create a New 3270 Terminal Component

Specify one or more Temp and Fault XML Templates to help design temporary parts and fault handling for this Component or Web Service. Use Temp documents for creating intermediate results or holding values for reference. Specify XML Templates to serve as Fault documents to be passed back to clients under error conditions.

Temp Message

Part	Template Category	Template Name

Fault Message

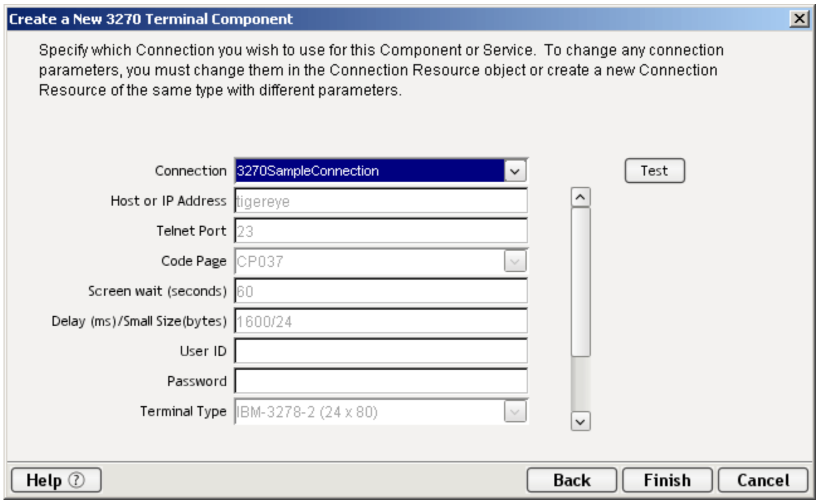
Part	Template Category	Template Name
_SystemFault	{System}	{Fault}

9 If desired, specify a template to be used as a scratchpad under the "Temp Message" pane of the dialog window. This can be useful if you need a place to hold values that will only be used temporarily during the execution of your component or are for reference only. Specify the templates as indicated in Step 6 above.

10 Under the "Fault Message" pane, select an XML template to be used to pass back to clients when an error condition occurs.

As above, to add additional temp or fault XML templates, click **Add** and choose a Template Category and Template Name for each. Repeat as many times as desired. To *remove* an XML template, select an entry and click **Delete**.

11 Click **Next.** The Connection Info panel of the Create a New 3270 Component Wizard appears.



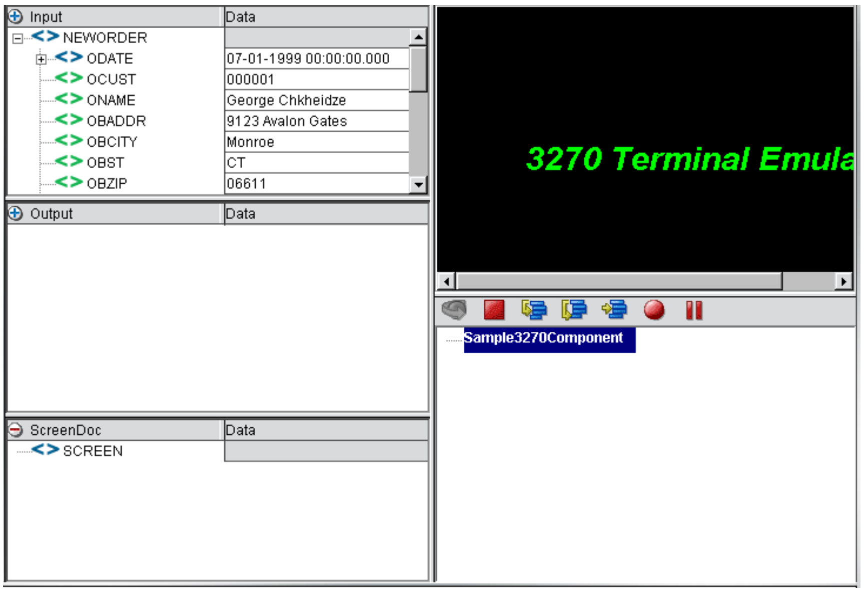
Create a New 3270 Terminal Component

Specify which Connection you wish to use for this Component or Service. To change any connection parameters, you must change them in the Connection Resource object or create a new Connection Resource of the same type with different parameters.

Connection	3270SampleConnection
Host or IP Address	tigereye
Telnet Port	23
Code Page	CP037
Screen wait (seconds)	60
Delay (ms)/Small Size(bytes)	1600/24
User ID	
Password	
Terminal Type	IBM-3278-2 (24 x 80)

12 Select a **Connection** name from the pull down list. For more information on the 3270 Connection, see “Creating a Connection Resource” in Section 2 of this Guide.

13 Click **Finish**. The component is created and the 3270 Component Editor appears.



Input	Data
NEWORDER	
ODATE	07-01-1999 00:00:00.000
OCUST	000001
ONAME	George Chkheidze
OBADDR	9123 Avalon Gates
OBCITY	Monroe
OBST	CT
OBZIP	06611

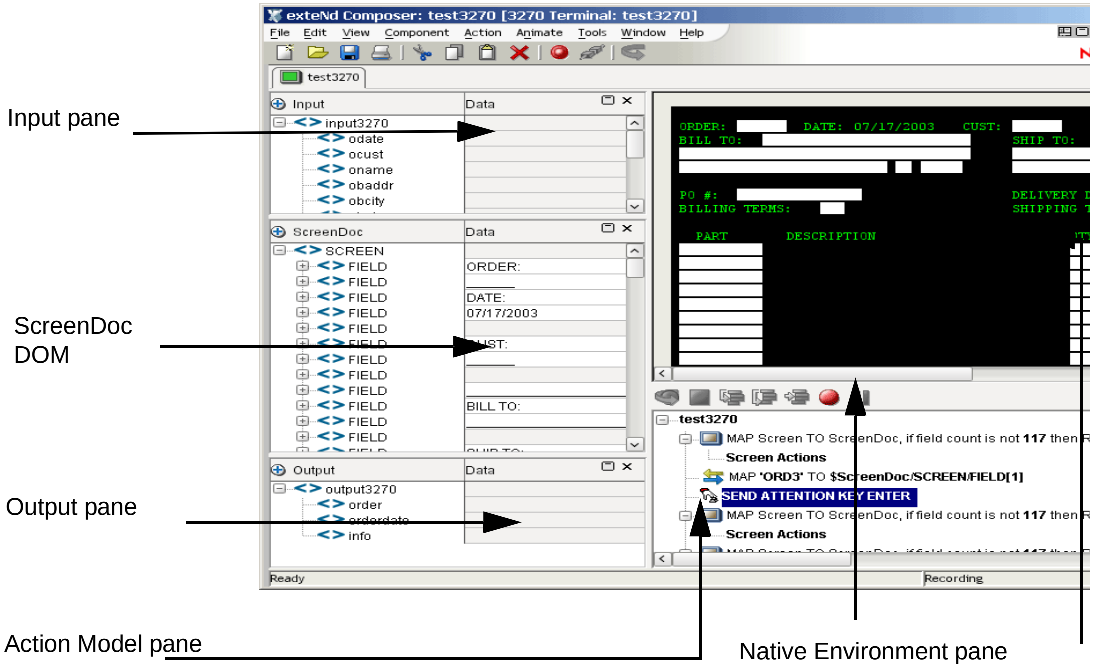
Output	Data

ScreenDoc	Data
SCREEN	

About the 3270 Component Editor Window

The 3270 Component Editor includes all the functionality of the XML Map Component Editor. It contains mapping panes for Input and Output XML documents as well as an Action pane.

There are two key differences, however. The first is that the 3270 Component Editor also includes a Native Environment pane common to all Connects. It contains a 3270 or emulator and appears black until you select the Record or connect button in the 3270 Component Editor window. Pressing the Record button establishes a 3270 emulation session inside the Native Environment pane with the host specified in the connection used by the 3270 component. The second difference is the addition of a panel containing only an XML DOM called ScreenDoc to the component editor window. This DOM presents an XML document representation of each screen received from the host and is available for reference and creating mapping actions within the component, and is available in the expression builder, allowing the user to easily reference a screen field. For those who want to create a quick HTML representation of the 3270 screen, they can use the output of the ScreenDoc DOM and apply a style sheet to create an HTML representation of the screen.


Input pane
ScreenDoc DOM
Output pane
Action Model pane
Native Environment pane
exteNd Composer: test3270 [3270 Terminal: test3270]
File Edit View Component Action Animate Tools Window Help
test3270
Input
Data
input3270
odate
ocust
oname
obaddr
obcity
ScreenDoc
SCREEN
FIELD
ORDER:
DATE:
07/17/2003
BILL TO:
Output
output3270
order
orderdate
info
ORDER: DATE: 07/17/2003 CUST:
BILL TO:
SHIP TO:
PO #:
BILLING TERMS:
DELIVERY
SHIPPING
PART
DESCRIPTION
test3270
MAP Screen TO ScreenDoc, if field count is not 117 then
Screen Actions
MAP 'ORD3' TO $ScreenDoc/SCREEN/FIELD[1]
SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 117 then
Screen Actions
Ready
Recording

About the 3270 Native Environment Pane

The 3270 Native Environment pane provides 3270 emulation of your mainframe environment. From this pane, you can perform the following:

- Map data from an Input XML document (or other available DOM) and use it as input for a 3270 screen field. For example, you could drag a SKU number from an input DOM into the part field of a 3270 screen, which would then query the mainframe and return data associated with that part number, such as description and price.
- Map the data from the returned 3270 screen and put it into an Output XML document (or other available DOM, e.g., Temp, MyDom, etc.).
- Map header and detail information (such as an invoice with multiple line items) from an XML document into the transaction accessed in the native environment pane using a special Multi-Row action.
- Map header and detail information (such as customer name and transaction history) from the transaction in the native environment pane into an XML document.

The transaction functionality of the Native Environment pane is identical to that of a 3270 terminal or terminal emulator.

About 3270 Keyboard Support

The 3270 Native Environment pane supports the use of several attention identifier or AID keys: Enter, Clear, PA1-3, and PF1-24. The function for each attention key varies, depending on the host application. These keys are mapped to the PC keyboard as follows:

Table 3-1

3270 Key	**PC Key**
Enter	Enter
Clear	ESC
PF1 through PF12	F1 through F12
PF13 through PF24	Shift F1 through Shift F12
PA1 through PA3	Ctrl F1 through Ctrl F3

You can either use the keys directly from the keyboard as you create a 3270 component or you can use a keypad tool bar available from the View menu.

About the ScreenDoc DOM

The ScreenDoc DOM is an XML document representation of the current screen received through the terminal data stream in the Native Environment pane. All Mapping actions to and from the screen display (including drag and drop) actually reference elements in the ScreenDoc DOM. This provides you with the advantage of being able to see and reference your familiar application screens while at the same time working with them as XML documents.

What it does

The 3270 component communicates with the host environment via the block mode terminal data stream in an asynchronous fashion. A block of data essentially represents a screen. The host sends a screen block that is displayed in the component. The screen is edited by the user (and ultimately by the component you create) and the modified screen is sent back to the host for processing after you press an AID key.

How it works

During the recording mode, each time a screen block of data is received by the component, four things happen simultaneously:

- The new screen is displayed in the Native Environment pane.
- A Map Screen action appears in the Action Model. The Map Screen action is where you will add, change, and delete actions for this particular screen. Each time a new screen block is received as you build the component, a new Map Screen action is created.
- The Map Screen action calculates and records how many TDS fields were received for this particular screen. This information is used for validation purposes the next time the component is run.
- The ScreenDoc DOM is refreshed with a new DOM that reflects the screen just received. Block mode terminals send their data as a stream of fields. These fields are defined using screen creation utilities in the host environment (such as BMS in CICS).

The fields are represented in the ScreenDoc DOM in the order they appear on the screen starting from the upper left corner of the screen, moving across to the right, then down a row starting again at the left and so on until the entire 80x24 screen area is covered. Depending on how the original screen layout was defined, there can be many fields. Some FIELDs are text labels on the screen and usually have display attributes of protected (prot) and bright (brt). Some FIELDs are for data entry and have an attribute of unprotected (unprot). Other FIELDs contain data but are hidden from display with an attribute of dark (drk). Finally there are special screen fields for implementing tabbing features on the screen, which are protected from entering data and called bypass fields with an attribute bypass.

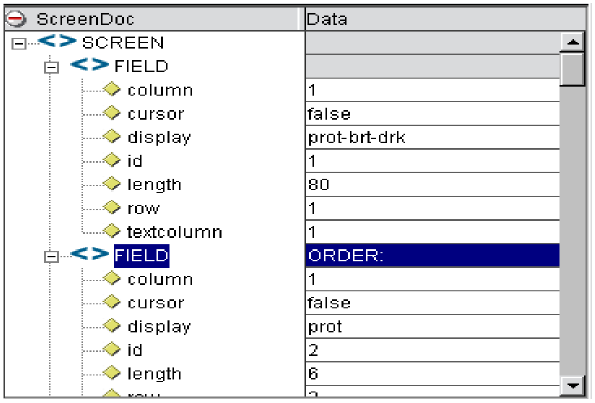
The end result of listing all the TDS fields is that the ScreenDoc DOM can be quite large. Its use is primarily intended for finding hidden fields, verifying fields and their locations by their attributes, and in cases where it is convenient, mapping from the ScreenDoc DOM to the Output DOM using Composer’s drag and drop features.

NOTE: Normally it is much quicker and more efficient to map directly to and from the Native Environment pane using drag and drop instead of mapping to the ScreenDoc DOM.

Each field in the TDS is represented in the ScreenDoc DOM as an element titled FIELD. The ScreenDoc DOM contains as many FIELD elements as there are fields in the TDS. The element displays any data defined for the field. Note that screen fields are used for both field labels and data-entry fields.

About the ScreenDoc DOM

Whenever a Map Screen action executes, a new screen is displayed in the Native Environment Pane (NEP). Each time a screen displays in NEP, an XML representation of the screen is created in the ScreenDoc DOM. (To make this DOM visible, you may need to select a View from the menu bar, then the Window Layout choice next pick the XML Layout Tab, and finally move ScreenDoc to the Visible list and click OK. To help locate the screen cursor position programatically, each ScreenDoc Field returned by the TDS will have an attribute named “cursor.” The attribute value of the cursor will be “false” for all the Fields except one whose value will be “true.” To determine which Field has the cursor, you can write an expression to check the cursor attribute and return the Field’s id attribute as shown below.



ScreenDoc	Data
SCREEN	
FIELD	
column	1
cursor	false
display	prot-brt-drk
id	1
length	80
row	1
textcolumn	1
FIELD	ORDER:
column	1
cursor	false
display	prot
id	2
length	6

In the sample list of ScreenDoc Fields in the above graphic, the XPath location ScreenDoc.XPath (“SCREEN/FIELD[@cursor= ‘true’]/@id”) would return the number “4” indicating that Field 4 is the current cursor location.

Each element also displays the following attributes for the field:

Table 3-2

Attribute	**Meaning**
Column	The screen column (1-80) where the field begins starting from the left
Cursor	Attribute value which can be “false” for all fields except one must be “true”
Display	Display attributes defined for the field in the TDS (prot = protected, brt = bright, unprot= unprotected, dark = dark, bypass = bypass)
Id	An absolute sequence number representing the fields position in the TDS
Length	The fixed length of the field (1-80)
Row	The screen row (1-24) where the field begins starting from the top

About 3270-Specific Menu Bar Items

View Menu

Keypad Tool Bar - This selection displays a keypad tool bar for the 3270 terminal keys. It is docked to the top of the native environment pane. You can drag the keypad from this location, changing it to a floating window. When you close the floating window, the keypad returns to the native environment pane. To remove the keypad from sight, select **View**, then **Keypad Tool bar** from the menu bar.

Component Menu

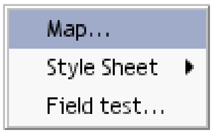
Style Sheet - This selection displays the style sheet editor dialog. It contains a few pre-defined style sheets that appear as resources on the main Composer window.

Start/Stop Recording - This selection manages the automatic creation of actions as you interact with a screen transaction. **Start** will create actions as you interact with the screen and **Stop** will end action creation.

Connect/Disconnect - This selection allows you to control the connection to the host. When you are recording or animating, a connection is automatically established, so you are not required to use this button at that time. However, this button is useful if you want to establish a connection simply to navigate through the 3270 environment when you are not recording or animating.

About 3270-Specific Context Menu Items

The 3270 Connector also includes context menu items that are specific to this Connector. To view the context menu, place your cursor in a field in the native environment pane and click the right mouse button. The context menu appears as shown.

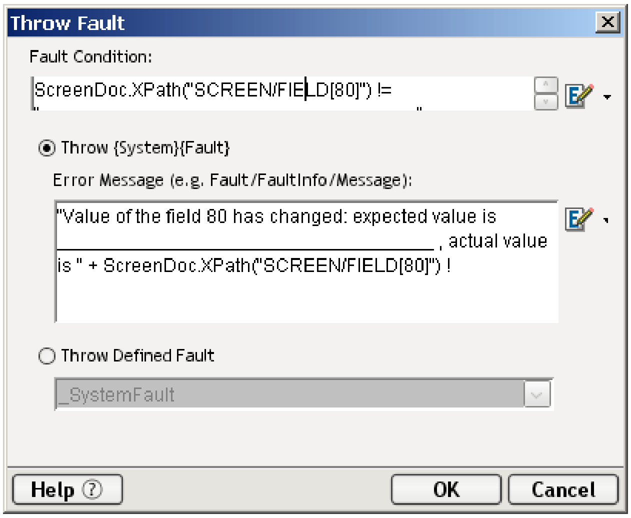


The function of the context menu items are as follows:

Map - Allows you to create a Map action. This is done by highlighting a source in the Input DOM, then highlighting a source in the Native Environment Pane. As a result, a map action is created. Also you can click the RMB in the Native Environment pane, select Map, and an action is created.

Style Sheet - Allows you to change the appearance of the native environment pane by applying a different Style Sheet.

Field Test - Allows you to create a Throw Fault action for the selected field. An expression for the fault condition will automatically be entered for you, based on the field you clicked when you brought up the context menu. Select **Throw System Fault** to define a new error message. (You have access to the ECMAScript expression builder.) Alternatively, you may select **Throw Defined Fault** to select a previously defined Fault Document from the dropdown list.



Fault Condition:

ScreenDoc.XPath("SCREEN/FIELD[80]") !=

[● Throw {System}{Fault}]

Error Message (e.g. Fault/FaultInfo/Message):

"Value of the field 80 has changed: expected value is ____________________________ , actual value is " + ScreenDoc.XPath("SCREEN/FIELD[80]") !

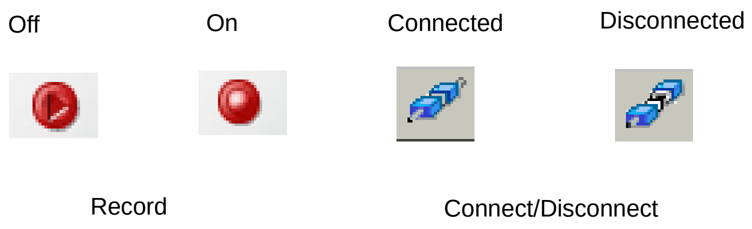
Throw Defined Fault

_SystemFault

For more information on how to use Throw Fault Actions, refer to Chapter 9 of the *Composer User's Guide*.

About 3270-Specific Buttons

The 3270 Connect includes two additional tools on the component editor tool bar: the **Record** button and the **Connect/Disconnect** button. The **Record** button enables the automatic creation of actions in the Action Model as you interact with screen transactions. The **Connect/Disconnect** button controls your connection to the mainframe. They appear as shown.



4 Performing 3270 Actions

About Actions

An *action* is similar to a programming statement in that it takes input in the form of parameters and performs specific tasks. Please see the chapters in the *Composer User's Guide* devoted to Actions.

Within the 3270 Component Editor, a set of instructions for processing XML documents or communicating with non-XML data sources is created as part of an *Action Model*. The Action Model performs all data mapping, data transformation, data transfer between mainframes and XML documents, and data transfer within components and services.

An Action Model is made up of a list of actions. All actions within an Action Model work together. As an example, one Action Model might read invoice data from a disk, retrieve data from a mainframe inventory database, map the result to a temporary XML document, make a conversion, and map the converted data to an output XML document.

The Action Model mentioned above would be composed of several actions. These actions would:

- Open an invoice document and perform a 3270 command to retrieve invoice data from a mainframe database
- Map the result to a temporary XML document
- Convert a numeric code using a Code Table and map the result to an Output XML document.

About 3270-Specific Actions

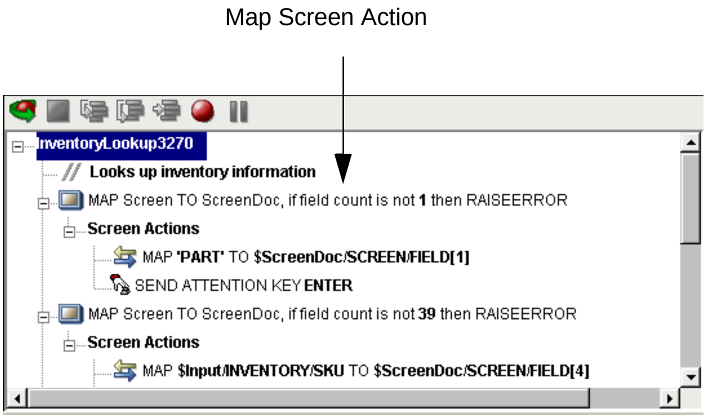
The 3270 Connector includes several actions that are specific to the 3270 and are not included with Composer.

Table 4-1

3270 Action	Description
Map Screen	Indicates location in the Action Model to place actions relative to a specific transaction screen.
Multi Row	This action allows you to specify the mapping of many-to-many data relationships between a DOM and the 3270 screen.
Send Attention Key	This action is created automatically by pressing any Aid key. The action can be edited to change the key sent back to the host.

Map Screen

In addition to showing where in the Action Model a specific transaction screen appears, the Map Screen action is also used for error checking. When a screen is first recorded, Composer saves a count of how many fields are in the screen. This count is compared later during execution for error checking to ensure the actions recorded will return the proper information. The Map Screen action appears in the Action Model as shown.



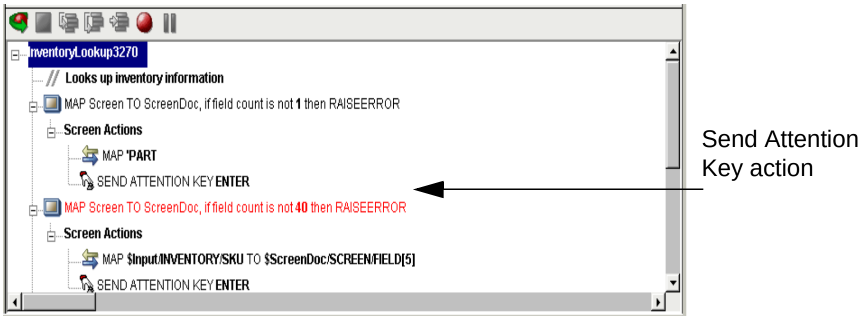
```
InventoryLookup3270
  // Looks up inventory information
  MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
    Screen Actions
      MAP 'PART' TO $ScreenDoc/SCREEN/FIELD[1]
      SEND ATTENTION KEY ENTER
  MAP Screen TO ScreenDoc, if field count is not 39 then RAISEERROR
    Screen Actions
      MAP $Input/INVENTORY/SKU TO $ScreenDoc/SCREEN/FIELD[4]
```

Multi Row

The Multi Row action can be used to input data from an XML document to a 3270 screen, or to output data from a 3270 screen to an XML document. This action essentially creates repeat loops within the Action Model that map multiple rows of data automatically from one document or screen to another. The Multi Row action is discussed in detail in "About the 3270 Multi Row Wizard" on page 67.

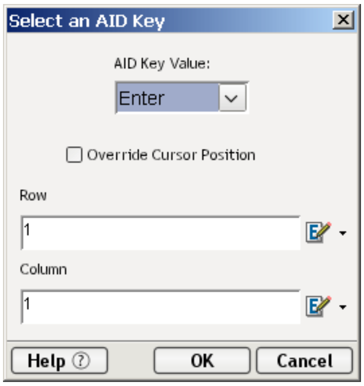
Send Attention Key

Each time you select one of the AID keys displayed in the native environment tool bar, or its corresponding keyboard key, or keys, (See "About 3270 Keyboard Support" on page 39) a Send Attention Key action is mapped in the Action Model.



```
InventoryLookup3270
  // Looks up inventory information
  MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
    Screen Actions
      MAP 'PART
      SEND ATTENTION KEY ENTER
  MAP Screen TO ScreenDoc, if field count is not 40 then RAISEERROR
    Screen Actions
      MAP $Input/INVENTORY/SKU TO $ScreenDoc/SCREEN/FIELD[5]
      SEND ATTENTION KEY ENTER
```

Double-clicking the Send Attention Key action in the Action Model displays a dialog box that allows you to edit the key. Select from the dropdown list box the Value Key. Click on the checkbox to override the cursor position. Edit the row and column if needed or click on the expression builder icon to add a calculation.



Select an AID Key

AID Key Value: Enter

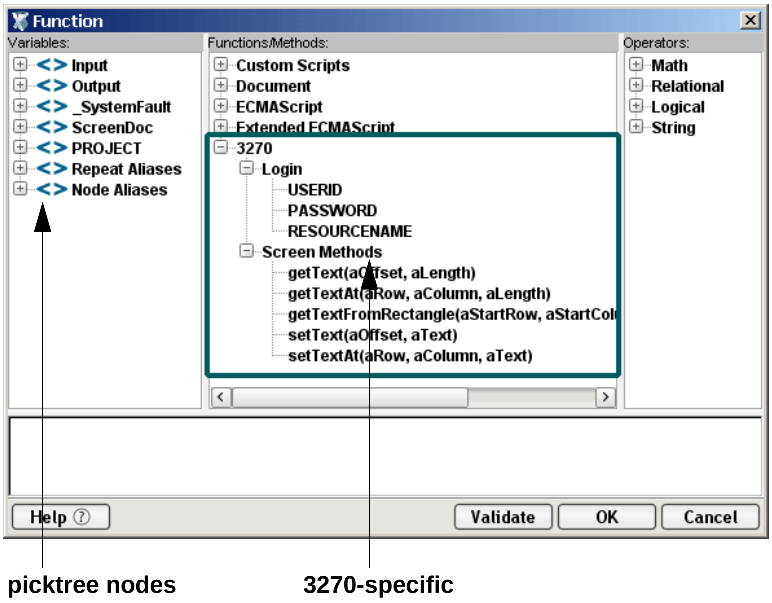
☐ Override Cursor Position

Row: 1

Column: 1

3270 Specific Expression Builder Extensions

The Connect for TN3270 exposes a number of 3270-specific ECMAScript variables and object extensions, which are visible in Expression Builder picklists. The 3270-specific items are listed under the node labelled “3270.” There are two child nodes: Login and Screen Methods. See illustration below.



Login

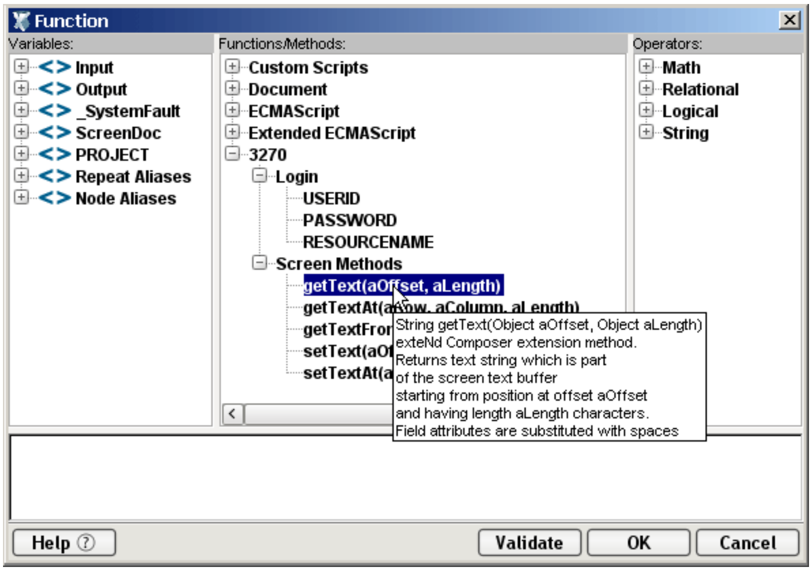
3270 Connection Resources have three global variables that are accessible from Expression Builder dialogs: USERID, PASSWORD and RESOURCENAME. The first two properties (available under the Login node of the picktree) specify the User ID and Password values that may be requested by the host system when you connect. The RESOURCE NAME variable specifies the Name of the Resource as specified in the Advanced dialog of the 3270 Connection Dialog. You can map any of these variables into the terminal screen, which eliminates the need for typing user, password and resource information explicitly in a map action.

NOTE: You can also create a Send Buffer action where the XPath source is defined as $PASSWORD.

Screen Methods

When an Expression Builder window is accessed from a Map or Function action in the 3270 Component, the picklists at the top of the window expose special 3270-specific ECMAScript extensions, consisting of various methods of the Screen object and predefined escape sequences corresponding to various “special keys” on the virtual terminal’s keyboard.

Hover-help is available if you let the mouse loiter over a given picktree item. (See illustration.)



String getText(Object aOffset, Object aLength)
exteNd Composer extension method.
Returns text string which is part
of the screen text buffer
starting from position at offset aOffset
and having length aLength characters.
Field attributes are substituted with spaces

In addition, you can obtain more complete online help by clicking Help in the lower left corner of the dialog.

The Screen object offers methods with the following names, signatures, and usage conventions:

String getText(*nOffset, nLength*)

This method returns the string of characters (of length `nLength`) that occurs in the Screen object at the byte offset given by `nOffset`. Note that the offset is one-based, not zero-based. Thus, to obtain *all* of a 24 x 80 screen as an ECMAScript String, you would do:

```
var wholeScreen = Screen.getText( 1, 24 * 80 );
```

Any attempt to obtain character data beyond the bounds of the screen buffer will result in an exception. For example, the following call will fail:

```
var wholeScreen = Screen.getText( 1, 1 + 24 * 80 ); // ERROR!
```

String getTextAt(*nRow, nColumn, nLength*)

This method returns an ECMAScript String that represents the sequence of characters (of length `nLength`) in the current screen starting at the row and column position specified. Note that `nRow` and `nColumn` are one-based, not zero-based. *A zero value for either of these parameters will cause an exception.*

To obtain all of row 20 of a 24x80 screen, you would do:

```
var myRow = Screen.getTextAt( 20, 1, 80 );
```

The `getTextAt()` technique is used internally in drag-and-drop Map actions involving screen selections.

String getTextFromRectangle(*nStartRow, nStartColumn, nEndRow, nEndColumn*)

This method returns a single String consisting of substrings (one per row) comprising all the characters within the bounding box defined by the top left and bottom right row/column coordinates specified as parameters. So for example, in 24x80 mode, you could obtain the upper left quarter of the screen by doing:

```
var topLeftQuadrant =
Screen.getTextFromRectangle(1,1,12,40);
```

The `getTextFromRectangle()` method is used internally in drag-and-drop Map actions involving rectangular screen selection regions created using the Shift-selection method.

Note that the string returned by this method contains newline (\u000a) delimiters between substrings. That is, there will be one newline at the end of each row's worth of data. The overall length of the returned string will thus be the number of rows times the number of columns, plus the number of rows. For example, `Screen.getTextFromRectangle(1,1,4,4).length` will equal 20.

void setText(*nOffset, aText*)

The `setText()` method allows you to send data to a field in the screen programmatically, beginning at a byte offset given by `nOffset`. Note that the offset is one-based, not zero-based. If the text string is longer than the available field, text is inserted up to the length of the current field. The following example places a phone number string in the second row and the 12th column (This would be an offset of 92, in a screen with 80 columns per row):

```
var myPhone = "(203) 225-1800";

if (Screen.getPrompt().indexOf("Phone") != -1)

     Screen.setText(92, myPhone ); // send string
```

void setTextAt(*nRow, nColumn, aText*)

The `setTextAt()` method allows you to send data to a field in the screen programmatically, specifying a Row and Column location to start. If the text string is longer than the available field, text is inserted up to the length of the current field. The following example places a phone number in the 8th row, 20th column:

```
var myPhone = "(203) 225-1800";

if (Screen.getPrompt().indexOf("Phone") != -1)

     Screen.setTextAt( 8, 20, myPhone ); // send string
```

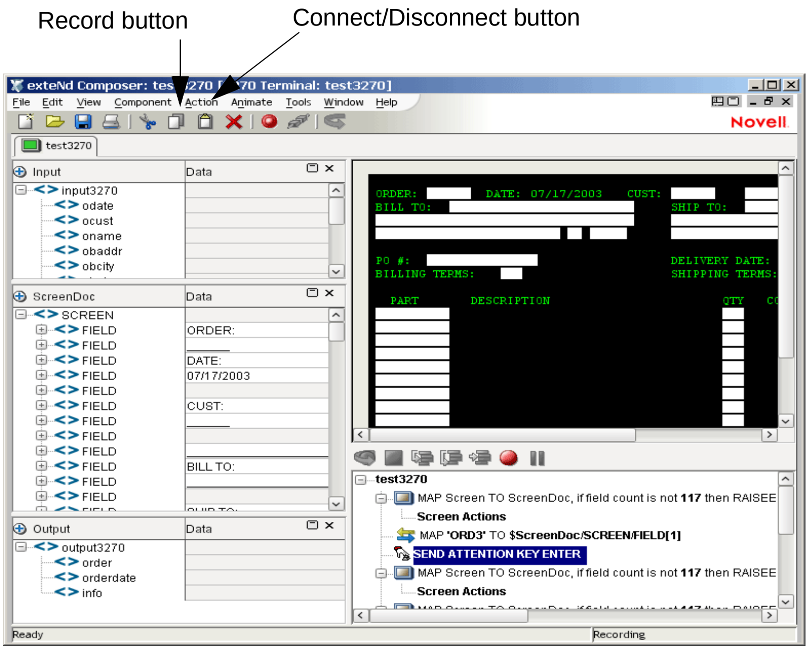
Recording a 3270 Session

The 3270 component differs from other components because a major portion of the Action Model is built for you automatically. This happens by interacting with a live session from the host in the Native Environment pane and Composer recording your activity as a set of actions in the Action Model. In other components, you must manually create actions in the Action Model, which then perform mapping, transformation, and transferring tasks. When you create a 3270 component, you essentially record the requests and responses to and from the mainframe, which are entered as actions in the Action Model pane. For example, when you select the **Enter** button in the 3270 Native Environment pane, the Action Model records the action as shown in the graphic below. In addition, you can add actions to the Action Model just the same as other components.

NOTE: You should be familiar with 3270 commands and the application you are interfacing into your XML integration project in order to successfully build a 3270 component.

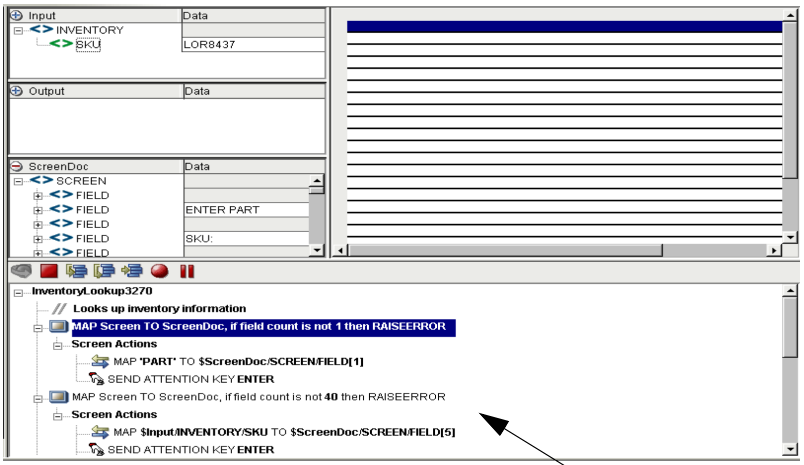
To record a 3270 session:

1 Create a 3270 component per the instructions in “Before Creating a 3270 Component” on page 33. Once created, the new 3270 component appears in the 3270 Component Editor window.


Record button
Connect/Disconnect button
exteNd Composer: test3270 [3270 Terminal: test3270]
File Edit View Component Action Animate Tools Window Help
Novell
test3270
Input
Data
input3270
odate
ocust
oname
obaddr
obcity
ScreenDoc
SCREEN
FIELD
ORDER:
DATE:
07/17/2003
CUST:
BILL TO:
Output
output3270
order
orderdate
info
ORDER: DATE: 07/17/2003 CUST:
BILL TO: SHIP TO:
PO #: DELIVERY DATE:
BILLING TERMS: SHIPPING TERMS:
PART DESCRIPTION QTY
test3270
MAP Screen TO ScreenDoc, if field count is not 117 then RAISEE
Screen Actions
MAP 'ORD3' TO $ScreenDoc/SCREEN/FIELD[1]
SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 117 then RAISEE
Screen Actions
Ready
Recording

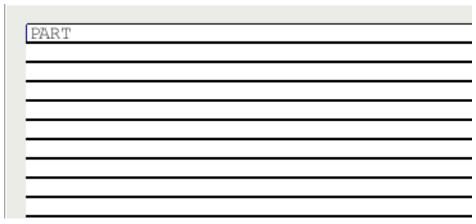
NOTE: In addition to the buttons found on the XML Map Component Editor tool bar, the 3270 Component Editor includes a **Record** and **Connect/Disconnect** button as shown above.

2 Click the **Record** button. An input screen appears in the Native Environment pane and a “Map Screen” action is recorded in the Action Model pane.


Input | Data
INVENTORY
SKU | LOR8437
Output | Data
ScreenDoc | Data
SCREEN
FIELD
FIELD | ENTER PART
FIELD
FIELD | SKU:
FIELD

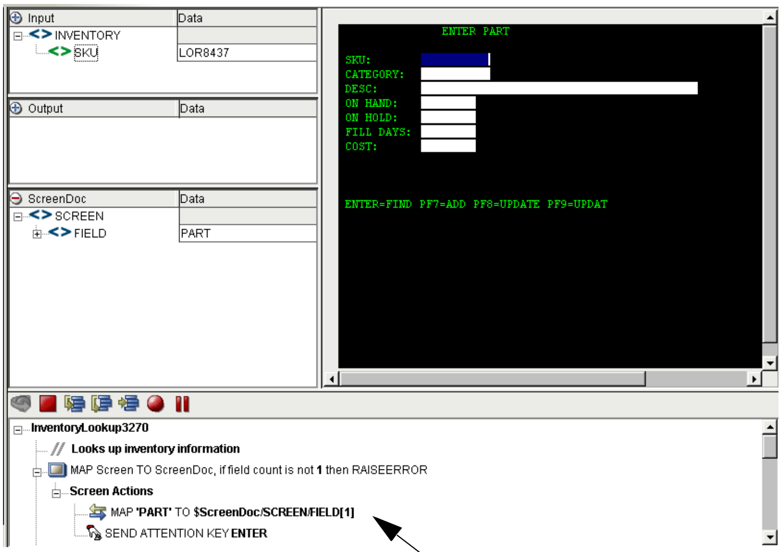
InventoryLookup3270
// Looks up inventory information
MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
Screen Actions
MAP 'PART' TO $ScreenDoc/SCREEN/FIELD[1]
SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 40 then RAISEERROR
Screen Actions
MAP $Input/INVENTORY/SKU TO $ScreenDoc/SCREEN/FIELD[5]
SEND ATTENTION KEY ENTER

3 Type “PART” in the input screen of the 3270 environment pane as shown below.


PART

NOTE: The 3270 commands are case sensitive and must be entered in ALL CAPS.

4 Click the **Enter** button in the 3270 pane. The ENTER PART screen appears in the 3270 pane and the actions are recorded in the Action Model pane under Screen Actions.


Input | Data
INVENTORY
SKU | LOR8437

Output | Data

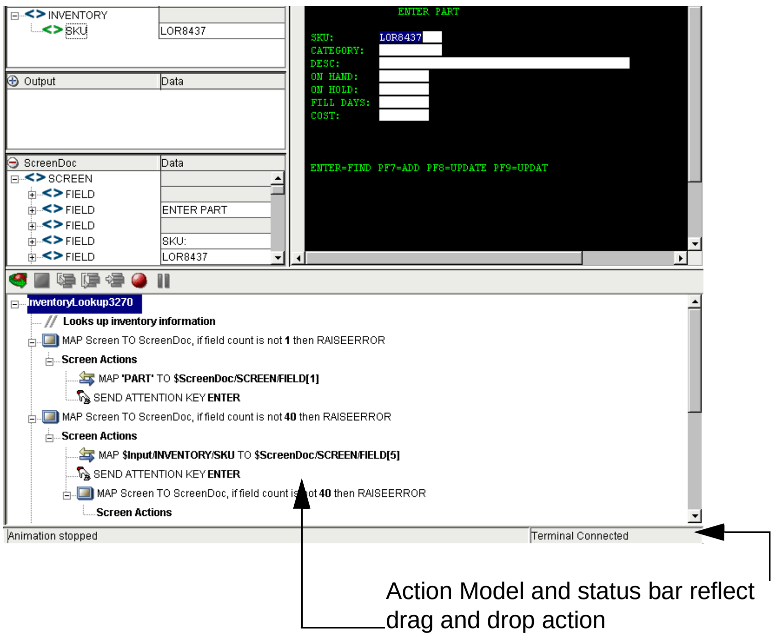
ScreenDoc | Data
SCREEN
FIELD | PART

```
                ENTER PART

SKU:
CATEGORY:
DESC:
ON HAND:
ON HOLD:
FILL DAYS:
COST:


ENTER=FIND  PF7=ADD  PF8=UPDATE  PF9=UPDAT
```

```
InventoryLookup3270
  // Looks up inventory information
  MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
    Screen Actions
      MAP 'PART' TO $ScreenDoc/SCREEN/FIELD[1]
      SEND ATTENTION KEY ENTER
```

Typing in “PART” and clicking **Enter** are recorded as screen actions.

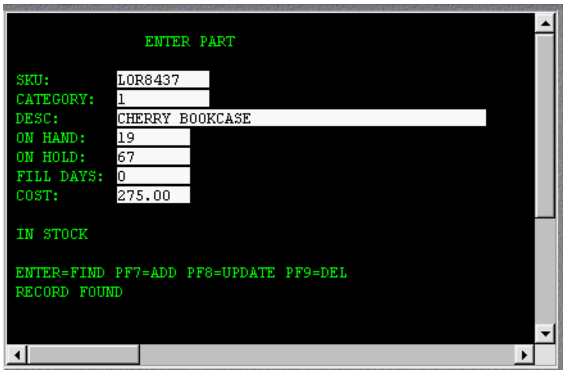
5 Drag the SKU data from the Input DOM to the SKU field in the 3270 ENTER PART screen. The action is recorded in the Action Model pane and appears in the status bar.



```
INVENTORY
  SKU                LOR8437
Output               Data
ScreenDoc            Data
SCREEN
  FIELD
  FIELD              ENTER PART
  FIELD
  FIELD              SKU:
  FIELD              LOR8437

               ENTER PART

SKU:        LOR8437
CATEGORY:
DESC:
ON HAND:
ON HOLD:
FILL DAYS:
COST:

ENTER=FIND  PF7=ADD  PF8=UPDATE  PF9=UPDAT

InventoryLookup3270
  // Looks up inventory information
  MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
    Screen Actions
      MAP 'PART' TO $ScreenDoc/SCREEN/FIELD[1]
      SEND ATTENTION KEY ENTER
  MAP Screen TO ScreenDoc, if field count is not 40 then RAISEERROR
    Screen Actions
      MAP $Input/INVENTORY/SKU TO $ScreenDoc/SCREEN/FIELD[5]
      SEND ATTENTION KEY ENTER
      MAP Screen TO ScreenDoc, if field count is not 40 then RAISEERROR
        Screen Actions
Animation stopped                          Terminal Connected
```

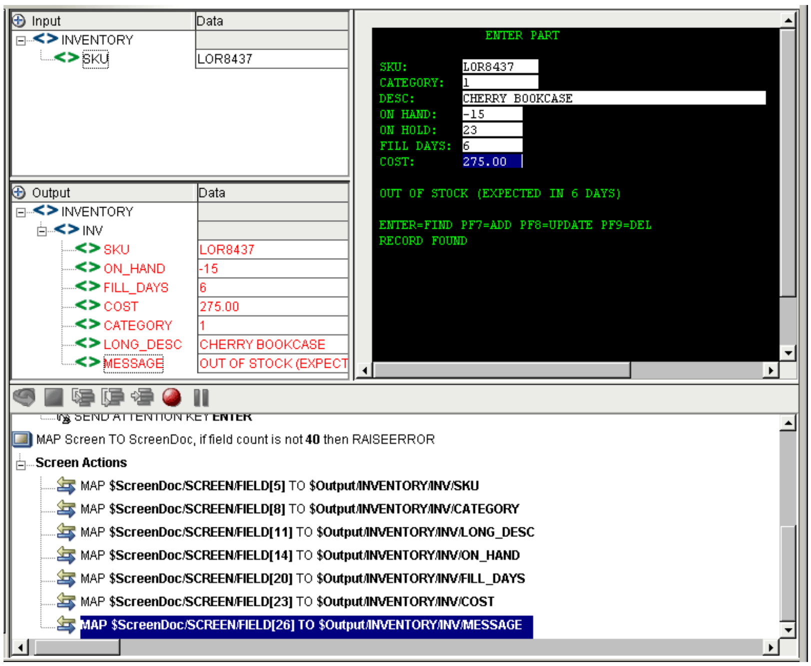
NOTE: You can also use the Map Action to map the Input SKU to the ENTER PART screen SKU field; however, dragging and dropping is much quicker and easier. For more information on the Map Action feature, see the *exteNd Composer User's Guide*.

6 Click the **Enter** button in the 3270 pane. The 3270 ENTER PART screen is populated with the SKU's associated data.



```
               ENTER PART

SKU:        LOR8437
CATEGORY:   1
DESC:       CHERRY BOOKCASE
ON HAND:    19
ON HOLD:    67
FILL DAYS:  0
COST:       275.00

IN STOCK

ENTER=FIND  PF7=ADD  PF8=UPDATE  PF9=DEL
RECORD FOUND
```

7 Drag and drop an element from the ENTER PART screen to the Output DOM, for example, the SKU number. The data you drag and drop appears in red in the Output DOM.



Input	Data
INVENTORY	
SKU	LOR8437

```
                    ENTER PART

SKU:           LOR8437
CATEGORY:      1
DESC:          CHERRY BOOKCASE
ON HAND:       -15
ON HOLD:       23
FILL DAYS:     6
COST:          275.00

OUT OF STOCK (EXPECTED IN 6 DAYS)

ENTER=FIND PF7=ADD PF8=UPDATE PF9=DEL
RECORD FOUND
```

Output	Data
INVENTORY	
INV	
SKU	LOR8437
ON_HAND	-15
FILL_DAYS	6
COST	275.00
CATEGORY	1
LONG_DESC	CHERRY BOOKCASE
MESSAGE	OUT OF STOCK (EXPECT

```
SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 40 then RAISEERROR
Screen Actions
    MAP $ScreenDoc/SCREEN/FIELD[5] TO $Output/INVENTORY/INV/SKU
    MAP $ScreenDoc/SCREEN/FIELD[8] TO $Output/INVENTORY/INV/CATEGORY
    MAP $ScreenDoc/SCREEN/FIELD[11] TO $Output/INVENTORY/INV/LONG_DESC
    MAP $ScreenDoc/SCREEN/FIELD[14] TO $Output/INVENTORY/INV/ON_HAND
    MAP $ScreenDoc/SCREEN/FIELD[20] TO $Output/INVENTORY/INV/FILL_DAYS
    MAP $ScreenDoc/SCREEN/FIELD[23] TO $Output/INVENTORY/INV/COST
    MAP $ScreenDoc/SCREEN/FIELD[26] TO $Output/INVENTORY/INV/MESSAGE
```

8 Continue to drag and drop data elements from the ENTER PART screen to the desired field in the Output Part until complete. Each time an element is dragged from the ENTER PART screen to the Output Part an action is recorded in the Action Model pane.

9 Click the **Save** icon on the toolbar.

Editing a Previously Recorded Action Model

You will undoubtedly encounter times when you need to edit a previously recorded action model. Unlike editing other components, editing a 3270 component requires extra attention. When a 3270 component executes, it plays back a sequence of actions that expect certain screens and data to appear in order to work properly. So when editing a component you must be careful not to make the action model sequence inconsistent with the mainframe transaction execution sequence you recorded earlier.

In general, to ensure successful edits, the following recommendations apply:

- Do not cut or copy “Map Screen” action blocks and paste them into other locations in your action model.
- Carefully check and edit individual Map actions that interact with the screen after copying and pasting them within an action model.

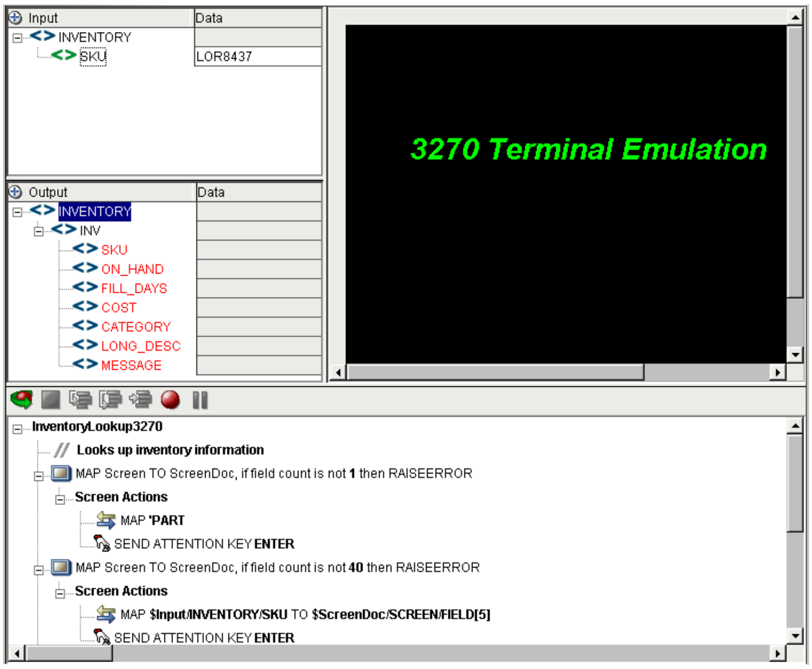
- Use Composer's drag and drop features to add new Map actions that interact with the screen. Animate to the line of interest in your Action Model, pause animation, and turn on Record mode. This will prevent your Action Model from getting out of sync with the proper ScreenDoc DOM and /or fields within a specific ScreenDoc DOM.
- Don't delete any Multi-Row related lines (or any actions for that matter) in your Action Model during animation. This may prevent your component from functioning properly.

Changing an Existing Action

The following procedure will explain how to change an existing action in a previously recorded session.

➤ **To Change an existing action in a previously recorded Action Model:**

1 Open the component that includes the previously recorded Action Model that you'd like to edit. The component appears in the 3270 Component Editor window.

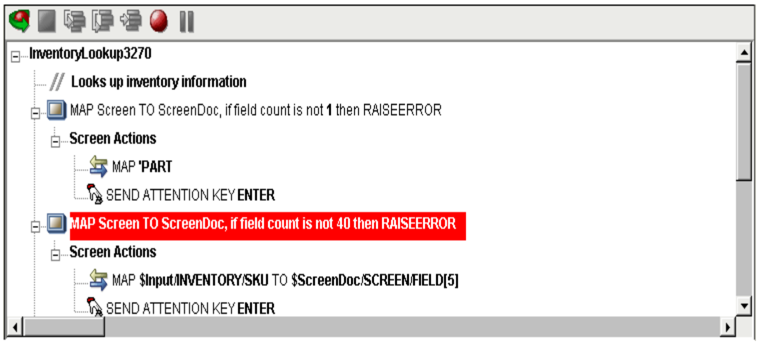


Input	Data
INVENTORY	
SKU	LOR8437

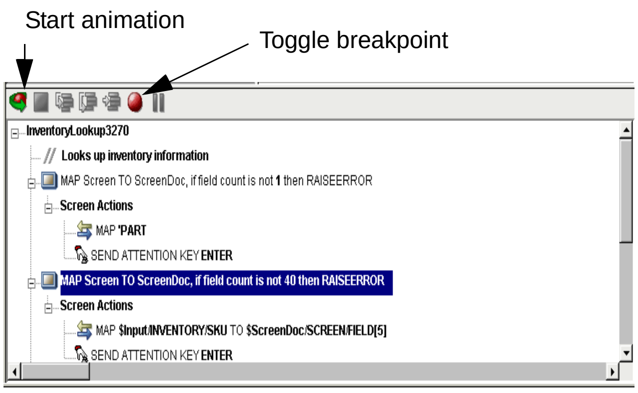
Output	Data
INVENTORY	
INV	
SKU	
ON_HAND	
FILL_DAYS	
COST	
CATEGORY	
LONG_DESC	
MESSAGE	

```
InventoryLookup3270
    // Looks up inventory information
    MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
        Screen Actions
            MAP 'PART
            SEND ATTENTION KEY ENTER
    MAP Screen TO ScreenDoc, if field count is not 40 then RAISEERROR
        Screen Actions
            MAP $Input/INVENTORY/SKU TO $ScreenDoc/SCREEN/FIELD[5]
            SEND ATTENTION KEY ENTER
```

2 Navigate to the action in the Action Model where you'd like to make your edit and highlight the action.

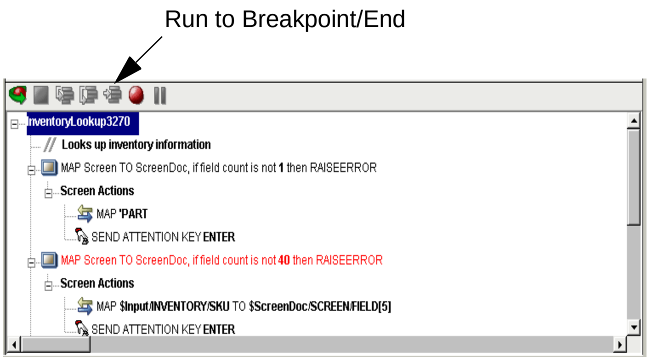

InventoryLookup3270
// Looks up inventory information
MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
Screen Actions
MAP 'PART
SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 40 then RAISEERROR
Screen Actions
MAP $Input/INVENTORY/SKU TO $ScreenDoc/SCREEN/FIELD[5]
SEND ATTENTION KEY ENTER

3 Click the **Toggle Breakpoint** button. The highlighted action becomes red.



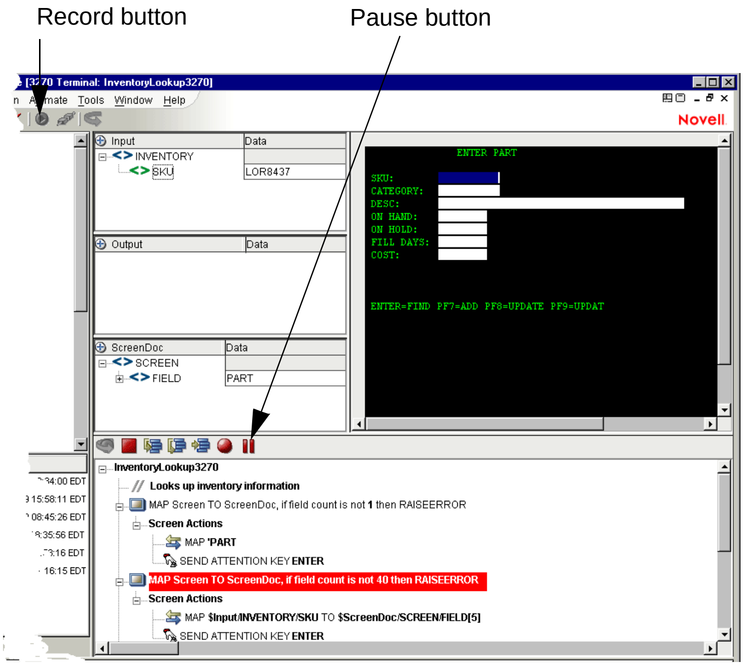
InventoryLookup3270
// Looks up inventory information
MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
Screen Actions
MAP 'PART
SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 40 then RAISEERROR
Screen Actions
MAP $Input/INVENTORY/SKU TO $ScreenDoc/SCREEN/FIELD[5]
SEND ATTENTION KEY ENTER

4 Click the **Start Animation** button. The animation tools become active.



InventoryLookup3270
// Looks up inventory information
MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
Screen Actions
MAP 'PART
SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 40 then RAISEERROR
Screen Actions
MAP $Input/INVENTORY/SKU TO $ScreenDoc/SCREEN/FIELD[5]
SEND ATTENTION KEY ENTER

5 Click the **Step to Breakpoint/End** button. The Action Model executes all of the actions from the beginning to the breakpoint you set in step 3 above and appears as shown.



```
ENTER PART

SKU:
CATEGORY:
DESC:
ON HAND:
ON HOLD:
FILL DAYS:
COST:

ENTER=FIND PF7=ADD PF8=UPDATE PF9=UPDAT
```

Input	Data
INVENTORY	
SKU	LOR8437

ScreenDoc	Data
SCREEN	
FIELD	PART

```
InventoryLookup3270
  // Looks up inventory information
  MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
    Screen Actions
      MAP 'PART
      SEND ATTENTION KEY ENTER
  MAP Screen TO ScreenDoc, if field count is not 40 then RAISEERROR
    Screen Actions
      MAP $Input/INVENTORY/SKU TO $ScreenDoc/SCREEN/FIELD[5]
      SEND ATTENTION KEY ENTER
```

6 Click the **Pause** button on the animation tool bar.

7 In the Component Editor tool bar, click the **Record** button.

8 Execute any additional actions that you'd like to add to the Action Model.

9 Select **File**, then **Save**, or click the **Save** button on the Component Editor tool bar.

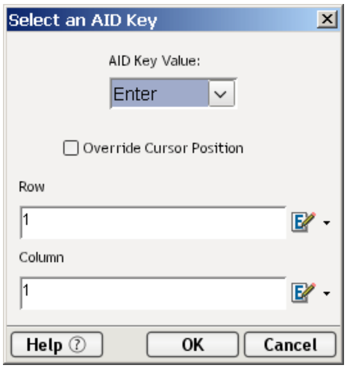
10 Follow the instructions in "Using the Animation Tools" to test your component.

Editing Attention Keys

Whenever you press **Enter** on the keyboard or click one of the many attention keys in the 3270 Native Environment pane while recording a session, an action is recorded in the Action Model. An example of this is shown below.

SEND ATTENTION KEY **ENTER**

These actions, like any other, can be deleted, moved, or copied. You may also double click an Attention Key action and edit it from a dialog box.

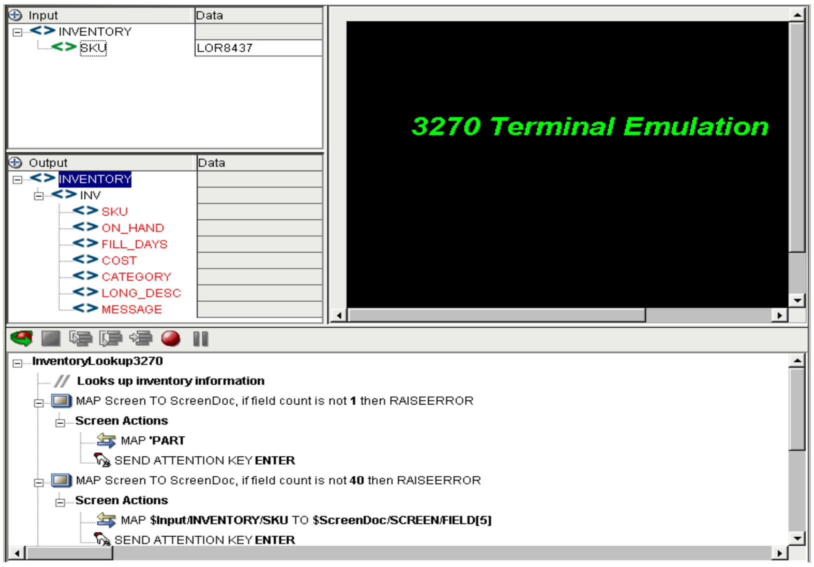


Adding a New Action

The following procedure explains how to add a new action in a previously recorded session.

➢ **To Add a Action to a previously recorded Action Model:**

1 Open the component that includes the previously recorded Action Model that you’d like to add an action in. The component appears in the 3270 Component Editor window.



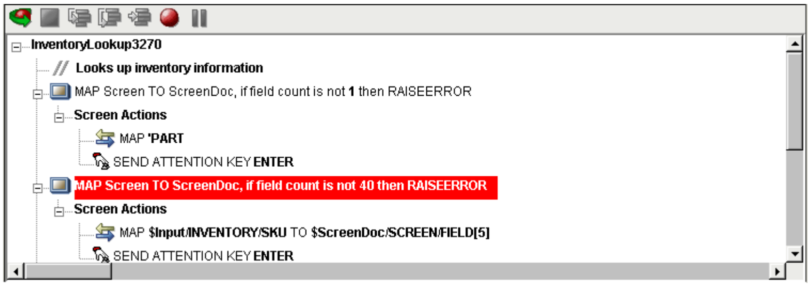
Input | Data
INVENTORY
SKU | LOR8437

Output | Data
INVENTORY
INV
SKU
ON_HAND
FILL_DAYS
COST
CATEGORY
LONG_DESC
MESSAGE

3270 Terminal Emulation

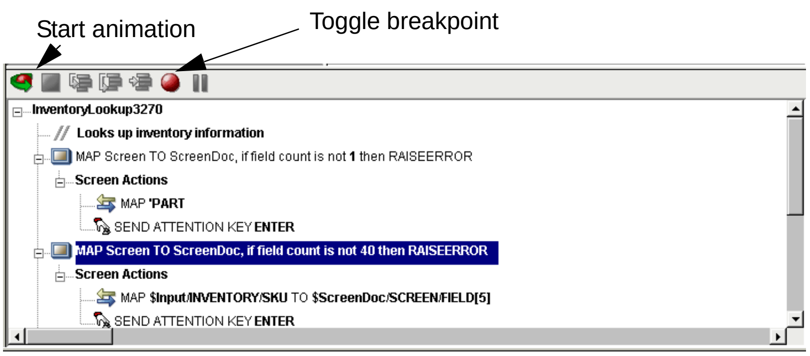
```
InventoryLookup3270
    // Looks up inventory information
    MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
        Screen Actions
            MAP 'PART
            SEND ATTENTION KEY ENTER
    MAP Screen TO ScreenDoc, if field count is not 40 then RAISEERROR
        Screen Actions
            MAP $Input/INVENTORY/SKU TO $ScreenDoc/SCREEN/FIELD[5]
            SEND ATTENTION KEY ENTER
```

2 Navigate to the action in the Action Model where you'd like to add the action and highlight the action.



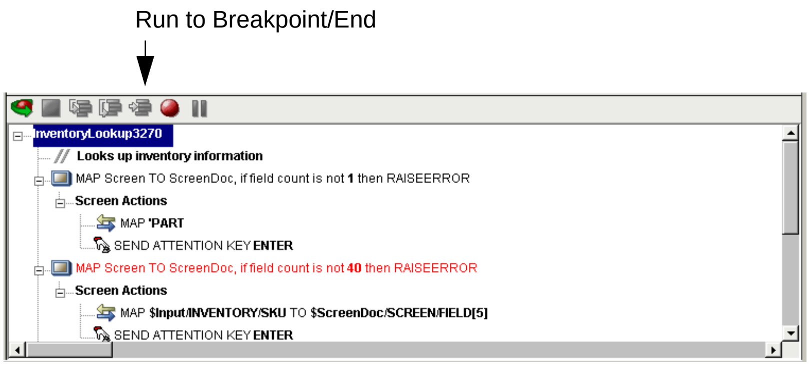
```
InventoryLookup3270
    // Looks up inventory information
    MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
        Screen Actions
            MAP 'PART
            SEND ATTENTION KEY ENTER
    MAP Screen TO ScreenDoc, if field count is not 40 then RAISEERROR
        Screen Actions
            MAP $Input/INVENTORY/SKU TO $ScreenDoc/SCREEN/FIELD[5]
            SEND ATTENTION KEY ENTER
```

3 Click the **Toggle Breakpoint** button. The highlighted action becomes red.



InventoryLookup3270
// Looks up inventory information
MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
Screen Actions
MAP 'PART
SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 40 then RAISEERROR
Screen Actions
MAP $Input/INVENTORY/SKU TO $ScreenDoc/SCREEN/FIELD[5]
SEND ATTENTION KEY ENTER

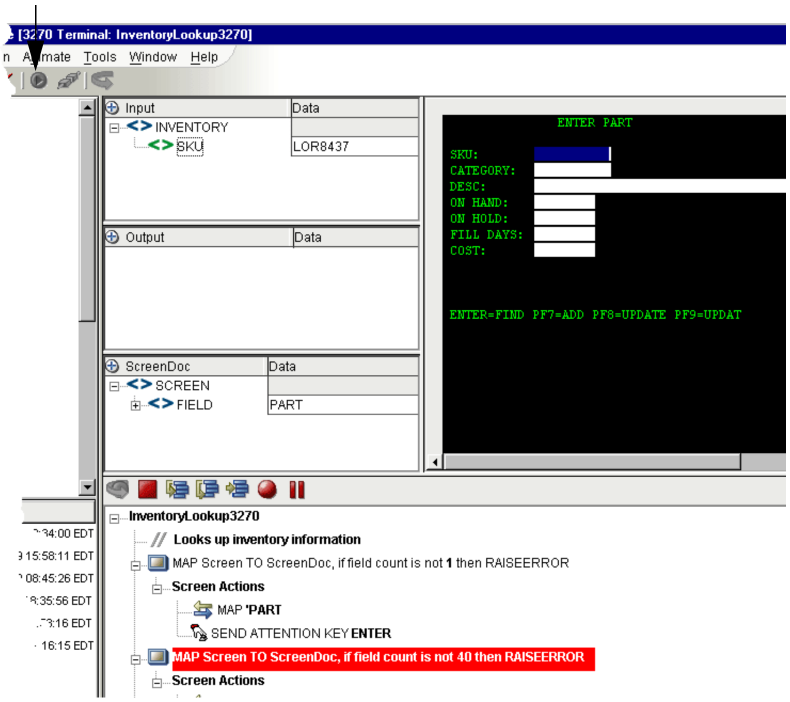
4 Click the **Start Animation** button. The animation tools become active.



InventoryLookup3270
// Looks up inventory information
MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
Screen Actions
MAP 'PART
SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 40 then RAISEERROR
Screen Actions
MAP $Input/INVENTORY/SKU TO $ScreenDoc/SCREEN/FIELD[5]
SEND ATTENTION KEY ENTER

5 Click the **Run to Breakpoint/End** button. The Action Model executes all of the actions from the beginning to the breakpoint you set in step 3 above and appears as shown.

Record button



```
ENTER PART
SKU:
CATEGORY:
DESC:
ON HAND:
ON HOLD:
FILL DAYS:
COST:
ENTER=FIND PF7=ADD PF8=UPDATE PF9=UPDAT
```

InventoryLookup3270
// Looks up inventory information
MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
Screen Actions
MAP 'PART
SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 40 then RAISEERROR
Screen Actions

6 Click the **Pause** button on the animation tool bar.

7 In the Component Editor tool bar, click the **Record** button.

8 Use Composer's drag and drop features to add new Map actions that interact with the screen. The new action will be added directly under the highlighted line.

9 Select **File**, then **Save**, or click the **Save** button on the Component Editor tool bar.

10 Follow the instructions in "Using the Animation Tools" to test your component.

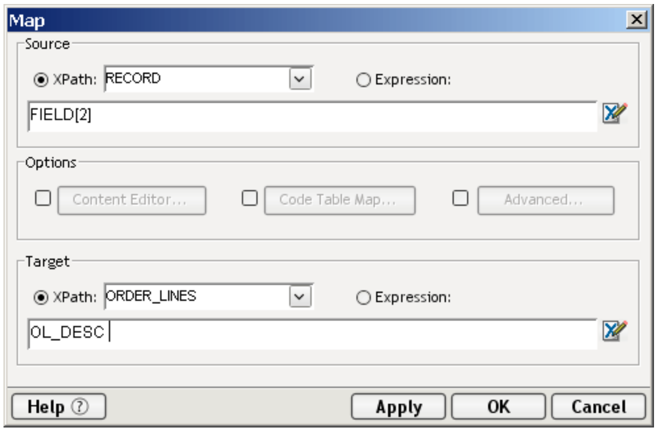
About Adding Alias Actions

If you are adding Map Actions in a loop that are alias perform the following steps:

➤ **To Add an Alias Action to a previously recorded Action Model:**

1 Open a component.

From the **Action** menu, select **New Action,** then **Map**. The Map Action dialog box displays.



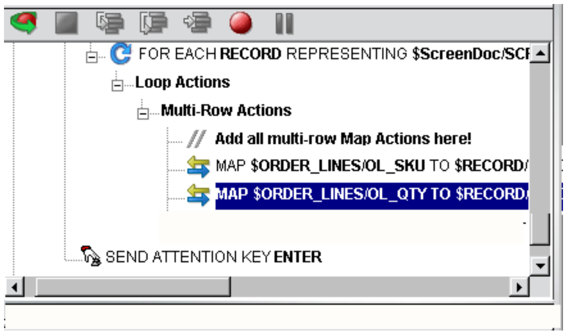
2 Select the Xpath and from the dropdown list for Source, Order_Lines is selected from the dropdown list.

3 Either type in the information, or click the **Expression Builder** button and create a new expression.

4 Create an XPath to be represented by the alias. Click from the dropdown list for the alias

5 Click **OK**.

The new action is inserted below the line you select. (New line is highlighted in the screen below to show it was inserted.

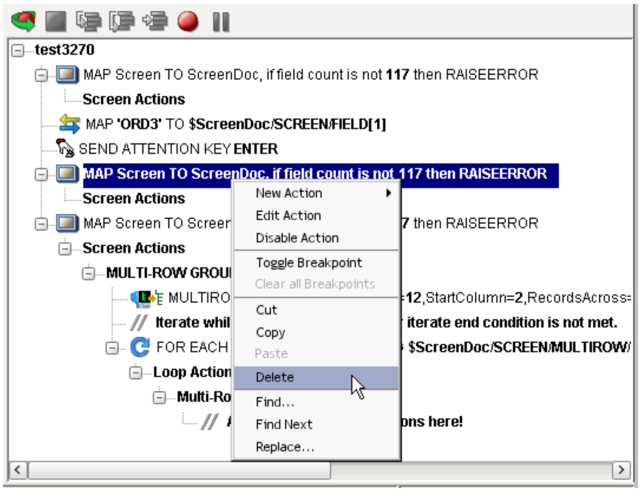

FOR EACH RECORD REPRESENTING $ScreenDoc/SCF
Loop Actions
Multi-Row Actions
// Add all multi-row Map Actions here!
MAP $ORDER_LINES/OL_SKU TO $RECORD/
MAP $ORDER_LINES/OL_QTY TO $RECORD/
SEND ATTENTION KEY ENTER

Deleting an Action

The following procedure explains how to delete an action in a previously recorded session

➤ To Delete an Action to a previously recorded Action Model:

1. Highlight the action line that you want to delete.
2. Click on the RMB and select Delete from the menu. You may also highlight the line and press the Delete button on your keyboard.



```
test3270
  MAP Screen TO ScreenDoc, if field count is not 117 then RAISEERROR
    Screen Actions
  MAP 'ORD3' TO $ScreenDoc/SCREEN/FIELD[1]
  SEND ATTENTION KEY ENTER
  MAP Screen TO ScreenDoc, if field count is not 117 then RAISEERROR
    Screen Actions
  MAP Screen TO Screer                  7 then RAISEERROR
    Screen Actions
      MULTI-ROW GROUI
        MULTIRO                         12,StartColumn=2,RecordsAcross=
        Iterate whil                    iterate end condition is not met.
        FOR EACH                        $ScreenDoc/SCREEN/MULTIROW/
          Loop Action
            Multi-Ro
              //                        ons here!
```

About the 3270 Multi Row Wizard

The 3270 Connect includes drag and drop capabilities that allow you to drag data from an XML document into a 3270 or from a 3270 screen into an XML document quickly and easily; however, in cases where there are a variable number of rows and possibly multiple screens of data, using the drag and drop method is not effective. To resolve this issue, the 3270 Connector includes a *Multi Row Wizard* that allows you to specific the mapping of many-to-many data relationships between a DOM and the 3270 screen. The Multi Row Wizard, which appears as an option in the **Action>New Action** menu of a 3270 component, can be used to input data from an XML document to the repeating area of a 3270 screen, or to output the repeating data from a 3270 screen to an XML document. The Multi Row Wizard essentially creates repeat loops within the Action Model that map multiple rows of data automatically from one document or screen to another until complete, regardless of the number of rows or number of screen pages.

About the Multi Row Samples in this Document

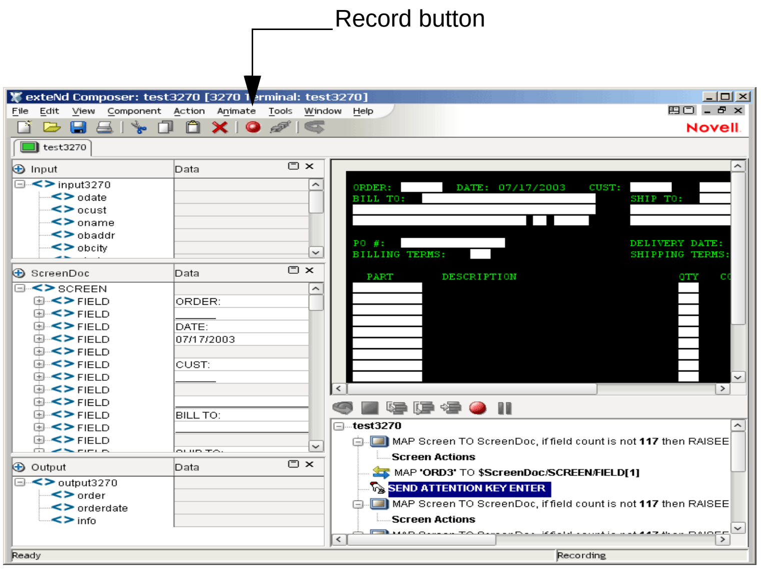
There are many ways the Multi Row Wizard can be used when inputting or outputting multiple rows of data to and from XML documents and 3270 screens, and not every scenario can be addressed in this document. Two basic sample components are provided here that should give you a clear understanding of how the Multi Row Wizard is generally used.

The first procedure shows you a component that uses the Multi Row Wizard to **input** multiple elements of data from an XML document to a 3270 screen. The second procedure shows a component that uses the Multi Row Wizard to **output** multiple rows of data from a 3270 screen to an Output DOM.

The procedures in this section are strictly for demonstration purposes. Unlike the Composer tutorial, theses procedures are not intended for following along step-by-step.

➤ To prepare for using the Multi Row Wizard to input data:

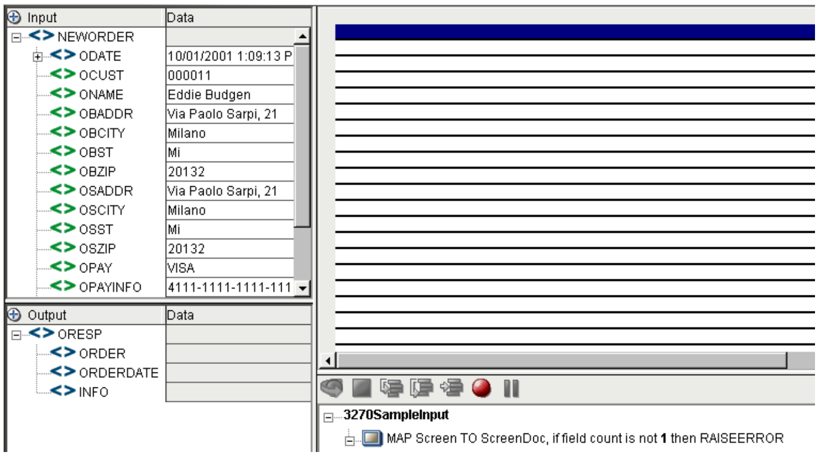
1 Create a component per the instructions in “Before Creating a 3270 Component” on page 33. For this example, the component is called “3270SampleInput.” The component is shown below in the 3270 Component Editor.



```
ORDER:           DATE:  07/17/2003    CUST:
BILL TO:                                        SHIP TO:

PO #:                                           DELIVERY DATE:
BILLING TERMS:                                  SHIPPING TERMS:

  PART          DESCRIPTION                       QTY
```

test3270
MAP Screen TO ScreenDoc, if field count is not **117** then RAISEE
Screen Actions
MAP **'ORD3'** TO **$ScreenDoc/SCREEN/FIELD[1]**
SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not **117** then RAISEE
Screen Actions

2 Click the **Record** button. A blank screen appears in the Native Environment pane.



Input	Data
NEWORDER	
ODATE	10/01/2001 1:09:13 P
OCUST	000011
ONAME	Eddie Budgen
OBADDR	Via Paolo Sarpi, 21
OBCITY	Milano
OBST	Mi
OBZIP	20132
OSADDR	Via Paolo Sarpi, 21
OSCITY	Milano
OSST	Mi
OSZIP	20132
OPAY	VISA
OPAYINFO	4111-1111-1111-111

Output	Data
ORESP	
ORDER	
ORDERDATE	
INFO	

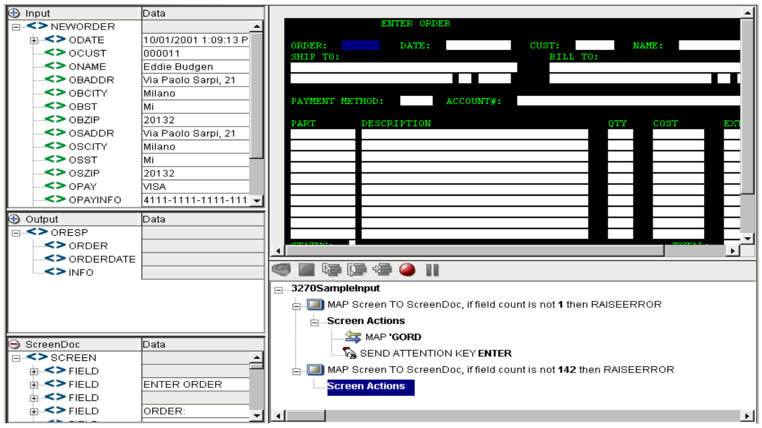
```
3270SampleInput
  MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
```

3 The user can then type in a transaction in the transaction line. This example uses “GORD” as the transaction.

4 Press **Enter** on the keyboard.

NOTE: Alternatively, you may select **Enter** on the 3270 keypad toolbar within the Native Environment pane. To view this tool bar, select **View** from the 3270 Component Editor menu bar, then **Keypad Toolbar.**

This example then displays an “ENTER ORDER” 3270 screen in the Native Environment pane.



Input	Data
NEWORDER	
ODATE	10/01/2001 1:09:13 P
OCUST	000011
ONAME	Eddie Budgen
OBADDR	Via Paolo Sarpi, 21
OBCITY	Milano
OBST	Mi
OBZIP	20132
OSADDR	Via Paolo Sarpi, 21
OSCITY	Milano
OSST	Mi
OSZIP	20132
OPAY	VISA
OPAYINFO	4111-1111-1111-111

Output	Data
ORESP	
ORDER	
ORDERDATE	
INFO	

ScreenDoc	Data
SCREEN	
FIELD	
FIELD	ENTER ORDER
FIELD	
FIELD	ORDER:

```
                 ENTER ORDER

ORDER:           DATE:            CUST:         NAME:
SHIP TO:                                 BILL TO:

PAYMENT METHOD:          ACCOUNT#:

PART         DESCRIPTION                              QTY     COST       EXT
```

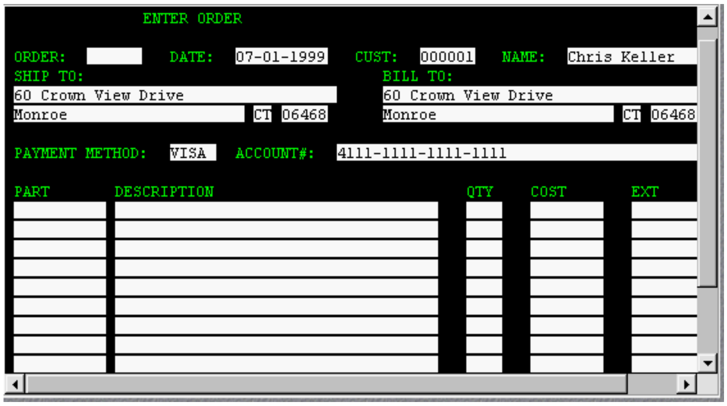
```
3270SampleInput
  MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
    Screen Actions
      MAP 'GORD
      SEND ATTENTION KEY ENTER
  MAP Screen TO ScreenDoc, if field count is not 142 then RAISEERROR
    Screen Actions
```

5 The user then drags and drops the applicable header data from the input DOM to the corresponding field in the 3270 ENTER ORDER screen. For this example, the following data is dragged and dropped:

Table 4-2

From: DOM Field	**To: 3270 ENTER ORDER Screen Field**
ODATE	DATE:
OCUST	CUST:
ONAME	NAME:
OBADDR	BILL TO: (First line)
OBCITY	BILL TO: (Second line)
OBST	BILL TO: (Second line, 2-character field to the right of city)
OBZIP	BILL TO: (Second line, 5-character field to the right of state)
OSADDR	SHIP TO: (First line)
OSCITY	SHIP TO: (Second line)
OSST	SHIP TO: (Second line, 2-character field to the right of city)
OSZIP	SHIP TO: (Second line, 5-character field to the right of state)
OPAY	**PAYMENT METHOD:**
OPAYINFO	**ACCOUNT #:**

The 3270 ENTER ORDER screen appears as shown.



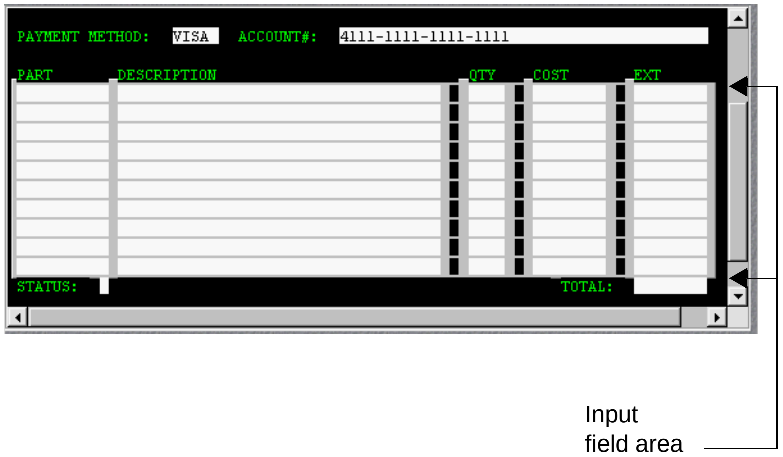
```
              ENTER ORDER

ORDER:         DATE: 07-01-1999   CUST: 000001   NAME: Chris Keller
SHIP TO:                              BILL TO:
60 Crown View Drive                   60 Crown View Drive
Monroe                     CT 06468   Monroe                   CT 06468

PAYMENT METHOD: VISA   ACCOUNT#: 4111-1111-1111-1111

PART       DESCRIPTION                          QTY    COST      EXT
```

6 Identify the area of repetitive rows that you will select for the Multi Row action.

7 For multi-row actions the user needs to define the area that contains repeating data either manually or by highlighting the area. In this example, the user drags the cursor over the line item input field area in the 3270 ENTER ORDER screen, beginning in the upper left corner and moving to the lower right corner. A gray background highlight appears over the drag marquee.

NOTE: Make sure that you start your drag process with your cursor **outside** of the first field. If you start dragging within a field, Composer assumes you are trying to move the field itself.

The 3270 input field area appears as shown.



PAYMENT METHOD: VISA ACCOUNT#: 4111-1111-1111-1111

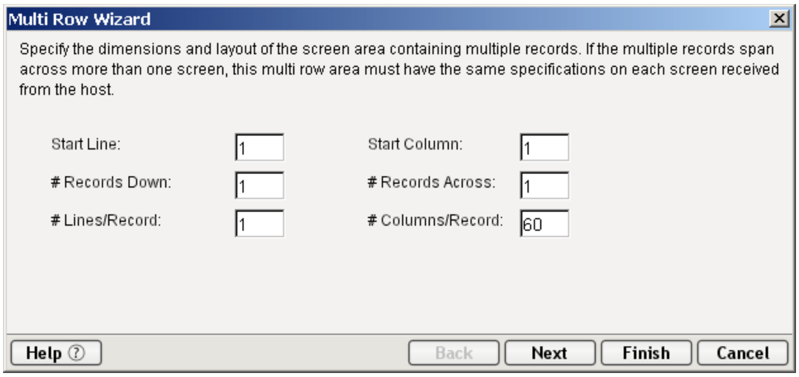
PART DESCRIPTION QTY COST EXT

STATUS: TOTAL:

➢ **To use the Multi Row Wizard to input data:**

NOTE: If you are outputting data, see discussion further below ("To use the Multi Row Wizard to output data:" on page 78 of the guide).

From the 3270 Component Editor menu, select **Action**, then **New Action**, then **Multi Row**. The Multi Row Wizard appears. The wizard automatically fills in the dialog based on the area you highlighted. See "To prepare for using the Multi Row Wizard to input data:" on page 68 for an example.



Specify the dimensions and layout of the screen area containing multiple records. If the multiple records span across more than one screen, this multi row area must have the same specifications on each screen received from the host.

Start Line: 1
Start Column: 1
Records Down: 1
Records Across: 1
Lines/Record: 1
Columns/Record: 60

If you would like to edit the fields or you did not highlight an area, you would do so as follows:

NOTE: The 3270 screen is comprised of columns and rows - the default is 24 lines x 80 columns.

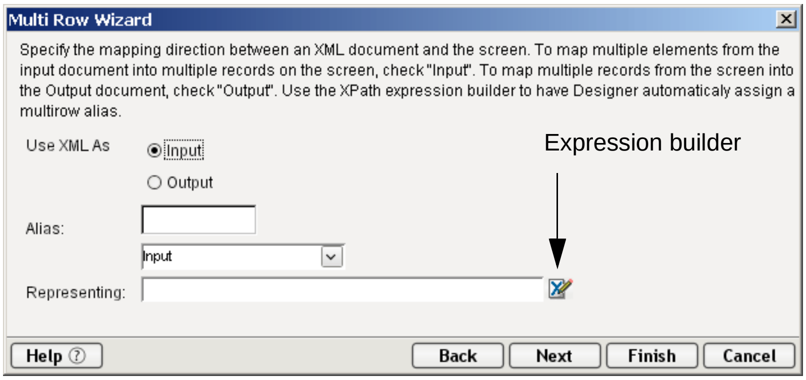
- **Start Line** - This is the first row from the top where you want the wizard to start counting rows.
- **Start Column** - This is the first column where you want the wizard to begin. Column 1 is the first column on the left.
- **# Records Down** - This is the number of records you want the wizard to include in the loop.
- **# Records Across** - It is a common practice for COBOL programmers to list records down a page and then continue to wrap records back to the top. The result is several records side-by-side down a page. You would use this field to specify how many records are listed side-by-side in a given row.
- **# Lines/Record** - If a record is longer than 80 characters, it wraps to the next line. If it exceeds 160 characters, it wraps to a third line. You need to indicate how many lines are required per record.
- **# Columns/Record** - This field indicates how many columns are included in the record, 80 being the maximum.

8 Edit the fields if desired. When satisfied with the parameters, click **Next**. The second panel of the Multi Row Wizard appears.

➢ To set up a repeat action in the Multi Row Wizard:

NOTE: If you are outputting data, see “To use the Multi Row Wizard to output data:” on page 78.

After completing the first panel of the Multi Row Wizard, continue as follows.


Multi Row Wizard

Specify the mapping direction between an XML document and the screen. To map multiple elements from the input document into multiple records on the screen, check "Input". To map multiple records from the screen into the Output document, check "Output". Use the XPath expression builder to have Designer automaticaly assign a multirow alias.

Use XML As ● Input ○ Output

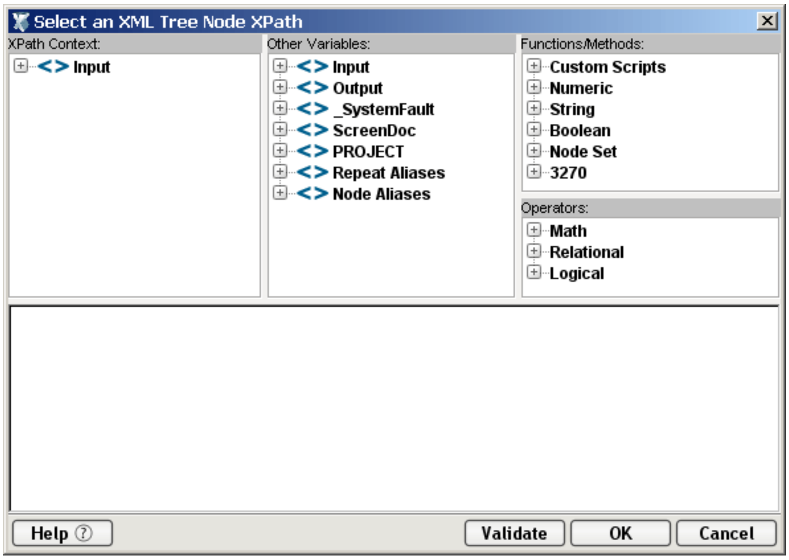
Alias:

Input

Representing:

9 Select the **Input** radio button in the **Use XML As** area. This panel is used to create a repeat action for processing the multiple elements or screen rows. Its use is similar to the basic “Repeat for Element” action available in all components.

In the **Representing** field, click the Expression builder button. The Expression builder window appears.


Select an XML Tree Node XPath

XPath Context:
- Input

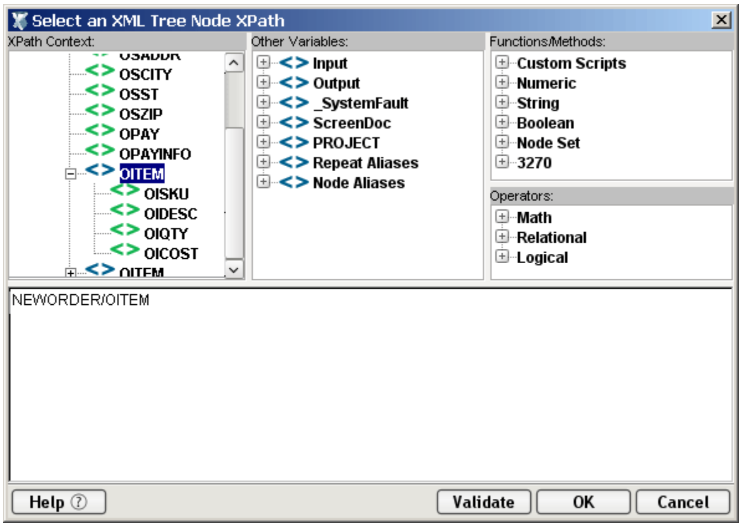
Other Variables:
- Input
- Output
- _SystemFault
- ScreenDoc
- PROJECT
- Repeat Aliases
- Node Aliases

Functions/Methods:
- Custom Scripts
- Numeric
- String
- Boolean
- Node Set
- 3270

Operators:
- Math
- Relational
- Logical

10 Expand the **Input** element in the **XPath Content** frame.

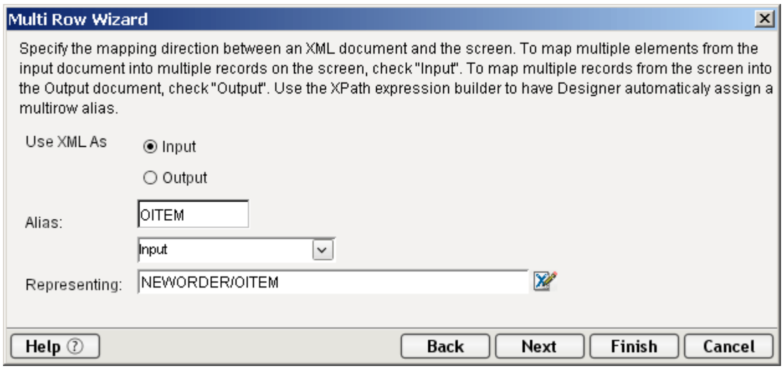
Navigate to the **OITEM** element and double click. The expression appears in the comment pane in the bottom of the window. For this example, the OITEM element is the one on which we will loop.



NEWORDER/OITEM

Click **OK**. You are returned to the second screen of the Multi Row Wizard, which now appears with the new expression in the **Representing** field.

NOTE: When using the Expression builder, Composer automatically creates an **Alias**. In this example, Composer created the alias called OITEM. For more information on Aliases, please see the *exteNd Composer User's Guide*.


Multi Row Wizard

Specify the mapping direction between an XML document and the screen. To map multiple elements from the input document into multiple records on the screen, check "Input". To map multiple records from the screen into the Output document, check "Output". Use the XPath expression builder to have Designer automaticaly assign a multirow alias.

Use XML As [✓ Input] Output

Alias: OITEM

Input

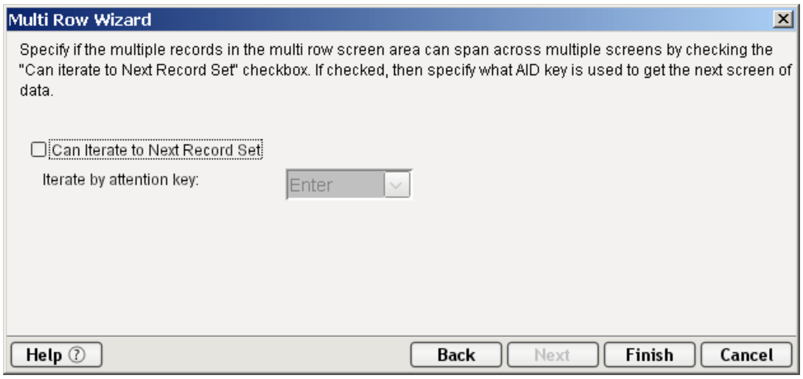
Representing: NEWORDER/OITEM

11 Click **Next**. The Iterate screen of the Multi Row Wizard appears.

➢ To iterate to the next record:

NOTE: If you are outputting data, see "To use the Multi Row Wizard to output data:" on page 78.

This screen allows you to tell the Multi Row Wizard what to do when it encounters an end of page. In the example (see "About the Multi Row Samples in this Document" on page 68), there is only one page of data, so there is no check mark in the **Can Iterate to Next Record Set** box.

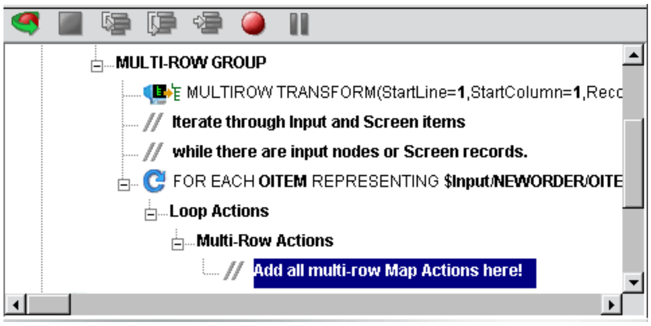

Multi Row Wizard

Specify if the multiple records in the multi row screen area can span across multiple screens by checking the "Can iterate to Next Record Set" checkbox. If checked, then specify what AID key is used to get the next screen of data.

☐ Can Iterate to Next Record Set

Iterate by attention key: Enter

Click **Finish**. The Multi Row actions you created in the wizard appear in the Action Model pane, with the **Add all multi row map actions here!** comment highlighted.



MULTI-ROW GROUP
MULTIROW TRANSFORM(StartLine=1,StartColumn=1,Recc
// Iterate through Input and Screen items
// while there are input nodes or Screen records.
FOR EACH OITEM REPRESENTING $Input/NEWORDER/OITE
Loop Actions
Multi-Row Actions
// Add all multi-row Map Actions here!

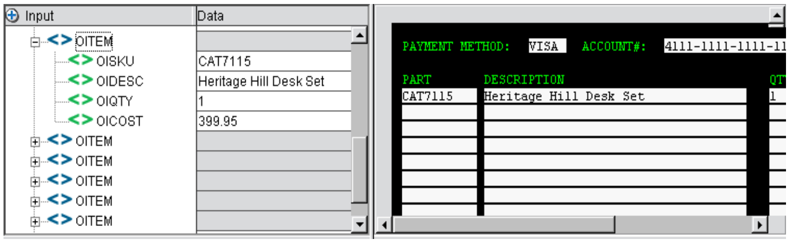
➤ To add Multi Row Actions to input data:

1. Highlight the **Add all multi row map actions here!** comment in the Action Model if not already highlighted.
2. Navigate to the first instance of OITEM in the Input Part pane.
3. Drag and drop the children of OITEM into the applicable fields in the first row of line item fields in the 3270 screen as follows:

Table 4-3

From: Input DOM	To: ENTER ORDER 3270 Screen
OISKU	PART
OIDESC	DESCRIPTION
OIQTY	QTY
OICOST	COST

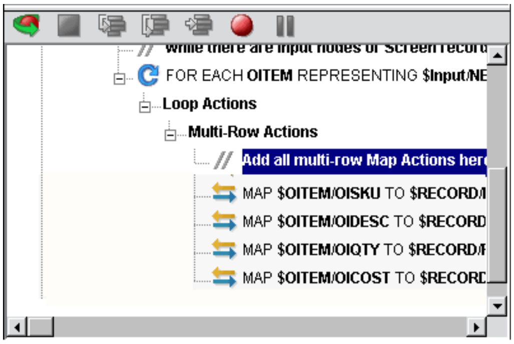
4. The Input Part and ENTER ORDER 3270 screens appear as shown.



Input	Data
OITEM	
OISKU	CAT7115
OIDESC	Heritage Hill Desk Set
OIQTY	1
OICOST	399.95
OITEM	
OITEM	
OITEM	
OITEM	
OITEM	

```
PAYMENT METHOD: VISA   ACCOUNT#: 4111-1111-1111-11

PART      DESCRIPTION                          QT
CAT7115   Heritage Hill Desk Set               1
```

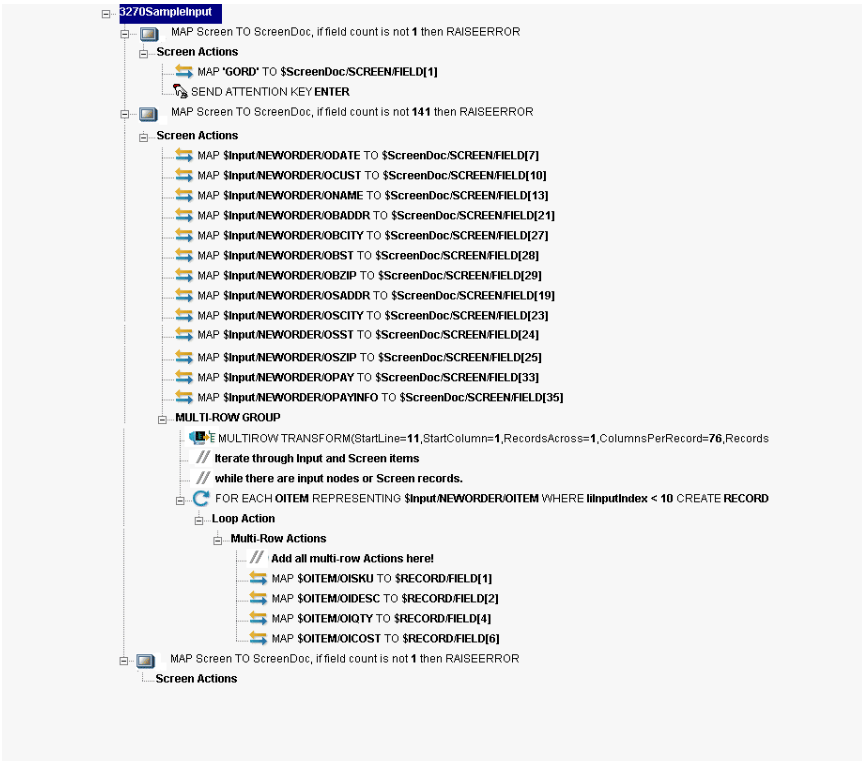
5. The Action Model pane appears as shown.



```
while there are input nodes or Screen record
FOR EACH OITEM REPRESENTING $Input/NE
  Loop Actions
    Multi-Row Actions
      // Add all multi-row Map Actions here
      MAP $OITEM/OISKU TO $RECORD/I
      MAP $OITEM/OIDESC TO $RECORD
      MAP $OITEM/OIQTY TO $RECORD/F
      MAP $OITEM/OICOST TO $RECORD
```

6 Notice that Map actions within the Multi Row actions block reference fields by their relative position in the row and not their absolute position within the screen. So the target of the first Map action is $RECORD/FIELD[1]. Using drag and drop to create your Map actions within the context of the Multi Row will create and assign these field indexes for you automatically.

The final Action Model appears as shown.



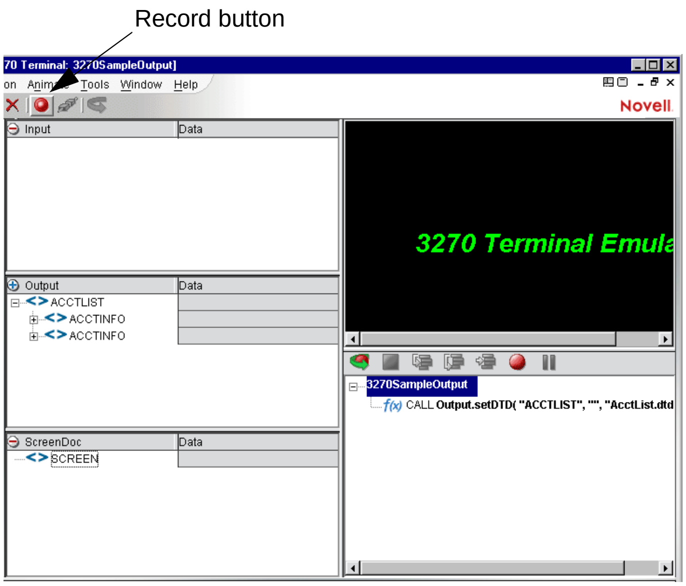
```
3270SampleInput
  MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
    Screen Actions
      MAP 'GORD' TO $ScreenDoc/SCREEN/FIELD[1]
      SEND ATTENTION KEY ENTER
  MAP Screen TO ScreenDoc, if field count is not 141 then RAISEERROR
    Screen Actions
      MAP $Input/NEWORDER/ODATE TO $ScreenDoc/SCREEN/FIELD[7]
      MAP $Input/NEWORDER/OCUST TO $ScreenDoc/SCREEN/FIELD[10]
      MAP $Input/NEWORDER/ONAME TO $ScreenDoc/SCREEN/FIELD[13]
      MAP $Input/NEWORDER/OBADDR TO $ScreenDoc/SCREEN/FIELD[21]
      MAP $Input/NEWORDER/OBCITY TO $ScreenDoc/SCREEN/FIELD[27]
      MAP $Input/NEWORDER/OBST TO $ScreenDoc/SCREEN/FIELD[28]
      MAP $Input/NEWORDER/OBZIP TO $ScreenDoc/SCREEN/FIELD[29]
      MAP $Input/NEWORDER/OSADDR TO $ScreenDoc/SCREEN/FIELD[19]
      MAP $Input/NEWORDER/OSCITY TO $ScreenDoc/SCREEN/FIELD[23]
      MAP $Input/NEWORDER/OSST TO $ScreenDoc/SCREEN/FIELD[24]
      MAP $Input/NEWORDER/OSZIP TO $ScreenDoc/SCREEN/FIELD[25]
      MAP $Input/NEWORDER/OPAY TO $ScreenDoc/SCREEN/FIELD[33]
      MAP $Input/NEWORDER/OPAYINFO TO $ScreenDoc/SCREEN/FIELD[35]
      MULTI-ROW GROUP
        E MULTIROW TRANSFORM(StartLine=11,StartColumn=1,RecordsAcross=1,ColumnsPerRecord=76,Records
        // Iterate through Input and Screen items
        // while there are input nodes or Screen records.
        FOR EACH OITEM REPRESENTING $Input/NEWORDER/OITEM WHERE liInputIndex < 10 CREATE RECORD
          Loop Action
            Multi-Row Actions
              // Add all multi-row Actions here!
              MAP $OITEM/OISKU TO $RECORD/FIELD[1]
              MAP $OITEM/OIDESC TO $RECORD/FIELD[2]
              MAP $OITEM/OIQTY TO $RECORD/FIELD[4]
              MAP $OITEM/OICOST TO $RECORD/FIELD[6]
  MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
    Screen Actions
```

7 Select **File**, then **Save** from the 3270 Component Editor menu bar, or click the **Save** button.

8 Follow the instructions in “Using the Animation Tools” on page 88 to test your component.

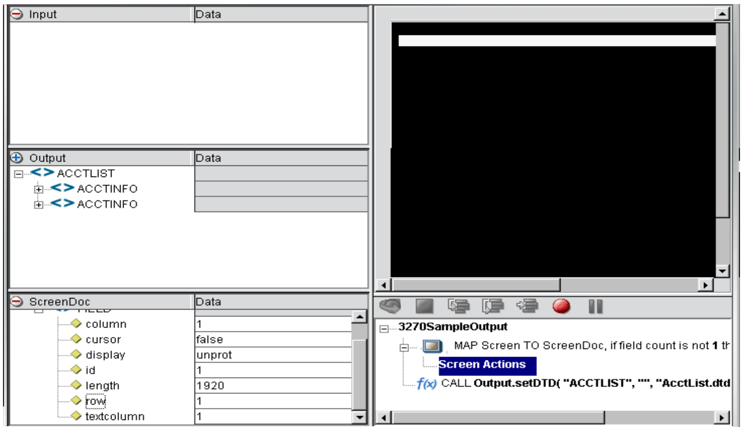
➢ To use the Multi Row Wizard to output data:

1 Create a component per the instructions in “Before Creating a 3270 Component” on page 33. For this example, a component called “3270SampleOutput” is created. The component is shown below in the 3270 Component Editor.



CALL Output.setDTD("ACCTLIST", "", "AcctList.dtd

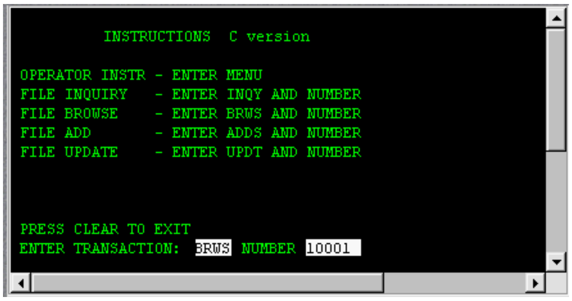
2 Click the **Record** button. A blank screen with one line appears in the Native Environment pane.



3 The user would navigate to the correct screen for this example the user would type in a transaction in the transaction line. This examples uses ‘MENU” as the transaction.

Press **Enter** on the keyboard. This example displays an “INSTRUCTIONS” 3270 screen in the Native Environment pane.

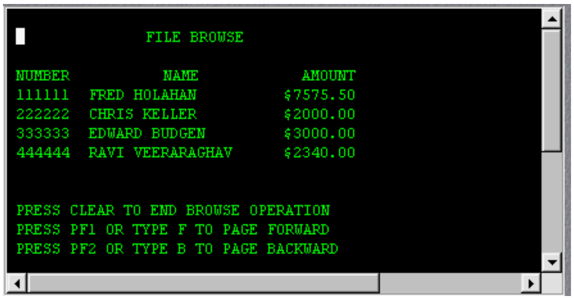
NOTE: Alternatively, you may select **Enter** on the 3270 tool bar within the Native Environment pane. To view this tool bar, select **View** from the 3270 Component Editor menu bar, then **Keypad Toolbar.**



```
         INSTRUCTIONS  C version

OPERATOR INSTR - ENTER MENU
FILE INQUIRY   - ENTER INQY AND NUMBER
FILE BROWSE    - ENTER BRWS AND NUMBER
FILE ADD       - ENTER ADDS AND NUMBER
FILE UPDATE    - ENTER UPDT AND NUMBER



PRESS CLEAR TO EXIT
ENTER TRANSACTION: BRWS NUMBER 10001
```

4 In this example, “BRWS” is entered in the **TRANSACTION** field and “10001” in the **NUMBER** field.

Press **Enter** on the keyboard. A “FILE BROWSE” 3270 screen appears.

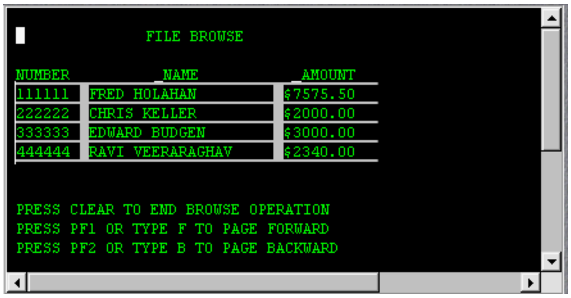


```
                FILE BROWSE

NUMBER          NAME            AMOUNT
111111  FRED HOLAHAN          $7575.50
222222  CHRIS KELLER          $2000.00
333333  EDWARD BUDGEN         $3000.00
444444  RAVI VEERARAGHAV      $2340.00


PRESS CLEAR TO END BROWSE OPERATION
PRESS PF1 OR TYPE F TO PAGE FORWARD
PRESS PF2 OR TYPE B TO PAGE BACKWARD
```

5 For multi-row actions the user needs to define the area that contains repeating data either manually or by highlighting the area. In this example, the user drags the cursor over the data area beginning in the upper right hand corner and moving to the lower left hand corner. A gray background highlight appears over the drag marquee.

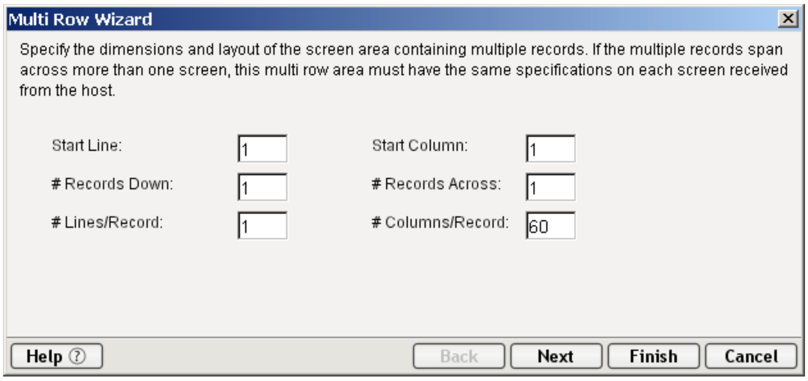
NOTE: Make sure that you start your drag process with your cursor **outside** of the first field. If you start dragging within a field, Composer assumes you are trying to move the field itself.

6 The FILE BROWSE screen appears as shown.



```
                FILE BROWSE

NUMBER         _NAME           _AMOUNT
111111  FRED HOLAHAN          $7575.50
222222  CHRIS KELLER          $2000.00
333333  EDWARD BUDGEN         $3000.00
444444  RAVI VEERARAGHAV      $2340.00


PRESS CLEAR TO END BROWSE OPERATION
PRESS PF1 OR TYPE F TO PAGE FORWARD
PRESS PF2 OR TYPE B TO PAGE BACKWARD
```

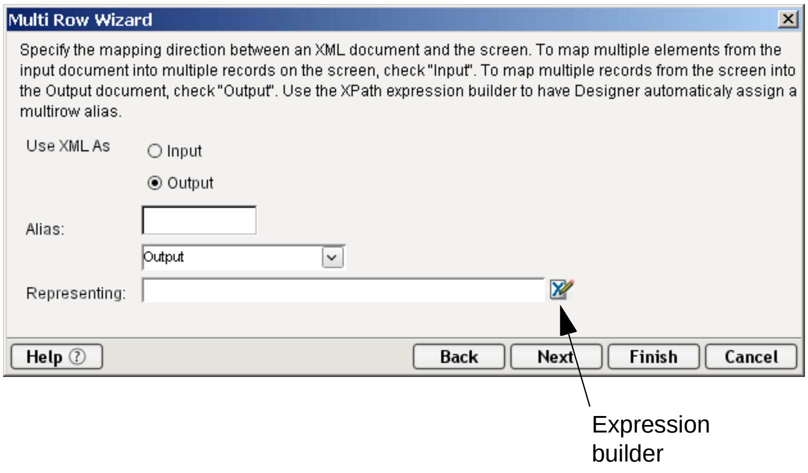
7 From the 3270 Component Editor menu, select **Action**, then **New Action**, then **Multi Row**. The Multi Row Wizard appears. The wizard automatically fills in the dialog based on the area you highlighted in the previous step.


Multi Row Wizard

Specify the dimensions and layout of the screen area containing multiple records. If the multiple records span across more than one screen, this multi row area must have the same specifications on each screen received from the host.

Start Line: 1
Start Column: 1
Records Down: 1
Records Across: 1
Lines/Record: 1
Columns/Record: 60

8 Edit the fields if desired. When satisfied with the parameters, click **Next**. The second panel of the Multi Row Wizard appears.

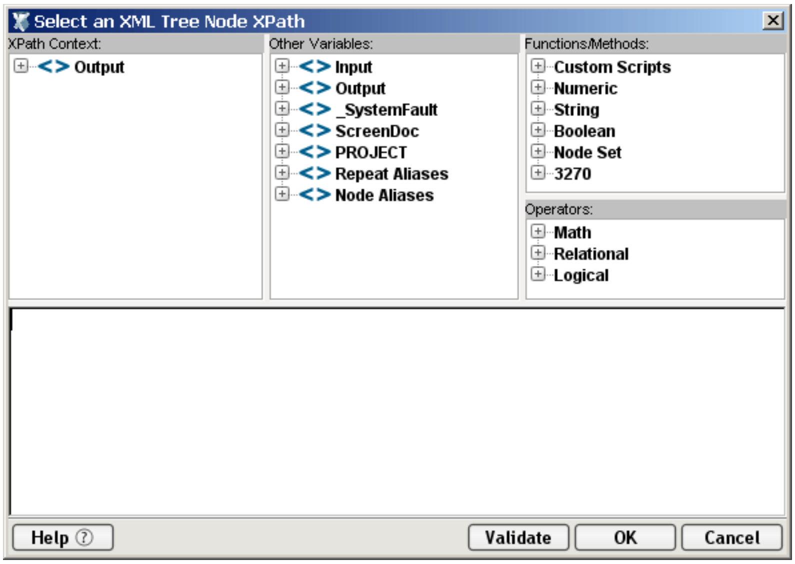

Multi Row Wizard

Specify the mapping direction between an XML document and the screen. To map multiple elements from the input document into multiple records on the screen, check "Input". To map multiple records from the screen into the Output document, check "Output". Use the XPath expression builder to have Designer automaticaly assign a multirow alias.

Use XML As: Input [✓ Output]
Alias:
Output
Representing:

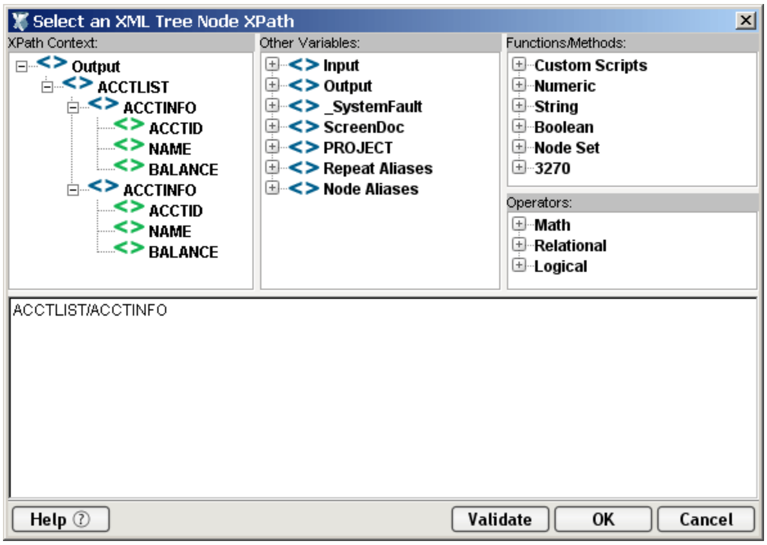
9 Select the **Output** radio button for the **Use XML As**. This panel is used to create a repeat action for processing the multiple elements or screen rows. Its use is similar to the basic “Repeat for Element” action available in all components.

In the **Representing** field, click the Expression builder button. The Expression builder window appears.



Expand the **Output** element in the **XPath Content** frame.

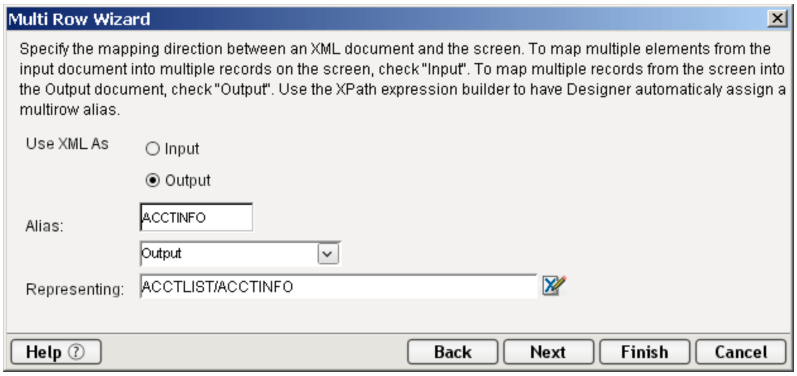
10 Navigate to the **ACCINFO** element and double click. The expression appears in the comment pane in the bottom of the window. For this example, the ACCINFO element is the one on which we will loop.



ACCTLIST/ACCTINFO

11 Click **OK**. You are returned to the second screen of the Multi Row Wizard, which now appears with the new expression in the **Representing** field.

NOTE: When using the Expression builder, Composer automatically creates an **Alias**. In this example, Composer created the alias called ACCINFO. For more information on aliases, please see the *exteNd Composer User’s Guide*.



Multi Row Wizard

Specify the mapping direction between an XML document and the screen. To map multiple elements from the input document into multiple records on the screen, check "Input". To map multiple records from the screen into the Output document, check "Output". Use the XPath expression builder to have Designer automaticaly assign a multirow alias.

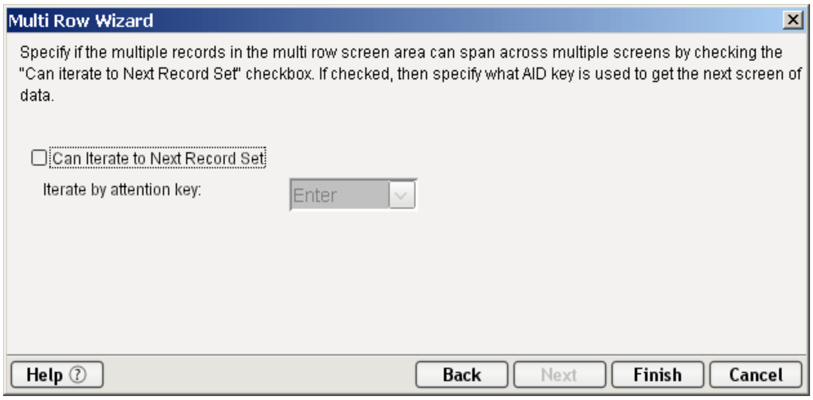
Use XML As: ○ Input [✓ ● Output]

Alias: ACCINFO

Output

Representing: ACCTLIST/ACCTINFO

12 Click the **Next** button. The Iterate screen of the Multi Row Wizard appears. This screen tells the wizard what to do when it encounters an end of page.



Multi Row Wizard

Specify if the multiple records in the multi row screen area can span across multiple screens by checking the "Can iterate to Next Record Set" checkbox. If checked, then specify what AID key is used to get the next screen of data.

☐ Can Iterate to Next Record Set

Iterate by attention key: Enter

13 Check the **Can Iterate to Next Record Set** box.

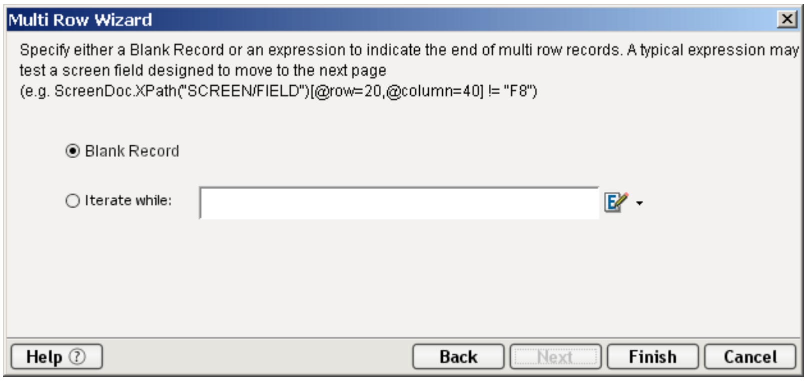
14 Select F1 from the **Iterate by attention key** pull down list.

NOTE: The 3270 screen itself contains the instructions for the iteration keys. For this example, the FILE BROWSE screen included instructions that said to use the F1 key to page forward.

15 Click **Next**.

➤ To complete the Boundary parameter page of the Multi Row Wizard:

1 Select the **Blank Record** radio button. This tells the wizard that a blank record indicates the end point of the loop action. You may also use the Expression builder to set a different indicator for the loop's end point. For more information on using the Expression builder, please see the *exteNd Composer User's Guide*

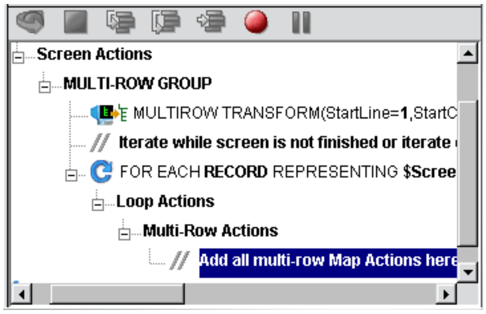

Multi Row Wizard

Specify either a Blank Record or an expression to indicate the end of multi row records. A typical expression may test a screen field designed to move to the next page
(e.g. ScreenDoc.XPath("SCREEN/FIELD")[@row=20,@column=40] != "F8")

[✓ Blank Record]

Iterate while:

2 Click **Finish**. The Multi Row actions you created in the wizard appear in the Action Model pane, with the **Add all multi row map actions here!** comment highlighted.


```
Screen Actions
  MULTI-ROW GROUP
    MULTIROW TRANSFORM(StartLine=1,StartC
    Iterate while screen is not finished or iterate
    FOR EACH RECORD REPRESENTING $Scree
      Loop Actions
        Multi-Row Actions
          Add all multi-row Map Actions here
```

➤ To add the Multi Row Actions to output data:

1 Highlight the **Add all multi row map actions here!** comment in the Action Model if not already highlighted.

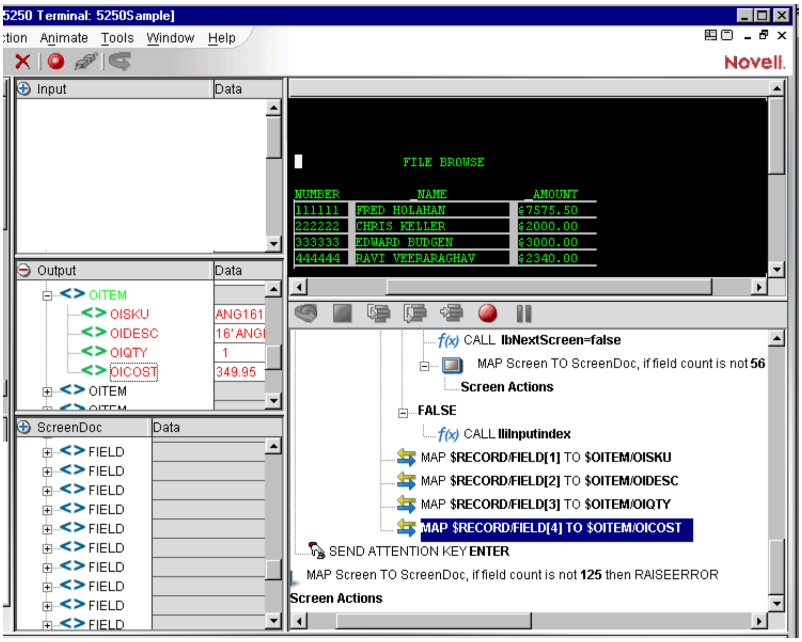
2 Navigate to the first instance of ACCINFO in the Output DOM pane.

3 Drag and drop the data from the 3270 FILE BROWSE screen to the Output DOM as follows:

Table 4-4

From: FILE BROWSE	To: Output DOM
NUMBER	ACCTID
NAME	NAME
AMOUNT	BALANCE

4 The Output DOM appears as shown.



```
            FILE BROWSE

NUMBER        NAME                 AMOUNT
111111   FRED HOLAHAN          $7575.50
222222   CHRIS KELLER          $2000.00
333333   EDWARD BUDGEN         $3000.00
444444   RAVI VEERARAGHAV      $2340.00
```

```
OITEM
    OISKU     ANG161
    OIDESC    16'ANGI
    OIQTY     1
    OICOST    349.95
OITEM
OITEM
```

```
FIELD
FIELD
FIELD
FIELD
FIELD
FIELD
FIELD
FIELD
FIELD
FIELD
```

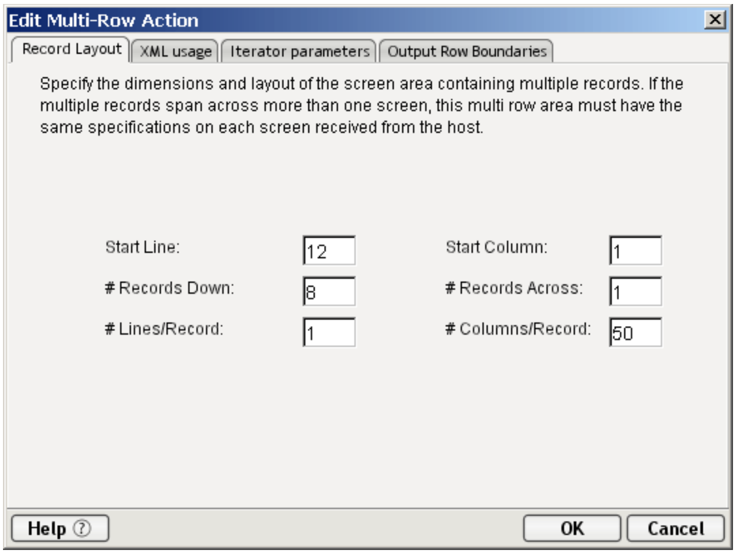
```
f(x) CALL lbNextScreen=false
MAP Screen TO ScreenDoc, if field count is not 56
    Screen Actions
FALSE
    f(x) CALL lIiInputindex
MAP $RECORD/FIELD[1] TO $OITEM/OISKU
MAP $RECORD/FIELD[2] TO $OITEM/OIDESC
MAP $RECORD/FIELD[3] TO $OITEM/OIQTY
MAP $RECORD/FIELD[4] TO $OITEM/OICOST
SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 125 then RAISEERROR
Screen Actions
```

5 Select **File**, then **Save** from the 3270 Component Editor menu bar, or click the **Save** button.

6 Follow the instructions in “Using the Animation Tools” on page 88 to test your component.

➤ To edit the Multi Row Actions:

1 Click on the MultiRow Action that you wish to change in the Action Pane. The dialog box appears.


Edit Multi-Row Action

Specify the dimensions and layout of the screen area containing multiple records. If the multiple records span across more than one screen, this multi row area must have the same specifications on each screen received from the host.

Start Line:	12	Start Column:	1
# Records Down:	8	# Records Across:	1
# Lines/Record:	1	# Columns/Record:	50

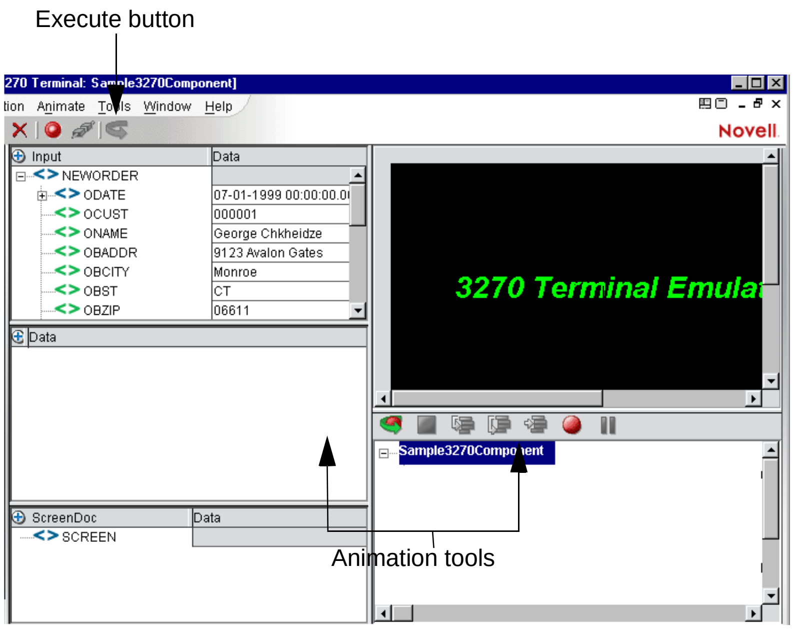
2 Click on the appropriate Tab, edit the fields and click the **OK** button. Refer to the Previous sections on using the MuliRow Wizard to Input or Output data.

Executing your 3270 Component

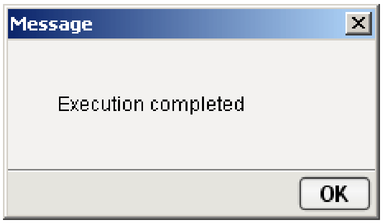
Composer includes animation tools that allow you to test your component. On the 3270 Component Editor tool bar you'll find the **Execute** button, which allows you to execute the entire Action Model and verify that your component works as you intend.

➤ **To execute a 3270 component:**

1 Open a 3270 component. The 3270 Component Editor window appears.



2 Select the **Execute** button. The actions in the Action Model execute and, when complete, a message appears.



Execution completed

3 Click **OK**.

4 From the **View** menu, select **Expand XML Documents**. This expands all of the parents, children, data elements, etc. of the XML Documents, which allows you to see the results of the executed component. If you do not expand the XML Documents, you won't see if the data you wanted to move from the 3270 environment made it to the Output DOM.

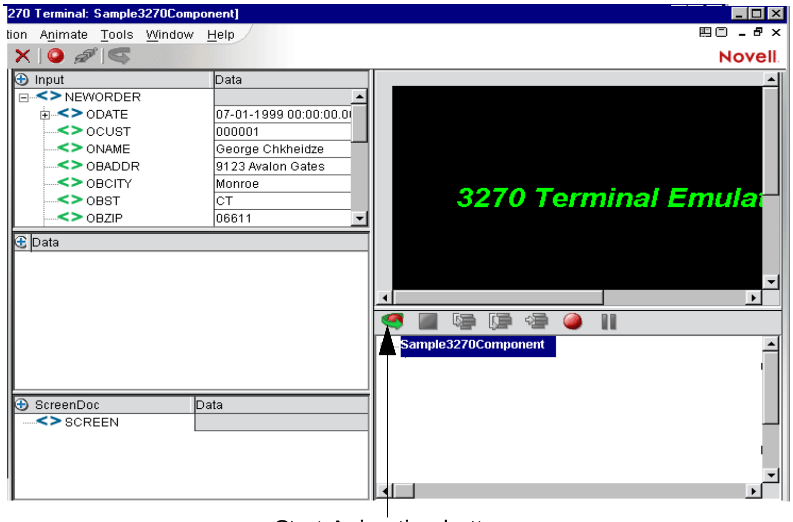
Using the Animation Tools

In the Action Model, you'll find animation tools that allow you to test a particular section of the Action Model by setting one or more breakpoints. This way, you can run through the actions that work properly, stop at the actions that are giving you trouble, and then troubleshoot the problem actions one at a time.

NOTE: The following procedure is a brief example of the functionality of the animation tools. For a complete description of all the animation tools and their functionality, please refer to the *exteNd Composer User's Guide*.

➤ **To run the animation:**

1 Open a 3270 Component. The component appears in the 3270 Component Editor window.


270 Terminal: Sample3270Component]
tion Animate Tools Window Help
Novell
3270 Terminal Emulat
Sample3270Component

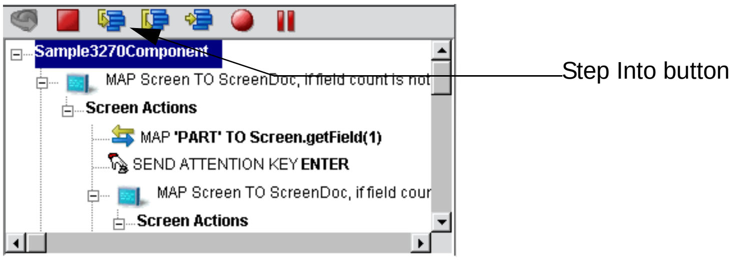
Input	Data
NEWORDER	
ODATE	07-01-1999 00:00:00.0
OCUST	000001
ONAME	George Chkheidze
OBADDR	9123 Avalon Gates
OBCITY	Monroe
OBST	CT
OBZIP	06611

ScreenDoc	Data
SCREEN	

Start Animation button

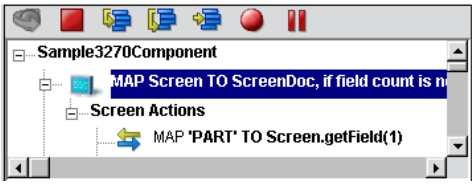
NOTE: Animation and Recording are mutually exclusive modes in the component. In order to record during animation, you must either pause, or stop animation and then turn on record mode.

2 Click the **Start Animation** button in the Action Model tool bar, or press **F5** on the keyboard. All of the tools on the tool bar become active.



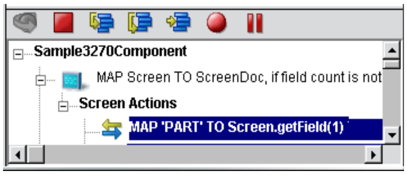
```
Sample3270Component
  MAP Screen TO ScreenDoc, if field count is not
    Screen Actions
      MAP 'PART' TO Screen.getField(1)
      SEND ATTENTION KEY ENTER
      MAP Screen TO ScreenDoc, if field cour
        Screen Actions
```

3 Click the **Step Into** button. The first Map Screen action becomes highlighted.



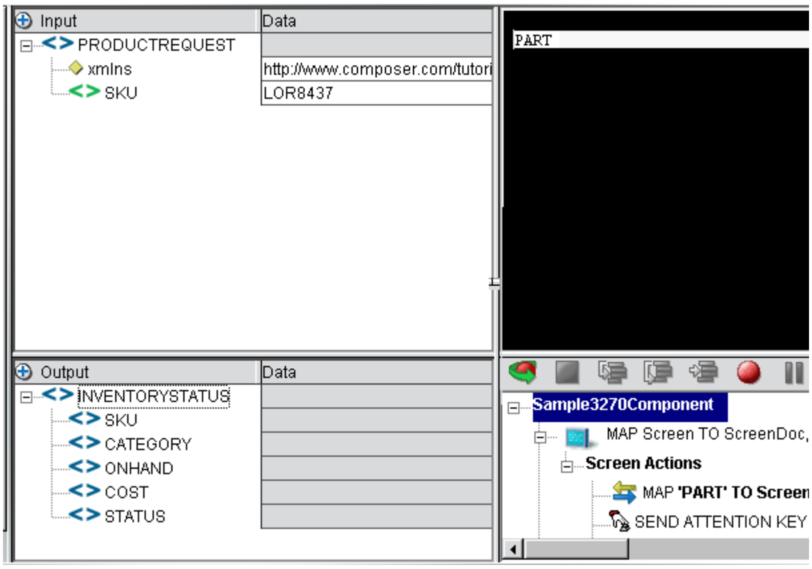
```
Sample3270Component
  MAP Screen TO ScreenDoc, if field count is n
    Screen Actions
      MAP 'PART' TO Screen.getField(1)
```

4 Click the **Step Into** button again. The instruction that enters the command “PART” into the input field of the Native Environment pane becomes highlighted.



```
Sample3270Component
  MAP Screen TO ScreenDoc, if field count is not
    Screen Actions
      MAP 'PART' TO Screen.getField(1)
```

5 Click the **Step Into** button again. The word “PART” appears in the input line of the Native Environment pane.



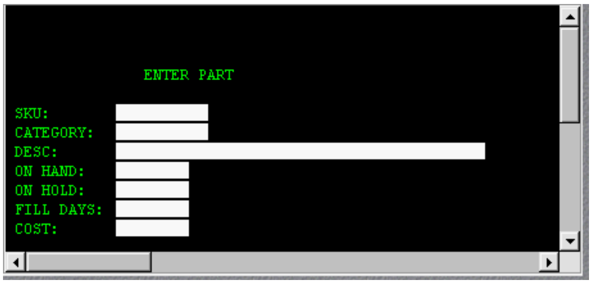
Input	Data
PRODUCTREQUEST	
xmlns	http://www.composer.com/tutori
SKU	LOR8437

Output	Data
INVENTORYSTATUS	
SKU	
CATEGORY	
ONHAND	
COST	
STATUS	

```
PART
```

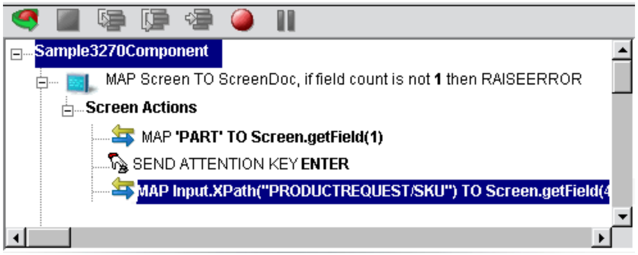
Sample3270Component
MAP Screen TO ScreenDoc,
Screen Actions
MAP 'PART' TO Screen
SEND ATTENTION KEY

6 Click the **Step Into** button again. The ENTER PART screen appears in the Native Environment pane.



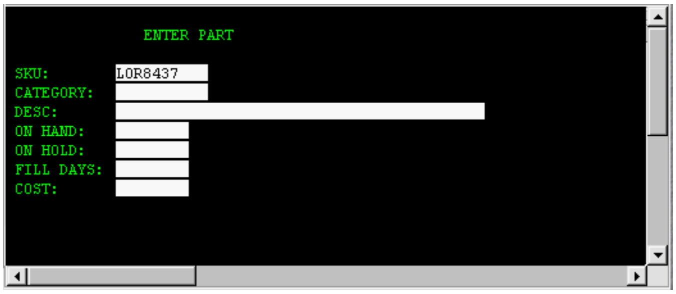
```
               ENTER PART

SKU:
CATEGORY:
DESC:
ON HAND:
ON HOLD:
FILL DAYS:
COST:
```

7 Click the **Step Into** button again. In the Action Model, the instruction for dragging and dropping the SKU from the Input DOM to the SKU field of the ENTER PART screen field becomes highlighted.



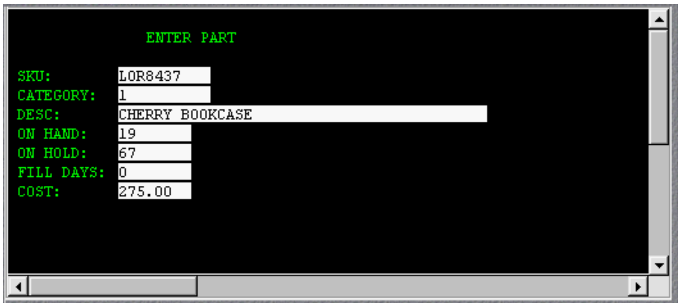
Sample3270Component
MAP Screen TO ScreenDoc, if field count is not **1** then RAISEERROR
Screen Actions
MAP **'PART' TO Screen.getField(1)**
SEND ATTENTION KEY **ENTER**
MAP Input.XPath("PRODUCTREQUEST/SKU") TO Screen.getField(4

8 Click the **Step Into** button again. The SKU data from the Input DOM appears in the SKU field of the ENTER PART screen.



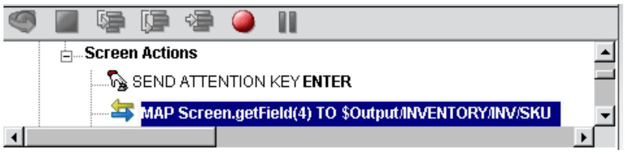
```
ENTER PART

SKU:        LOR8437
CATEGORY:
DESC:
ON HAND:
ON HOLD:
FILL DAYS:
COST:
```

9 Click the **Step Into** button again. The ENTER PART screen becomes populated with the data associated with the SKU number.



```
ENTER PART

SKU:        LOR8437
CATEGORY:   1
DESC:       CHERRY BOOKCASE
ON HAND:    19
ON HOLD:    67
FILL DAYS:  0
COST:       275.00
```

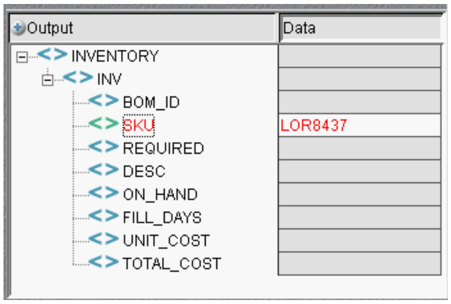
10 Click the **Step Into** button again. In the Action Model, the instruction to drag the SKU data from the ENTER PART screen to the Output DOM becomes highlighted.


Screen Actions

SEND ATTENTION KEY **ENTER**

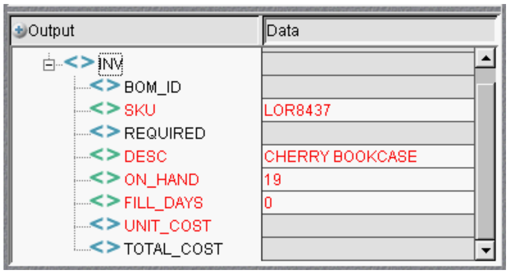
MAP Screen.getField(4) TO $Output/INVENTORY/INV/SKU

11 Click the **Step Into** button again. The SKU data from the ENTER PART screen appears in the SKU field of the Output DOM.



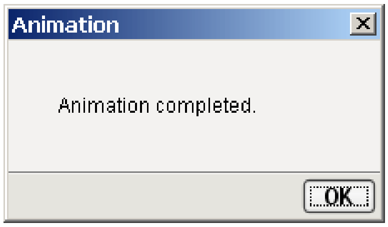
Output	Data
INVENTORY	
INV	
BOM_ID	
SKU	LOR8437
REQUIRED	
DESC	
ON_HAND	
FILL_DAYS	
UNIT_COST	
TOTAL_COST	

12 Continue to click the **Step Into** button until all the data elements from the ENTER PART fields appear in the Output DOM, as shown.



Output	Data
INV	
BOM_ID	
SKU	LOR8437
REQUIRED	
DESC	CHERRY BOOKCASE
ON_HAND	19
FILL_DAYS	0
UNIT_COST	
TOTAL_COST	

13 Once complete, the following message appears.


Animation

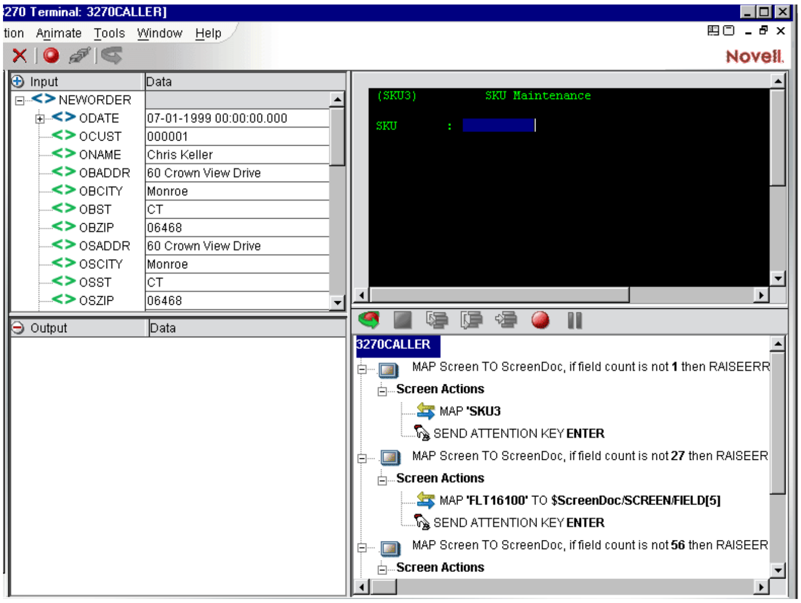
Animation completed.

Component with Connection Action

The Component with Connection Action is unique because it allows a 3270 component to call another component allowing that component to share the same connection. The action allows you to break up a large component into a main 3270 component and a sub-components so it is easier to maintain the Action Model. The ability to have the main component share with the sub-component the connection greatly reduces the amount of connection overhead and transaction navigation at run time. Before you begin, determine how many sub-components you will require and then build and save the shells (containing no recorded actions) for when you are ready to begin recording.

➢ To use the Component with Connect Action

1. Create and record the basic structure of the main component to the point where you are ready to call a sub-component. For this example, the main component will be entitled “3270 Caller.”



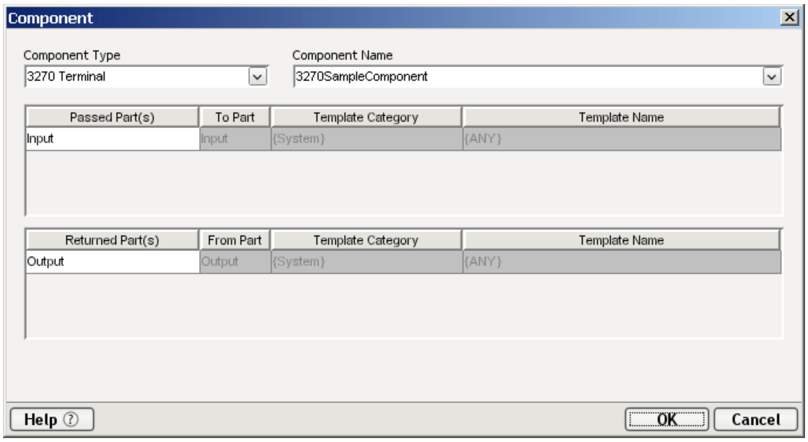
Input	Data
NEWORDER	
ODATE	07-01-1999 00:00:00.000
OCUST	000001
ONAME	Chris Keller
OBADDR	60 Crown View Drive
OBCITY	Monroe
OBST	CT
OBZIP	06468
OSADDR	60 Crown View Drive
OSCITY	Monroe
OSST	CT
OSZIP	06468

Output	Data

```
(SKU3)          SKU Maintenance

SKU       :
```

```
3270CALLER
MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERR
  Screen Actions
    MAP 'SKU3
    SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 27 then RAISEER
  Screen Actions
    MAP 'FLT16100' TO $ScreenDoc/SCREEN/FIELD[5]
    SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 56 then RAISEER
  Screen Actions
```

2. From the Main menu, or by clicking the RMB, select **New Action>Component/w connection**. The following dialog appears.



Component Type: 3270 Terminal

Component Name: 3270SampleComponent

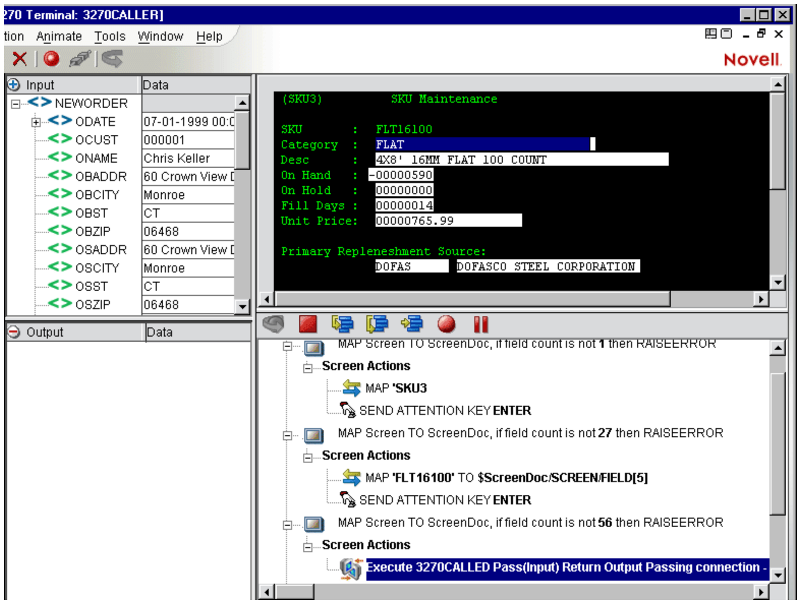
Passed Part(s)	To Part	Template Category	Template Name
Input	Input	{System}	{ANY}

Returned Part(s)	From Part	Template Category	Template Name
Output	Output	{System}	{ANY}

3 From the Component Type pull down list, select the name of the component type. From the Component Name pull down list, select the name of the Component.

4 Select the passed ID if you need to change it from the pull down list. Select the returned ID if you need to change it from the pull down list. Click OK.

5 The following action appears in the map pane.

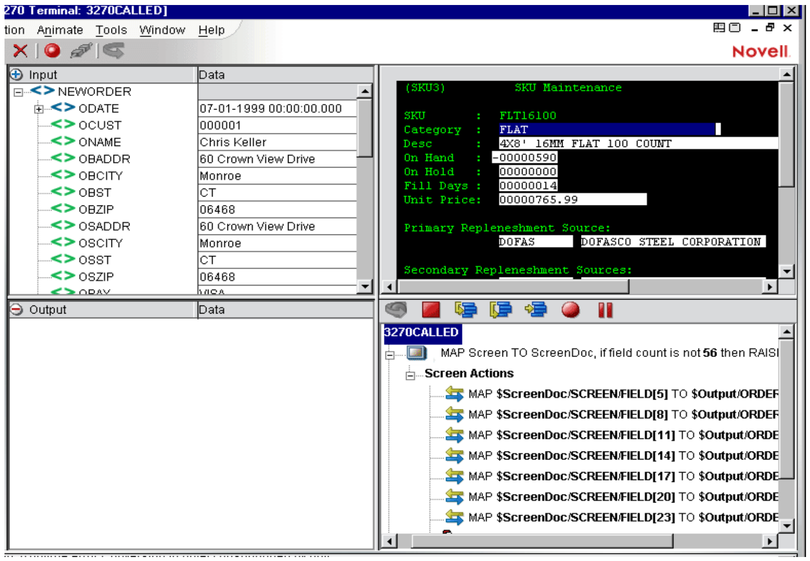


Input	Data
NEWORDER	
ODATE	07-01-1999 00:0
OCUST	000001
ONAME	Chris Keller
OBADDR	60 Crown View [
OBCITY	Monroe
OBST	CT
OBZIP	06468
OSADDR	60 Crown View [
OSCITY	Monroe
OSST	CT
OSZIP	06468

```
(SKU3)          SKU Maintenance

SKU       :  FLT16100
Category  :  FLAT
Desc      :  4X8'  16MM  FLAT  100  COUNT
On Hand   : -00000590
On Hold   :  00000000
Fill Days :  00000014
Unit Price:  00000765.99

Primary Replenishment Source:
             DOFAS        DOFASCO  STEEL  CORPORATION
```

```
MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
  Screen Actions
    MAP 'SKU3
    SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 27 then RAISEERROR
  Screen Actions
    MAP 'FLT16100' TO $ScreenDoc/SCREEN/FIELD[5]
    SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 56 then RAISEERROR
  Screen Actions
    Execute 3270CALLED Pass(Input) Return Output Passing connection -
```

6 Animate the Main component and step into the Component with Connection action, the sub-component will now open. See how the screen changed to the component entitled “3270Called.”



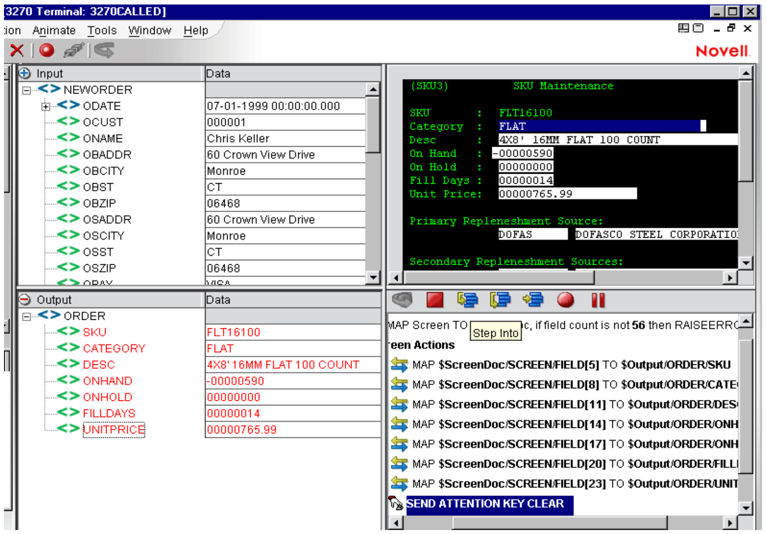
Input	Data
NEWORDER	
ODATE	07-01-1999 00:00:00.000
OCUST	000001
ONAME	Chris Keller
OBADDR	60 Crown View Drive
OBCITY	Monroe
OBST	CT
OBZIP	06468
OSADDR	60 Crown View Drive
OSCITY	Monroe
OSST	CT
OSZIP	06468
OPAY	VISA

```
(SKU3)              SKU Maintenance

SKU         :  FLT16100
Category    :  FLAT
Desc        :  4X8'  16MM  FLAT  100  COUNT
On Hand     : -00000590
On Hold     :  00000000
Fill Days   :  00000014
Unit Price:    00000765.99

Primary Repleneshment Source:
               DOFAS        DOFASCO  STEEL  CORPORATION

Secondary Repleneshment Sources:
```

```
3270CALLED
  MAP Screen TO ScreenDoc, if field count is not 56 then RAISI
    Screen Actions
      MAP $ScreenDoc/SCREEN/FIELD[5] TO $Output/ORDER
      MAP $ScreenDoc/SCREEN/FIELD[8] TO $Output/ORDER
      MAP $ScreenDoc/SCREEN/FIELD[11] TO $Output/ORDE
      MAP $ScreenDoc/SCREEN/FIELD[14] TO $Output/ORDE
      MAP $ScreenDoc/SCREEN/FIELD[17] TO $Output/ORDE
      MAP $ScreenDoc/SCREEN/FIELD[20] TO $Output/ORDE
      MAP $ScreenDoc/SCREEN/FIELD[23] TO $Output/ORDE
```

7 Click on the Pause button on the Animation tool bar to enable the Record button.

8 Click on Record button and record the sub-component actions.



Input	Data
NEWORDER	
ODATE	07-01-1999 00:00:00.000
OCUST	000001
ONAME	Chris Keller
OBADDR	60 Crown View Drive
OBCITY	Monroe
OBST	CT
OBZIP	06468
OSADDR	60 Crown View Drive
OSCITY	Monroe
OSST	CT
OSZIP	06468
OPAY	VISA

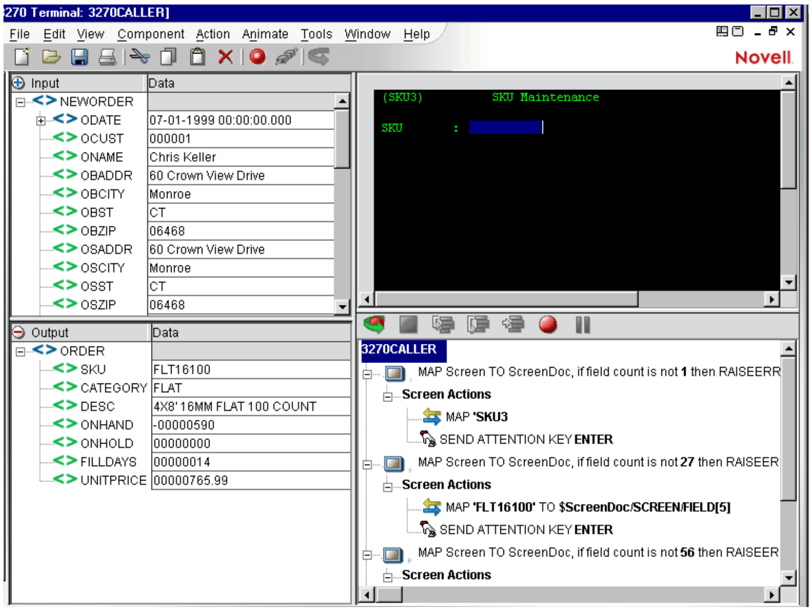
Output	Data
ORDER	
SKU	FLT16100
CATEGORY	FLAT
DESC	4X8' 16MM FLAT 100 COUNT
ONHAND	-00000590
ONHOLD	00000000
FILLDAYS	00000014
UNITPRICE	00000765.99

```
(SKU3)              SKU Maintenance

SKU         :  FLT16100
Category    :  FLAT
Desc        :  4X8'  16MM  FLAT  100  COUNT
On Hand     : -00000590
On Hold     :  00000000
Fill Days   :  00000014
Unit Price:    00000765.99

Primary Repleneshment Source:
               DOFAS        DOFASCO  STEEL  CORPORATIO

Secondary Repleneshment Sources:
```

```
MAP Screen TO          c, if field count is not 56 then RAISEERRC
een Actions
  MAP $ScreenDoc/SCREEN/FIELD[5] TO $Output/ORDER/SKU
  MAP $ScreenDoc/SCREEN/FIELD[8] TO $Output/ORDER/CATE
  MAP $ScreenDoc/SCREEN/FIELD[11] TO $Output/ORDER/DES
  MAP $ScreenDoc/SCREEN/FIELD[14] TO $Output/ORDER/ONH
  MAP $ScreenDoc/SCREEN/FIELD[17] TO $Output/ORDER/ONH
  MAP $ScreenDoc/SCREEN/FIELD[20] TO $Output/ORDER/FILL
  MAP $ScreenDoc/SCREEN/FIELD[23] TO $Output/ORDER/UNIT
  SEND ATTENTION KEY CLEAR
```

9 Save the component before stopping the recording process.

10 The 3270 Called window will now switch back to the 3270 Caller window.


270 Terminal: 3270CALLER]
File Edit View Component Action Animate Tools Window Help
Novell

Input	Data
NEWORDER	
ODATE	07-01-1999 00:00:00.000
OCUST	000001
ONAME	Chris Keller
OBADDR	60 Crown View Drive
OBCITY	Monroe
OBST	CT
OBZIP	06468
OSADDR	60 Crown View Drive
OSCITY	Monroe
OSST	CT
OSZIP	06468

Output	Data
ORDER	
SKU	FLT16100
CATEGORY	FLAT
DESC	4X8' 16MM FLAT 100 COUNT
ONHAND	-00000590
ONHOLD	00000000
FILLDAYS	00000014
UNITPRICE	00000765.99

```
(SKU3)          SKU Maintenance

SKU       :
```

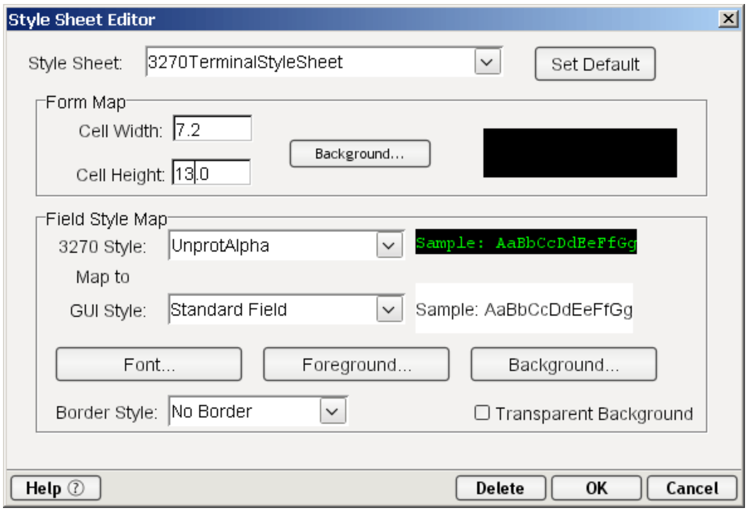
```
3270CALLER
MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERR
  Screen Actions
    MAP 'SKU3
    SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 27 then RAISEER
  Screen Actions
    MAP 'FLT16100' TO $ScreenDoc/SCREEN/FIELD[5]
    SEND ATTENTION KEY ENTER
MAP Screen TO ScreenDoc, if field count is not 56 then RAISEER
  Screen Actions
```

Using Style Sheets in the Native Environment Pane

The Style Sheet feature of the 3270 Component Editor provides you with options as to how you want to view the Native Environment pane.

➤ **To apply a Style Sheet to the Native Environment pane:**

1 From the **Component** menu on the 3270 Component Editor window, select **Style Sheet**. The Style Sheet Editor dialog appears.



Style Sheet Editor

Style Sheet: 3270TerminalStyleSheet

Form Map

Cell Width: 7.2

Cell Height: 13.0

Field Style Map

3270 Style: UnprotAlpha

Sample: AaBbCcDdEeFfGg

Map to GUI Style: Standard Field

Sample: AaBbCcDdEeFfGg

Border Style: No Border

☐ Transparent Background

2 Choose a Style Sheet from the Style Sheet drop down list. For detailed information on using the Style Sheet editor, see “Creating a Style Sheet Resource” on page 29.

3 Click **OK**.

Using Other Actions in the 3270 Component Editor

In addition to the Map Screen and Multi Row action, you have all the standard Basic and Advanced Composer actions at your disposal as well. The complete listing of Basic Composer Actions can be found in Chapter 7 of the *Composer User's Guide*. Chapter 8 contains a listing of the more Advanced Actions available to you.

Handling Errors and Messages

This section describes common errors you may see while executing the animation tools.

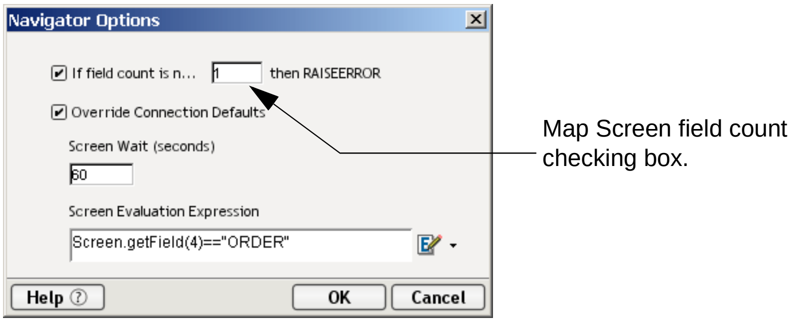
Screen Field Count Changed

This error occurs during animation or execution of certain transactions. One condition that can cause this is when a transaction sends one or more screens to the terminal that **do not** require a response from the user (i.e., pressing an Aid key) and then sends a screen that **does** require a response. For instance, some transactions are designed to display a message screen (e.g., "Please wait...") and then display the actual transaction screen that the user wants. The user cannot respond to the message screen, its display is under the control of the transaction. The problem with transactions behaving this way occurs during animation. As you step from action to action, Composer's Map Screen actions count how many fields each screen has and compares this number with the original field count when you recorded the component. Since the transaction can send a second screen before you have stepped to its corresponding Map Screen action, the field counts get out of sync.

To correct this error, you must determine why the field count has changed and then try one of the following remedial actions.

- Double-click the Map Screen action that failed in the Action Model and change the field count variable to the correct number of fields.

  Double-click the Map Screen action, disable the field count checking, and wrap the whole Map Screen action block within a Try On Error action that will allow you to conditionally process a field count error instead of an exception being thrown, which halts component
  execution.



☑ If field count is n... 1 then RAISEERROR

☑ Override Connection Defaults

Screen Wait (seconds)

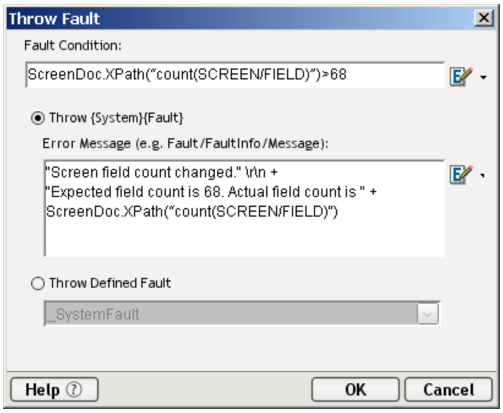
60

Screen Evaluation Expression

Screen.getField(4)=="ORDER"

The Navigator Options Dialog allows you to:

- Enter the **field count** (number) which will raise an error
- You MUST remember to re-select this check box before deployment.
- **Override connection defaults** by clicking in the checkbox to allow an override based on the seconds entered in the **Screen Wait** field for one navigation action.
- Set an expression that is checked each time a packet was received and processed
- Return to the next step (which may include Screen Field Count check) if the Screen Evaluation Expression is true.
- Disable the Map Screen field count checking and add your own action to check the field count. You can do this by creating a Throw Fault action that checks the field count (e.g., ScreenDoc.XPath(“count(SCREEN/FIELD)”)>68) needed for your application. An example is shown below.


Throw Fault

Fault Condition:

```
ScreenDoc.XPath("count(SCREEN/FIELD)")>68
```

(● Throw {System}{Fault})

Error Message (e.g. Fault/FaultInfo/Message):

```
"Screen field count changed." \r\n +
"Expected field count is 68. Actual field count is " +
ScreenDoc.XPath("count(SCREEN/FIELD)")
```

○ Throw Defined Fault

_SystemFault

There are additional circumstances where field counts can get of sync, for example:

- A spooler message from a print job arrives unexpectedly. To remedy this, the user would delete the errant Map Screen action from the Action Model.
- Applications that send the same screen but with a variable number of fields.

Errors Involving Connections

If connection pooling is used, and there has been an attempt to log on with a bad UserID or Password, that connection instance will not be usable and that member of the pool will be skipped over in subsequent connection requests. An error message will be sent to the server log saying "Logon connection in pool <Pool name> was discarded for User ID <User ID>." You should check for messages of this sort during preproduction testing and/or any time performance issues arise.

There are a few different steps you can take to resolve a bad UserID or Password:

- Alert your System Administrator that the UserID and Password doesn't work. If the administrator checks the ID and Password and it is good, then reset the server from the Connection Manager Console.
- If the UserID and Password is bad, remove it from the connection pool.

5 Logon Components, Connections, and Connection Pools

This section discusses certain features available in the 3270 Connect designed to maximize performance of deployed services.

About 3270 Terminal Session Performance

The overall performance of any service that uses back-end connectivity is usually dependent on the time it takes to establish a connection and begin interacting with the host. Obtaining the connection is “expensive” in terms of wait time. One strategy for dealing with this is *connection pooling*, a scheme whereby an intermediary process (whether the app server itself, or some memory-resident background process not associated with the server) maintains a set number of preestablished, pre-authenticated connections, and oversees the “sharing out” of these connections among client apps or end users.

Connection pooling overcomes the latency involved in opening a connection and authenticating to a host. But in terminal-based applications, a considerable amount of time can be spent “drilling down” through menu selections and navigating setup screens in order to get to the first bonafide application screen of the session. So even when connections are reused through pooling, session-prolog overhead can be a serious obstacle to performance.

Composer addresses these issues by providing connection pooling, managed by a special kind of component (called a *logon component*) that can maintain an open connection at a particular “drill-down” point in a terminal session, so that clients can begin transactions *immediately* at the proper point in the session.

When Will I Need Logon Components?

Logon Components are useful in several types of situations:

- When you have a need for multiple tiers of pooling based on multiple security challenges within your system. (For example, users may need one set of logon credentials to get into the network, another to get into the mainframe, and another to get into the CICS region.) Serial log-in requirements may dictate the use of multiple logon components.
- When your service needs stateful "session-based" connections.
- When you need the performance advantages available through connection pooling.

If performance under load is not a high-priority issue and your connectivity needs are relatively uncomplicated, you may not need to use Logon Components at all. But there is no way to know if performance is adequate merely by testing services at design time, on a desktop machine.

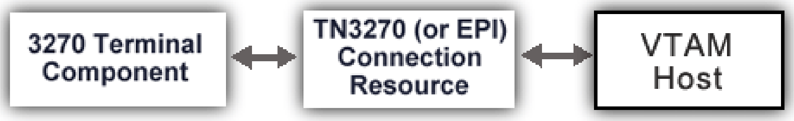
Components and services built with the 3270 Component Editor may appear to execute quickly at design time (in Animation Mode, for example). But in real-world conditions—which is to say under load, with dozens or even hundreds of requests per second arriving at the server—session overhead can be a significant factor in overall transaction time. *The only way to know whether you need to use the special performance enhancement features described in this chapter is to do load testing on a server, under test conditions that mimic real-world "worst case" conditions.*

Connection Pool Architecture

When you install the 3270 Terminal Connect, three types of Connection Resources are added to Composer's Connection Resource wizard:

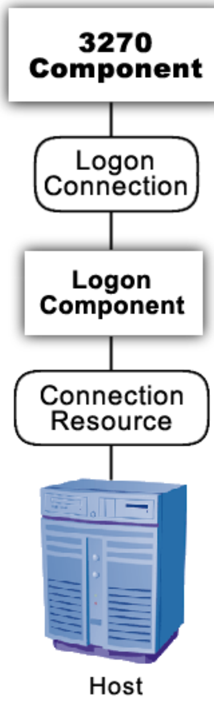
- TN3270 Connection
- EPI 3270 Connection
- 3270 Logon Connection (henceforth simply called a Logon Connection)

The TN3270 and EPI connections are true terminal connections that establish a session with a host system (e.g., a VTAM host—a system that supports the Virtual Telecommunications Access Method).



The TN3270 and EPI 3270 connection resources are designed to make individual connections to the host on an as-needed basis. The connection is made just-in-time and discarded as soon as the client is done. It is not reused in any way.

The Logon Connection, on the other hand, is different. It defines a pool of User IDs and passwords, each of which can make its own connection (TN3270 or EPI). The Logon Connection also serves as an indirection layer to allow clients to connect to the host at exactly the point in the host program (exactly the screen) where the client needs to start. This entry-point-location behavior is made possible by the Logon *Component*. (A Logon Connection always uses a Logon Component to get to the actual connection.) The architecture is shown in the graphic below.



A *Connection Resource* is always required in order to get to the host. (This is true for any Composer service that uses 3270 components.) For simplicity, this diagram shows the Connection Resource going directly to the host; in the real world, there may be intervening delegation layers involving VTAM.

The *Logon Component* contains Actions (an action model) designed to find a particular screen of interest in the host program. This drill-down location is the effective entry point of the transaction for any upstream process that uses this Logon Component. You can think of the Logon Component as a go-between between the *physical* connection (represented by the Connection Resource) and the logic layer (represented by the 3270 Component itself.

In order for a 3270 Component (at the top of the diagram) to use a Logon Component, it needs to enlist the aid of a *Logon Connection* resource. The Logon Connection is the bridge between the 3270 Component and the Logon Component.

The kinds and responsibilities of the various objects discussed above are summarized in the following table.

Object	Role
Connection Resource (TN3270 or EPI)	**Allows a connection to be established with a host system.**
Logon Component	**Specialized type of component in which the action model contains Logon, Keep Alive, and Logoff action blocks. This component can maintain a connection at a particular launch screen in a host program.**
Logon Connection	**Specialized type of Connection Resource that associates a pool of UserIDs and passwords with a given Logon Component type. At runtime, connections are established for client processes on demand (and reused), with one Logon Component instance per connection. Every connection in the pool provides ready access to a given point (a particular launch screen) in the host program, thanks to the associated Logon Component (see above).**
3270 Terminal Component	**Contains the action model that comprises the business logic for a particular 3270 interaction (or transaction).**

The Logon Connection’s Role in Pooling

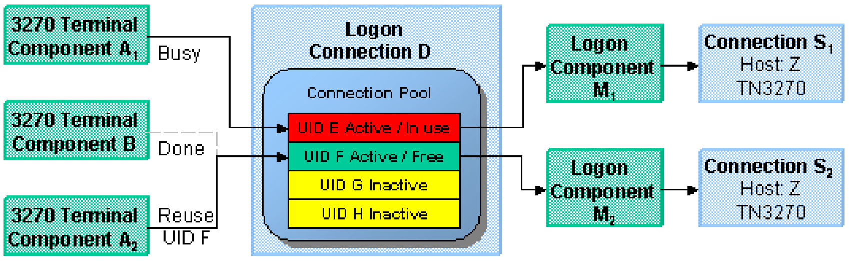
The Logon Connection differs from the ordinary “host-direct” connection resource in that it manages *pooling* (the sharing of connection instances and Logon Component instances at runtime).

In the context of a Composer service, pooling not only allows reuse of (open) connections at runtime, it also increases the effective bandwidth of a deployed service. Consider the simple case where you’ve designed a 3270 component that uses a regular connection resource. In creating the connection resource, you will have specified a UserID and password for the resource to use so that at runtime, the component can log in to the host. When an actual running instance of your component is using that connection, no other instance of the component can log in to the host using that same set of credentials. The bandwidth of your service is limited to one connected instance at a time.

With a Logon Connection, on the other hand, numerous host connections can be maintained in a “live” state so that multiple instances of your component can access the host (each on its own connection) without waiting. Throughput is dramatically increased.

The diagram below shows one possible runtime case where three component instances (two instance of 3270 Terminal Component A and one instance of 3270 Terminal Component B) are executing on the server. Instance 1 of Component A is using UserID ‘E’ to obtain a connection. This component has its own dedicated instances of Logon Component M and Connection S.

Terminal Component B has just finished executing and is relinquishing its connection (established through credentials defined by UID ‘F’). Note that because connection pooling is in effect, Component B’s downstream resources (its Logon Component instance, M2, and its Connection instance, S2) are not simply discarded; they remain live. As a result, Terminal Component A2 is able to obtain (reuse) the M2/S2 resource instances that were previously held byTerminal Component B.



In this diagram, Logon Connection D is associated with four connections based on four UIDs (user IDs or credentials: A-thru-F). One is in use; another (UID ‘F’) is alive but not being used; and two are inactive but available (i.e., valid UIDs have been assigned, so these two connections can be made live at any time).

How Many Pools Do I Need?

It’s possible for several different 3270 components to draw from the same connection pool. It’s also possible for different components to draw from different pools. This means different Logon Connections.

An important factor in deciding how many Logon Connection resources (in effect, how many pools) your service needs is the number of different start screens (or entry point screens) needed by the various components in your project. Suppose Terminal Component A needs to begin its work at a particular starting screen in a host application, but you’ve also designed another component—Terminal Component B—that needs to start on a different screen. Components A and B will need separate Logon Connections, and the separate Logon Connections will point to separate Logon Components. (In any given connection pool, Composer objects are shared in such a way that every user of the pool *must* start at the same screen.)

Pieces Required for Pooling

The combination of a Logon Connection, a Logon Component, and its Connection Resource form the basis of a connection pool. Starting from the host or VTAM layer and working up the chain:

- The *Connection Resource* defines the most basic parameters necessary for establishing a connection with the VTAM host. When connection pooling is in effect, runtime instances of this object are kept alive and reused.
- The *Logon Component* defines the set of steps (actions) necessary to get to a particular entry point in the host program. (At runtime, an instance of this component will actually carry out those steps in order to arrive at, and maintain ready-to-use, a particular screen location in the host program.) When connection pooling is in effect, instances of this object are kept alive and reused.
- The *Logon Connection* is a special type of resource that contains all the information needed to define a connection *pool*. This resource is designed to encapsulate pool-management info and does not establish host connections directly; instead, it delegates those responsibilities to the Logon Component (which delegates them, in turn, to the appropriate Connection Resource).

How Do I Implement Pooling?

To create the various objects required for pooling, you'll go through the following basic steps (each of which will be discussed in greater detail in the sections to follow):

1. First, you'll create a basic connection resource. The resource will be one of:
   - TN3270—a standard terminal connection, *or*
   - EPI 3270—a 3270 connection implemented through the External Presentation Interface
2. Next, you'll create a Logon Component that uses the connection resource defined in Step 1. As part of this process, you'll create an action model designed to navigate to a certain point in the host program.
3. You will create a Logon Connection resource, which is a specialized type of connection resource that relies on a Logon Component (from Step 2) to make the basic connection (through the resource defined in Step 1).
4. Finally, you'll create a 3720 Terminal Component and associate it with the Logon Connection resource of Step 3.

These steps are described in detail starting with the discussion at "Creating a Connection Pool" further below. Before going to that section, however, you should become familiar with the design principles behind the Logon Component (and also the Logon Connection). We'll start with the Logon Component, since it's impossible to create a Logon Connection without using a Logon Component.

The 3270 Logon Component

The Logon Component is a special type of component: It has an Action Model, yet can be used as a connection resource as well. The Action Model of this type of component is designed to manage a connection that will be used by multiple 3270 Terminal Components. In most respects, the Logon Component is the same as a 3270 Terminal component. The differences are:

1. In a Logon Component, the Action Model is organized around connection-management tasks. Those tasks are implemented via special actions: the Logon Action, KeepAlive Action, and Logoff Action.
2. A Logon Component is not invoked directly by another component or service. Its invocation is under the control of a Logon Connection.

   NOTE: A Logon Component must and can only be used in conjunction with a Logon Connection.

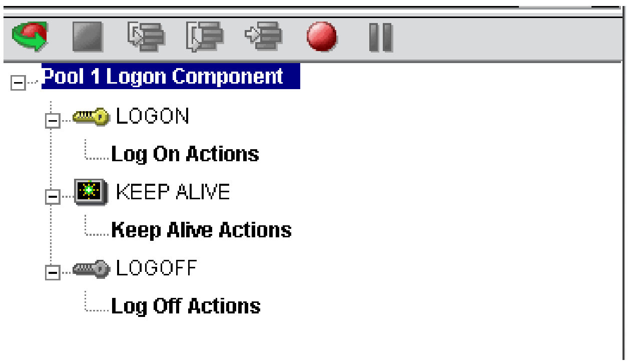
Instead of calling the Logon Component directly, using (for example) a Component Action, you will associate the Logon Component with a special connection resource called a Logon Connection. When your 3270 Terminal Component executes, it executes via the Logon Connection, which in turn executes the Logon Component.

Logon, Keep Alive, and Logoff Actions

The Logon Component provides several screen-management capabilities that are important factors in overall performance. These capabilities are implemented in terms of Logon, Keep Alive, and Logoff actions:

- **Logon Actions**—These actions navigate through the host environment and park at a desired **launch screen** in the host system. The connection is activated using UserIDs from the pool. The 3270 Terminal components that subsequently reuse the connection have the performance benefit of already being at the launch screen and won't incur the overhead of navigating to the launch screen as if they had come in under their own new session.
- **Keep Alive Actions**—These actions do two important tasks. First, they prevent the host from dropping a connection if it is not used within a standard timeout period defined by the host. Second, these actions must insure that the connection is always positioned at the "launch screen in the host, even after performing the Keep Alive actions needed to prevent the connection from dropping (the first important task).
- **Logoff Actions**—These actions exit the host environment in a manner you prescribe for all the connections made by User IDs from the pool, when a connection is being terminated.

These actions and their meanings will be discussed in greater detail below. For now, it's enough to know that these three action groupings are created for you automatically when you first create a Logon Component. Note the (empty) Logon, Keep Alive, and Logoff action blocks in the action model shown below:



Logon Actions

Actions you place in the Logon group are primarily concerned with signing into the host security screen and then navigating through the host menu system to a launch screen where each 3270 Terminal component's Action Model will start. It is important that any 3270 Terminal component using a Logon component be able to start execution at the same common screen. Otherwise, the performance gains of avoiding navigation overhead won't be realized and more importantly, the odd 3270 terminal component won't work.

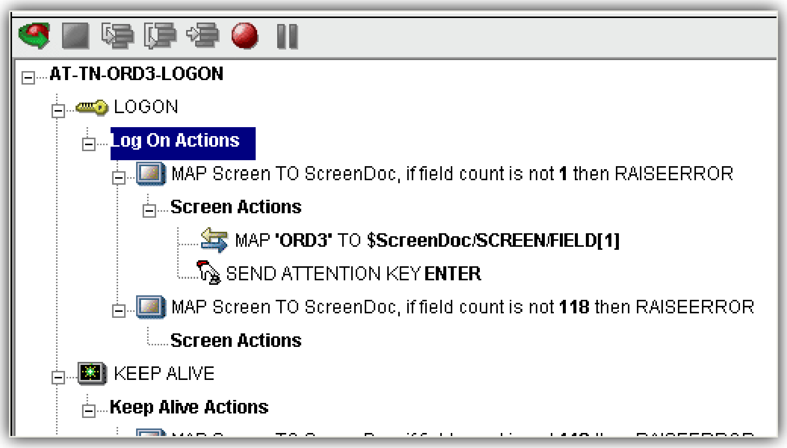
You can create actions under the Logon Actions block the same way as you would in an ordinary 3270 Terminal Component—namely by using the Record feature to create (in real time) whatever actions are necessary in order to enter sign-on info such as User ID and Password (as well as your initial menu choices to arrive at the launch screen).

NOTE: Remember to use the User IDs and Passwords from the Logon Connection Pool. (See the discussion at "Creating a Logon Connection using a Pool Connection" below.) To do this, you need to map the two special system variables called USERID and PASSWORD to the appropriate fields on the screen. By specifying these two variables, you make it possible for exteNd Composer to automatically locate and use values from the next active and free Pool slot.

The launch screen is a common point of execution for all the 3270 Terminal Components that use the User ID pool provided by a Logon Connection. The Logon actions in a Logon Component (which are executed only once when a new connection is established) let the calling component—your 3270 Terminal Component—begin execution at a given screen in the host program.

Maximizing Performance with the Logon Component

The Logon Actions must be structured properly and therefore always begin and end with a Map Screen Action as shown in the screen below.



```
AT-TN-ORD3-LOGON
  LOGON
    Log On Actions
      MAP Screen TO ScreenDoc, if field count is not 1 then RAISEERROR
        Screen Actions
          MAP 'ORD3' TO $ScreenDoc/SCREEN/FIELD[1]
          SEND ATTENTION KEY ENTER
      MAP Screen TO ScreenDoc, if field count is not 118 then RAISEERROR
        Screen Actions
  KEEP ALIVE
    Keep Alive Actions
```

The final Map Screen action in the Logon block guarantees that control is not turned over to the 3270 Component before the screen of interest has arrived in the connection. Without this, the 3270 Component could start at an invalid screen, throw an exception, and possibly corrupt a transaction. It is not necessary for the Map Screen to contain any actions, per se, but it is critical that the Map Screen prevents control being passed to the 3270 Component prematurely. It also performs a field count check on the screen when it arrives to make sure it is the same screen captured during the initial recording session.

NOTE: You may notice when animating a Logon Component that the ending Map Screen is skipped. This is normal design-time behavior. In a production environment , the actions in a Logon Component always execute in an interleaved manner with a 3270 Terminal component. Animating a Logon Component from start to finish actually creates an abnormal sequence of events that would result in two Map Screens being processed in succession, which is not allowed.

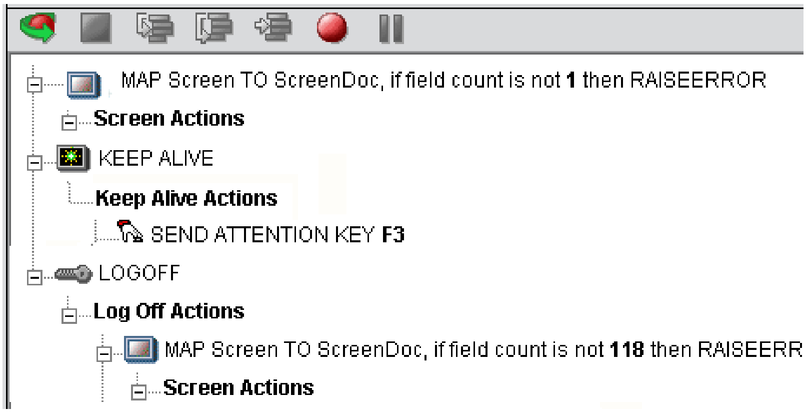
The performance benefit comes into play as a result not only of connection reuse but launch-screen reuse. For example, if a User ID pool of three entries is fully used and (ultimately) reused by the execution of a component fifteen times, the overhead of navigating to a menu item that executes the transaction of interest will occuro nly three times. Likewise, there will only be three logons to the host because the Logon actions at the top of a Logon Component are executed only once—when a new connection is activated (not when it is reused). This is key to obtaining maximum performance in a high-transaction-volume production settings.

NOTE: When possible, use the Try/On Error action to trap potential logon errors that may be recoverable. Otherwise, the UserID trying to establish the failed logon will be discarded from the pool, decreasing the potential pool size. The pool size will remain smaller until you manually reset the discarded connections using the exteNd Composer Enterprise Server Console for 3270. *See Managing Pools Sections in this Chapter for more details.*

Keep Alive Actions

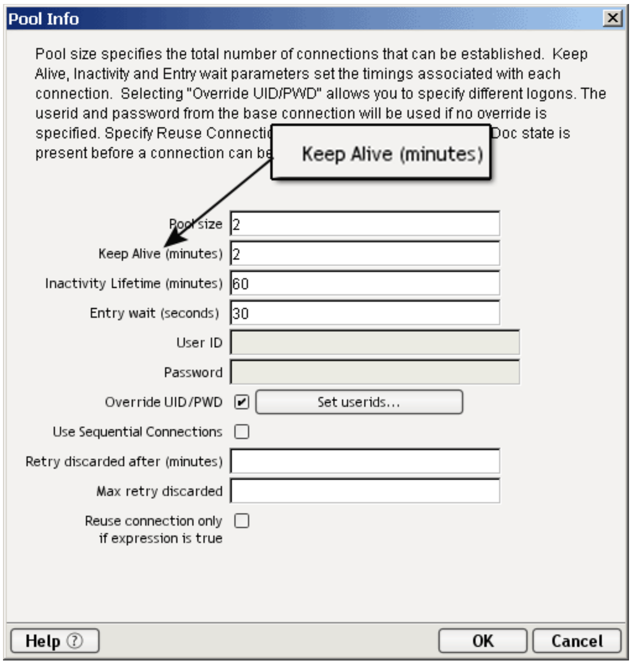
The Keep Alive block is where you will place actions that "ping the host" in whatever way necessary to keep the connection alive so that it can be reused.

Keep Alive actions usually involve sending an AID key, such as <ENTER>, to the host at some specified interval. However, if after sending the AID key the screen changes to some screen that is different than the launch screen, you must be sure to return the Logon Component to the launch screen in the Keep Alive section. Failure to do so will leave the next component at an incorrect screen, causing it to fail.



MAP Screen TO ScreenDoc, if field count is not **1** then RAISEERROR
Screen Actions
KEEP ALIVE
Keep Alive Actions
SEND ATTENTION KEY **F3**
LOGOFF
Log Off Actions
MAP Screen TO ScreenDoc, if field count is not **118** then RAISEERR
Screen Actions

The Pool Info dialog of the Logon Connection setup dialog (see discussion at "Creating a Logon Connection using a Pool Connection" below) is where you control how often the Keep Alive actions will execute. If you specify in your Logon Connection pool that you would like to keep a free connection active for three minutes, but the host will normally drop a connection after two minutes of inactivity, you can specify keyboard actions to take place at 30-second intervals to let the host know the connection is still active.


Pool Info

Pool size specifies the total number of connections that can be established. Keep Alive, Inactivity and Entry wait parameters set the timings associated with each connection. Selecting "Override UID/PWD" allows you to specify different logons. The userid and password from the base connection will be used if no override is specified. Specify Reuse Connecti… Doc state is present before a connection can be…

Pool size: 2
Keep Alive (minutes): 2
Inactivity Lifetime (minutes): 60
Entry wait (seconds): 30
User ID:
Password:
Override UID/PWD: ✓
Use Sequential Connections: ☐
Retry discarded after (minutes):
Max retry discarded:
Reuse connection only if expression is true: ☐

Keep Alive actions will be executed multiple times, at intervals defined by the Keep Alive parameter defined on the Pool Info dialog of the Logon Connection.

The Inactivity Lifetime parameter (just below Keep Alive on the Pool Info dialog) tells Composer how long it should wait, in the event the connection is not actually used by a 3270 Terminal Component, before relinquishing the connection.

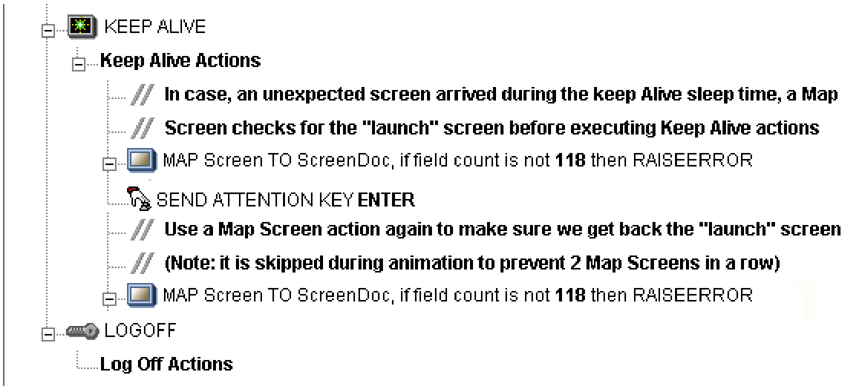
NOTE: The execution of the *Keep Alive actions* of a Logon Component will not cause the Inactivity Lifetime clock to reset in the Logon Connection. Only a 3270 Terminal component's execution will reset the Inactivity Lifetime. In other words, if a live connection is never actually used (but is merely kept alive by "Keep Alive" actions), then it will time out according to the Inactivity Lifetime value in the Pool Info dialog. But if the connection is used (by a 3270 component) before it times out, the timer is reset at that point.

The last action inside a Keep Alive block should be an empty but "enabled" navigation action. If a user disables this last action, animation will not work properly due to two consecutive empty navigation actions occurring. For example, if an action in Logon and the first action in Keep Alive are disabled, an error occurs.

Maximizing Performance with Keep Alive Actions

Map Screen actions must occur at the beginning and end of the Keep Alive section.

Not only must the Keep Alive section prevent the connection from closing, but it must make sure that the proper launch screen is present when the execution is completed. Therefore, the first Map Screen checks to make sure that during the time the connection was available but not in use, an unexpected screen didn't arrive from the host. The ending Map Screen prevents the premature release of the connection to the next 3270 Component. See below for a typical Keep Alive block.



```
KEEP ALIVE
  Keep Alive Actions
    // In case, an unexpected screen arrived during the keep Alive sleep time, a Map
    // Screen checks for the "launch" screen before executing Keep Alive actions
    MAP Screen TO ScreenDoc, if field count is not 118 then RAISEERROR
    SEND ATTENTION KEY ENTER
    // Use a Map Screen action again to make sure we get back the "launch" screen
    // (Note: it is skipped during animation to prevent 2 Map Screens in a row)
    MAP Screen TO ScreenDoc, if field count is not 118 then RAISEERROR
LOGOFF
  Log Off Actions
```

Logoff Actions

Logoff actions essentially navigate the User ID properly out of the host system after a timeout.

Logoff actions execute once for a given connection, and *only* when a connection times out (i.e. the Inactivity Lifetime expires) or the connection is closed via the 3270 Server console.

In a "best practices" sense, it's vitally important to make Logoff Actions bulletproof. If an exception occurs during execution of the Logoff actions, exteNd Composer will break its connection with the host, freeing the UserID in the pool. *But the UserID may still be active on the host*. Until the host kills the UserID (from inactivity), a subsequent attempt by the pool to log on with that UserID may fail, unless you've coded your logon to handle the situation. Logon failures cause the UserID to be discarded from the pool, reducing the potential pool size and performance overall. As with Logon and Keep Alive actions, the way to guarantee you are on the proper screen at the end of the logoff is to end with a Map Screen.

Logon Component Life Cycle

Each time a User ID is activated from the Logon Connection Pool, an instance of the corresponding Logon Component is created and associated with that User ID. Then the Logon actions are executed until the desired launch screen is reached. At this point the 3270 Terminal component execution begins. When it is finished another 3270 Terminal component using the same Logon Connection may begin executing, starting at the same launch screen.

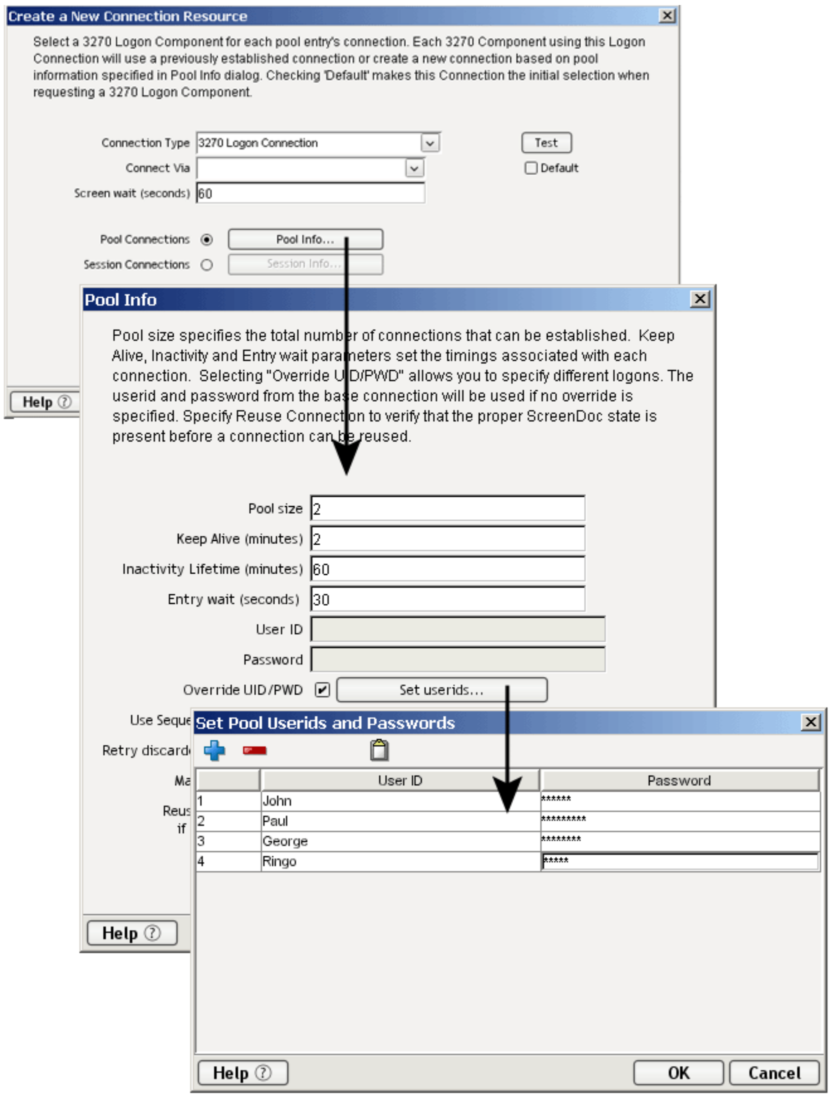
If no other component requests the connection, then the connection-instance in question enters an active but free state (an “idle state”) defined by the Inactivity Lifetime and KeepAlive settings on the Pool Info dialog of the Logon Connection. If the Keep Alive period (e.g., 2 minutes) is shorter than the Inactivity Lifetime (e.g., 120 minutes), then at appropriate (2-minute) intervals, the Keep Alive actions will be executed, preventing a host timeout and dropped connection; and the Keep Alive Period begins anew.

A Logon Component’s execution lifetime is dependent on the activity of the Logon Connection that uses it. As long as one entry in the Logon Connection pool is active, then one instance of the Logon Component will be in memory in a live state. A Logon Component instance will go out of scope (cease executing) when the last remaining pool entry expires due to inactivity. The only other way to stop execution of a Logon Component is through the 3270 Console on the Server.

About the 3270 Logon Connection

The Logon Connection is not a true connection object like a TN3270 Connection Resource, but a pointer to a Logon Component (which in turn connects to a host either through a conventional Connection Resource or yet more intervening Logon Connection/Logon Component pairs). The Logon Connection encapsulates information needed to describe a *pool of connections*. That includes User IDs and passwords, plus pool settings involving the time interval between retries on discarded connections, etc. Another function of the Logon Connection is that it ensures the use of different instances of the same Logon Component for all the User IDs for which connections are made.

The dialogs you’ll use in setting up a pool of User IDs for a Logon Connection are shown in the following set of illustrations. Arrows denote the buttons that lead to continuation dialogs.


Create a New Connection Resource

Select a 3270 Logon Component for each pool entry's connection. Each 3270 Component using this Logon Connection will use a previously established connection or create a new connection based on pool information specified in Pool Info dialog. Checking 'Default' makes this Connection the initial selection when requesting a 3270 Logon Component.

Connection Type: 3270 Logon Connection
Connect Via:
Screen wait (seconds): 60
Pool Connections ●
Session Connections ○

Pool Info

Pool size specifies the total number of connections that can be established. Keep Alive, Inactivity and Entry wait parameters set the timings associated with each connection. Selecting "Override UID/PWD" allows you to specify different logons. The userid and password from the base connection will be used if no override is specified. Specify Reuse Connection to verify that the proper ScreenDoc state is present before a connection can be reused.

Pool size: 2
Keep Alive (minutes): 2
Inactivity Lifetime (minutes): 60
Entry wait (seconds): 30
User ID:
Password:
Override UID/PWD ✓

Set Pool Userids and Passwords

	User ID	Password
1	John	******
2	Paul	*********
3	George	********
4	Ringo	*****

Every Logon Connection is associated with a given Logon Component. In addition, the Logon Connection provides the following User ID pool functionality:

1 It allows the specification of multiple User IDs in advance ensuring that clients are able to secure a connection when one is needed

2. It allows the reuse of a User ID/connection once it is established to eliminate repeated user authentications and disconnects
3. It allows a single User ID to use multiple connections if this is supported by the host system
4. It keeps a connection active to prevent host timeouts during inactive periods
5. It lets you specify when to remove a connection from the active pool
6. It sets a timeout period to use for a fully active pool to provide a free connection
7. It lets you specify error handling dependent on the state of the Logon Component used by the Logon Connection

Many-to-One Relationship of Components to Logons

In order for multiple instances of a 3270 Terminal component or different 3270 Terminal components to use a the same Logon Connection, the following conditions must be met:

1. All the 3270 Terminal components must use the same Connection Resource (thereby sharing the TN3270 Host, Port and data encoding parameters or EPI 3270 connection Gateway and Server parameters)
2. All the 3270 Terminal components must have a common launch screen in the host system from which they can begin execution (see “About the 3270 Logon Component” below for more detail).

Connection Pooling with a Single Sign-On

If your host system security supports multiple logins from a single user ID, you may have circumstances where you wish to pool the single User ID. This can be accomplished by performing the following steps:

- Specify a User ID/Password in the Connection Resource used by the Logon Component.
- On the Pool Info dialog of the Logon Connection, specify a Pool Size greater than 1.
- Do NOT check the **Override the UID/PWD** setting in the Pool Info dialog of the logon Connection.

These steps will cause each pool slot to use the User ID and Password contained in the Connection object and not use the user IDs from the pool.

Creating a Connection Pool

Overview

When creating a 3270 Terminal component, you normally first create the Connection object it needs. Similarly, when creating the objects comprising a Connection Pool, you must create certain objects first, starting (in essence) at the host and working your way backwards to the 3270 Terminal Component that will access the host.

A typical sequence of steps for creating a Connection Pool is:

- Create the host Connection
- Create the Logon Component that uses the Connection
- Create Logon Connection that uses the Logon Component
- Create one or more 3270 Terminal Components that use the Logon Connection

Creating a Connection

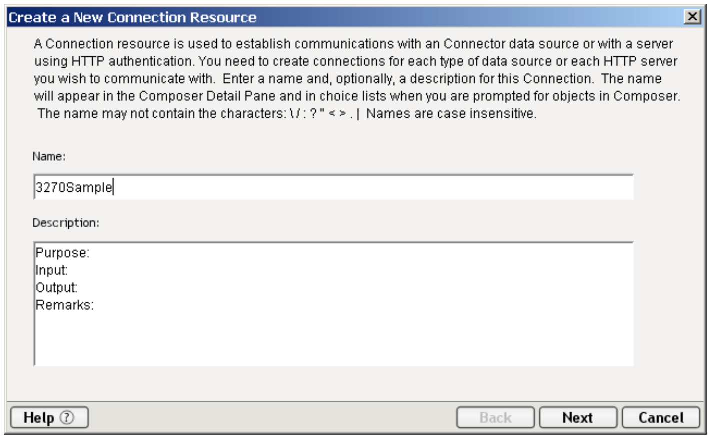
This step is simple. Create a new Connection Resource (TN3270 or EPI) as described in Chapter 2 of this Guide. Even though you will be using User IDs and Passwords defined in the Logon Connection later, you should still define one in the Connection as well. This will be needed when you define the Logon Component in the next step. Alternatively, you can simply use an existing Connection Resource.

Creating a Logon Component

➢ **To create a 3270 Logon Component:**

1 From the Composer **File** menu, select **New> xObject,** then select the **Component** tab and choose **3270 Logon**.

The Header Info panel of the New xObject Wizard appears.



A Connection resource is used to establish communications with an Connector data source or with a server using HTTP authentication. You need to create connections for each type of data source or each HTTP server you wish to communicate with. Enter a name and, optionally, a description for this Connection. The name will appear in the Composer Detail Pane and in choice lists when you are prompted for objects in Composer. The name may not contain the characters: \ / : ? " < > . | Names are case insensitive.

Name:

3270Sample

Description:

Purpose:
Input:
Output:
Remarks:

2 Type a **Name** for the connection object.

3 Optionally, type **Description** text.

4 Click **Next** and the **Connection Info** panel appears.

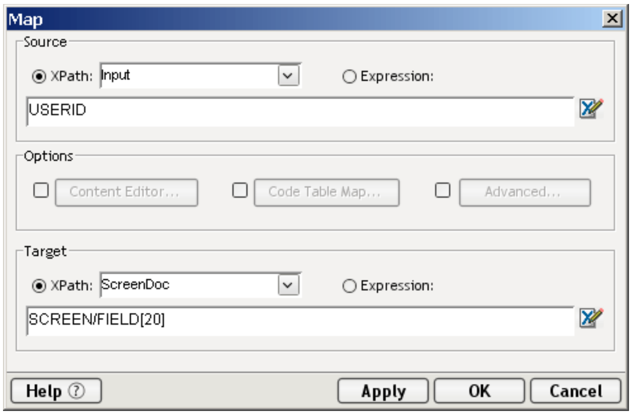
5 Select a Connection from the drop down list.

6 Click **Finish** and the Logon Component Editor appears.

NOTE: Recording actions follows a series of steps. The cursor must be positioned over Logon, turn record on, when you are done, turn Record off. Position the cursor to Keep Alive, turn Record on, when you are done, turn Record off. Position the cursor to Logoff, turn record on, when you are done, turn record off.

7 Record Logon Actions for logging into the host and navigating to the launch screen using the same Recording techniques described in Chapter 4 of this Guide.

8 Edit the Logon Map actions that enter a User ID and Password to instead use the special USERID and PASSWORD variables described in the section titled “3270 Specific Expression Builder Extensions” in Chapter 4 of this Guide.



Source: XPath: Input

USERID

Target: XPath: ScreenDoc

SCREEN/FIELD[20]

9 Create the needed SEND Key actions in the Keep Alive section of the Action Model (a quick way is to copy an existing SEND key action, Paste it, and then modify the key code sent).

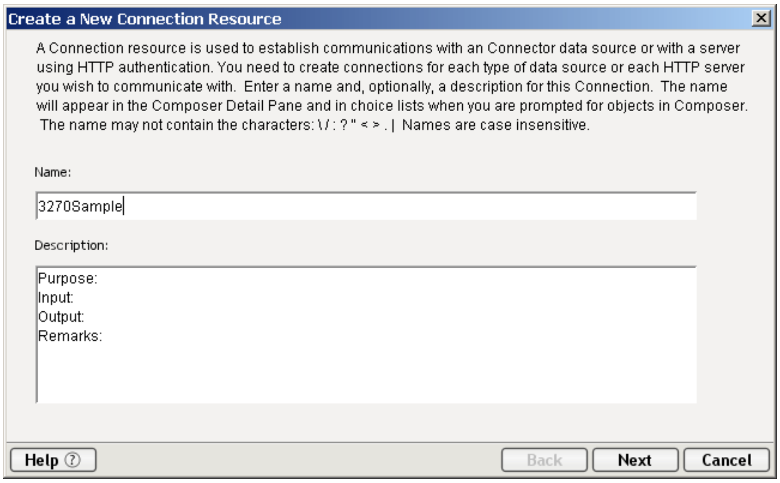
10 Record Logoff actions for properly exiting the host.

11 **Save** and **Close** the Logon Component.

Creating a Logon Connection using a Pool Connection

➢ **To create a 3270 Logon Connection:**

1 From the Composer **File** menu, select **New> xObject,** then open the **Resource** tab and select **Connection**, or you can click on the icon. The Header Info panel of the New xObject Wizard appears.



A Connection resource is used to establish communications with an Connector data source or with a server using HTTP authentication. You need to create connections for each type of data source or each HTTP server you wish to communicate with. Enter a name and, optionally, a description for this Connection. The name will appear in the Composer Detail Pane and in choice lists when you are prompted for objects in Composer. The name may not contain the characters: \ / : ? " < > . | Names are case insensitive.

Name:

3270Sample

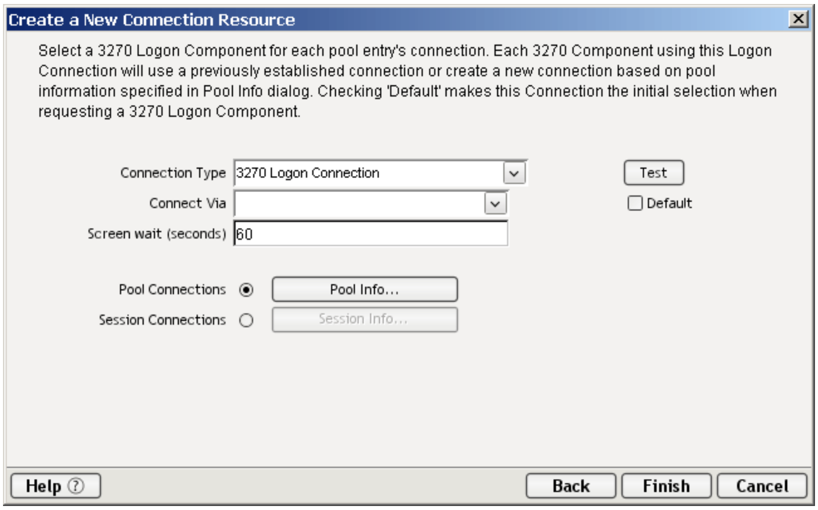
Description:

Purpose:
Input:
Output:
Remarks:

2 Type a **Name** for the connection object.

3 Optionally, type **Description** text.

4 Click **Next** and the **Connection Info** panel appears.



Create a New Connection Resource

Select a 3270 Logon Component for each pool entry's connection. Each 3270 Component using this Logon Connection will use a previously established connection or create a new connection based on pool information specified in Pool Info dialog. Checking 'Default' makes this Connection the initial selection when requesting a 3270 Logon Component.

Connection Type: 3270 Logon Connection

Connect Via:

Screen wait (seconds): 60

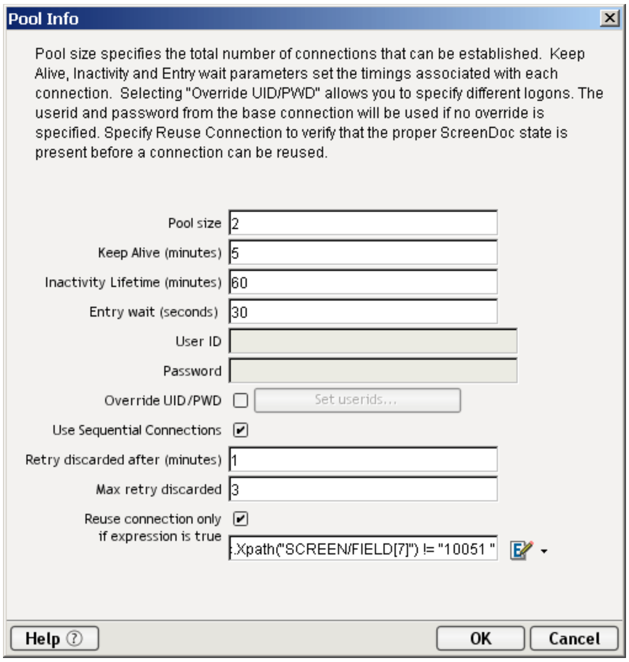
[ ] Default

[● Pool Connections] ○ Session Connections

5 For the Connection Type select “3270 Logon Connection” from the drop down list.

6 In the **Connect Via** control, select the Logon Component you just created.

7 Click on the Pool Info button and the Pool Info dialog appears.



Pool size specifies the total number of connections that can be established. Keep Alive, Inactivity and Entry wait parameters set the timings associated with each connection. Selecting "Override UID/PWD" allows you to specify different logons. The userid and password from the base connection will be used if no override is specified. Specify Reuse Connection to verify that the proper ScreenDoc state is present before a connection can be reused.

Pool size	2
Keep Alive (minutes)	5
Inactivity Lifetime (minutes)	60
Entry wait (seconds)	30
User ID	
Password	
Override UID/PWD	☐ Set userids...
Use Sequential Connections	☑
Retry discarded after (minutes)	1
Max retry discarded	3
Reuse connection only if expression is true	☑ .Xpath("SCREEN/FIELD[7]") != "10051 "

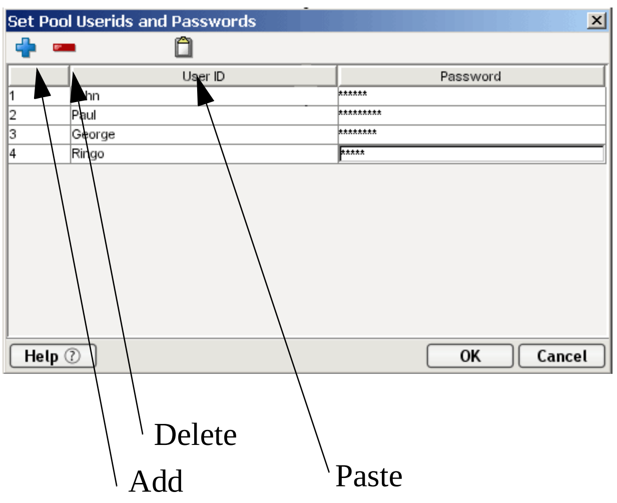
8 Enter a **Pool Size** number. This represents the total number of connections you wish to make available in this pool. For each connection, you will be expected to supply a UserID/Password combination later.

9 Enter a **KeepAlive** time period. This number represents (in minutes) how often you wish to execute the Keep Alive actions in the associated Logon Component whenever the connection is active but free (i.e. not being used by a 3270 Terminal component). The number you enter here should be less than the Timeout period defined on the host for an inactive connection.

10 Enter an **Inactivity Lifetime**. This number represents (in minutes) how long you wish to keep an active free connection available before closing out the connection and returning it to the inactive portion of the connection pool. Remember, that once the connection is returned to its inactive state in the pool, it will incur the overhead of logging in and navigating host screens when it is re-activated.

11 Enter an **Entry Wait** time in seconds. This time represents how long a 3270 Terminal component will wait for a free connection when all the pool entries are active and in use. If this time period is reached, an Exception will be thrown to the Application Server.

12 Checking **Override UID/PWD** means you wish to specify User ID/Password combinations for use in the connection pool. When checked, this activates the Set USERIDs button. Click on the button to display the Set USERIDs and PASSWORDS dialog.



	User ID	Password
1	John	******
2	Paul	*********
3	George	********
4	Ringo	*****

On the Toolbar there are three icons: Add which adds an empty row, Delete, which deletes a highlighted row and Paste which allows you to copy/paste information from a spreadsheet into the table.

NOTE: Alternate and faster ways to enter data are to copy data from a spread sheet and paste it into the table. Make sure your selection contains two columns. The first column must contain UserID; the second Password. Open the spreadsheet, copy the two columns and as many rows as needed. Open the table and immediately press the Paste icon located on the toolbar. You can also copy data from tables in a Microsoft Word® document using the same technique.

13 Enter as many **USERID/PASSWORD** combinations needed, until you reach the size of the pool you specified, then click **OK**. (If you enter more, then the pool size will automatically adjust to the new size.)

14 Click **OK** to dismiss the “Set User IDs and Passwords” dialog and return to the Pool Info dialog.

15 Optionally check the **Pool Resource Names** checkbox in the Pool Info dialog, if you are using 3270E (extended protocol) and intend to manage terminals by identifier strings. (Checking the checkbox will enable the **Set Device Names...** button. Click the button to bring up the “Set Resource Names” dialog. Then enter terminal IDs in the dialog and click **OK**.)

16 Optionally click the **Use Sequential Connections** checkbox if you want Composer to establish connections in the same order that User IDs were listed in the “Set User IDs and Passwords” dialog. Connections will be made in numerical sequence.

17 In the **Retry discarded after** field, enter a value representing the number of minutes to wait between connection retries (attempts to reestablish a connection).

18 In the **Max retry discarded** field, enter a value representing the number of connection attempts to try before giving up (which means permanently discarding the connection).

19 Optionally check the **Reuse connection only if expression is true** control. This control allows you to enter an ECMAScript expression that evaluates to true or false based on some test of the launch screen. The purpose of the expression is to check to make sure the launch screen is the proper one each time a new 3270 Component is about to reuse an active free connection. Under circumstances unrelated to your Composer service, it’s possible that the launch screen will be replaced by the host with a different screen. For instance, if there is a system ABEND on the host, the launch screen in the Logon Component may be replaced by a System Message screen.

NOTE: For instructions on how to create this expression, see the discussion at “Handling System messages” in Chapter 2 of this Guide. Also refer to “Maximizing Performance of the 3270 Logon Connection” in this Chapter.

The following a is a sample Custom Script used to see if a particular screen is present. If it is not, the script writes a message to the console stating that the screen is bad and the logon connection is being released. This function is called from the “Reuse connect only if expression is true” control on the Pool Info dialog.

```
function checkValidLaunchScreen(ScreenDoc)
{
       var screenText = ScreenDoc.XPath("SCREEN").item(0).text
       if((screenText.indexOf( "MENU") != -1 || screenText.indexOf("APLS") != -1) &&
          (screenText.indexOf("COMMAND UNRECOGNIZED") == -1 ||
             screenText.indexOf("UNSUPPORTED FUNCTION") == -1))
        {
            return true;
        }
        else
        {
             java.lang.System.out.println("Warning - Releasing logon connection at bad screen");
             java.lang.System.err.println("Warning - Releasing logon connection at bad screen");
            return false;
        }
}
```

20 Click **OK** to return to the Connection Info panel.

21 Click **Finish** and the Logon Connection is saved.

Maximizing Performance of 3270 Logon Connection

To prevent 3270 Components from beginning execution on a connection that may have been left on an invalid screen by a previous 3270 component, the Logon Connection Resource allows the connection itself to check for the presence of the launch screen. This is accomplished by using the option titled “Reuse connection only if expression is true” on the Pool Info dialog of the Logon Connection. The screen test you specify here is executed each time a 3270 Component completes execution. If the test fails, exteNd Composer will immediately disconnect from the host, possibly leaving a dangling UserID on the host. As noted before, the host will eventually kill the user, but the UserID may be discarded from the pool if it is accessed again before being killed, thereby reducing the pool size and consequently overall performance.

Another reason to use the “Reuse connection only if true” option is that you can perform very detailed tests against the screen to make sure it is your launch screen. While Map Screen actions do perform a screen check, they only look at the number of fields in the terminal data stream. In most cases, this is sufficient. However, it is possible two different screens can have the same number of fields in which case the expression based test that examines the content of the screen will produce more rigorous results. A best practices approach mandates that you use this feature all the time.

Static versus Dynamically Created Documents/Elements

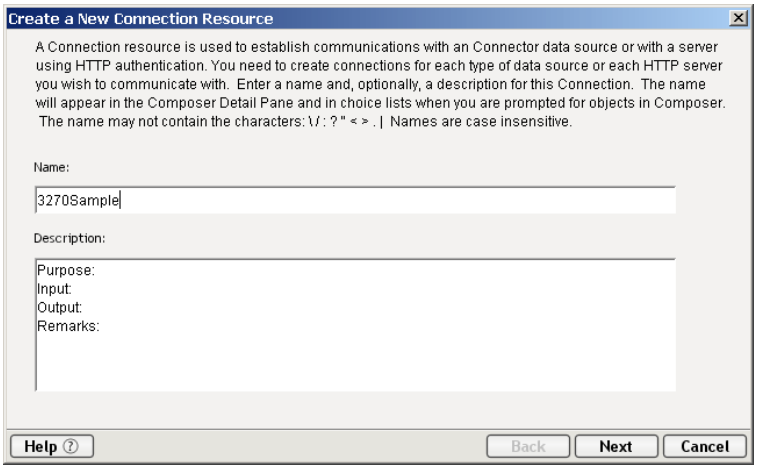
In some Composer applications, users have a need to place various control, auditing, and/or meta-data in an XML document. This document may or may not be in addition to the actual elements/documents being processed (i.e. created from an information source). If this document structure and data is dynamically created by multiple Map actions (i.e. over 100) performance of the component and therefore the entire service may suffer. To boost performance, create the portion of the document structure without the dynamic content ahead of time, then load it into the Service at runtime via an XML Interchange action and retain the Map actions for dynamic content. This can boost performance as much as 30% in some cases.

Creating a Logon Connection using a Session Connection

Sometimes, you may want the extra level of control over session parameters that a Logon Connection affords, without necessarily wanting to use pooling. In this case, you can follow the procedure outlined below.

➢ To create a 3270 Logon Connection:

1 From the Composer **File** menu, select **New> xObject,** then select the **Resource** tab and choose **Connection**, or you can click on the icon. The Header Info panel of the New xObject Wizard appears.



A Connection resource is used to establish communications with an Connector data source or with a server using HTTP authentication. You need to create connections for each type of data source or each HTTP server you wish to communicate with. Enter a name and, optionally, a description for this Connection. The name will appear in the Composer Detail Pane and in choice lists when you are prompted for objects in Composer. The name may not contain the characters: \ / : ? " < > . | Names are case insensitive.

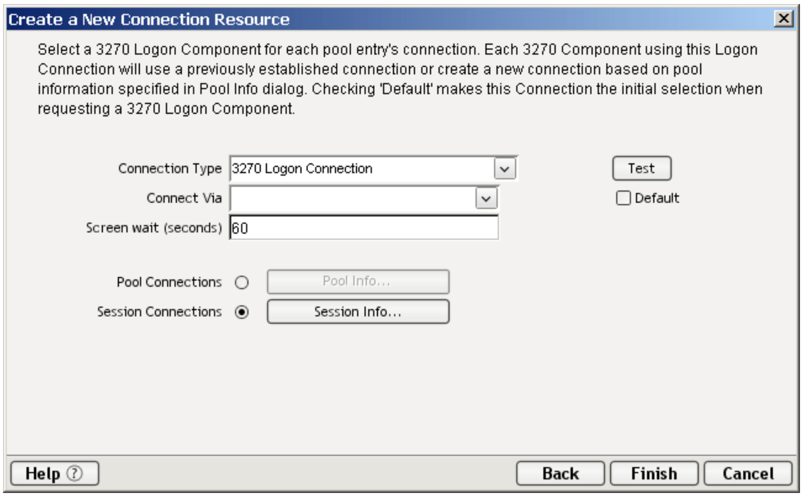
Name: 3270Sample

Description:
Purpose:
Input:
Output:
Remarks:

2 Type a **Name** for the connection object.

3 Optionally, type **Description** text.

4 Click **Next** and the **Connection Info** panel appears.



Select a 3270 Logon Component for each pool entry's connection. Each 3270 Component using this Logon Connection will use a previously established connection or create a new connection based on pool information specified in Pool Info dialog. Checking 'Default' makes this Connection the initial selection when requesting a 3270 Logon Component.

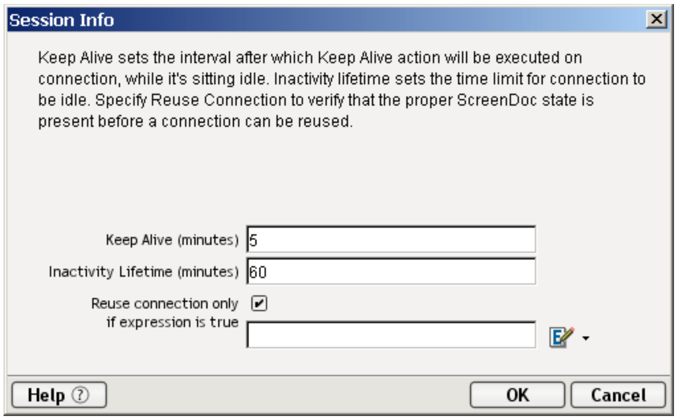
Connection Type: 3270 Logon Connection
Connect Via:
Screen wait (seconds): 60
☐ Default
Pool Connections ○
Session Connections ●

5 For the Connection Type select “3270 Logon Connection” from the drop down list.

6 In the **Connect Via** control, select the Logon Component you just created.

7 The **Screen wait (seconds)** field, displays the amount of time in seconds that a 3270 Terminal component will wait for a the next screen to arrive in the Map Action Pane.

8 Click the **Session Connections** radio button and then on Session Info button.



Keep Alive sets the interval after which Keep Alive action will be executed on connection, while it's sitting idle. Inactivity lifetime sets the time limit for connection to be idle. Specify Reuse Connection to verify that the proper ScreenDoc state is present before a connection can be reused.

Keep Alive (minutes) 5

Inactivity Lifetime (minutes) 60

Reuse connection only if expression is true ✓

9 The Keep Alive (minutes) number represents (in minutes) how often you wish to execute the Keep Alive actions in the associated Logon Component whenever the connection is active but free (i.e. not being used by a 3270 Terminal component). The number you enter here should be less than the Timeout period defined on the host for an inactive connection.

10 The Inactivity Lifetime (minutes) number represents (in minutes) how long you wish to keep an active free connection available before closing out the connection and returning it to the inactive portion of the connection pool. Remember, that once the connection is returned to its inactive state in the pool, it will incur the overhead of logging in and navigating host screens when it is re-activated.

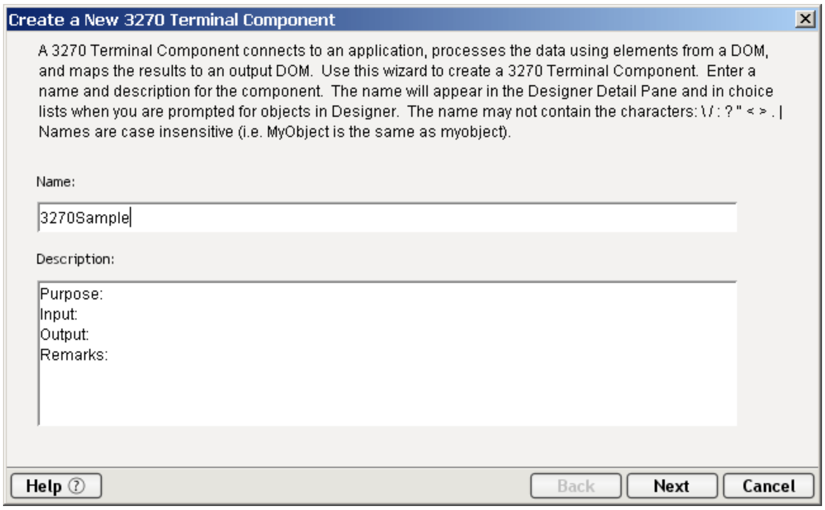
11 Click in the checkmark box if you want to Reuse connection only if expression is true. If you choose to do so, the expression field automatically displays and you can click on the expression icon to display the if the expression is true dialog.

Creating a 3270 Terminal Component That Uses Pooled Connections

At this point, you are ready to create a 3270 Terminal Component that can use the Connection Pool. For the most part, you will build the component as you would a normal 3270 Terminal component, the only difference being the Connection you specify on the connection panel of the New Component Wizard. (You'll specify a Logon Connection instead of a regular TN3270 or EPI Connection.)

➢ **To create a 3270 Terminal Component:**

1 From the Composer **File** menu, select **New>xObject,** then select the **Component** tab and choose **3270 Terminal**.The Header Info panel of the New xObject Wizard appears.



A 3270 Terminal Component connects to an application, processes the data using elements from a DOM, and maps the results to an output DOM. Use this wizard to create a 3270 Terminal Component. Enter a name and description for the component. The name will appear in the Designer Detail Pane and in choice lists when you are prompted for objects in Designer. The name may not contain the characters: \ / : ? " < > . | Names are case insensitive (i.e. MyObject is the same as myobject).

Name:

3270Sample

Description:

Purpose:
Input:
Output:
Remarks:

2 Type a **Name** for the component.

3 Optionally, type **Description** text.

4 Click **Next** and the **XML Property Info** panel appears.

5 Select the necessary Input and Output Templates and click **Next**, and the Temp/Fault panel appears.

6 Select the necessary Temp and Fault Templates and click **Next**, and the Connection Info panel appears.

7 Select the Logon Connection you created and click on **Next**, and the Component editor appears.

8 Build the component according to instructions in Chapter 3 of this Guide.

Maximizing Performance of 3270 Terminal Components

Once the launch screen is obtained by the Logon Component's logon actions, it is handed to the 3270 Terminal Component that uses the connection. Then the 3270 Terminal component (when finished executing) leaves the screen handler back at the launch screen. If the 3270 Component finishes without being on the launch screen (i.e. it releases the connection back to the pool with an invalid screen), then it is possible that all subsequent 3270 Components that use the connection may throw exceptions, rendering the connection useless. It also will degrade overall performance and possibly cause data integrity problems within the component processing.

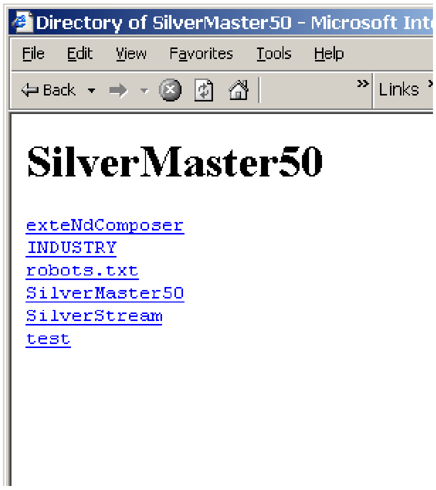
Once again, ensure that the launch screen is present. *The last action to execute in a 3270 Component must be a Map Screen that checks for the launch screen.* This can be tricky if your component has many decision paths that may independently end component execution. You must be sure that each path ends with a Map Screen action.

Managing Pools

Connections pools can be managed through the 3270 Console Screen.

➢ How to Access the Console

1 If you are using the Novell exteNd Application Server, log on to your server via your web browser using **http://localhost/SilverMaster50** (or whatever is appropriate for the version in use). In this example, Novell exteNd App Server 5.0 is used.


Directory of SilverMaster50 - Microsoft Int
File Edit View Favorites Tools Help
Back
Links

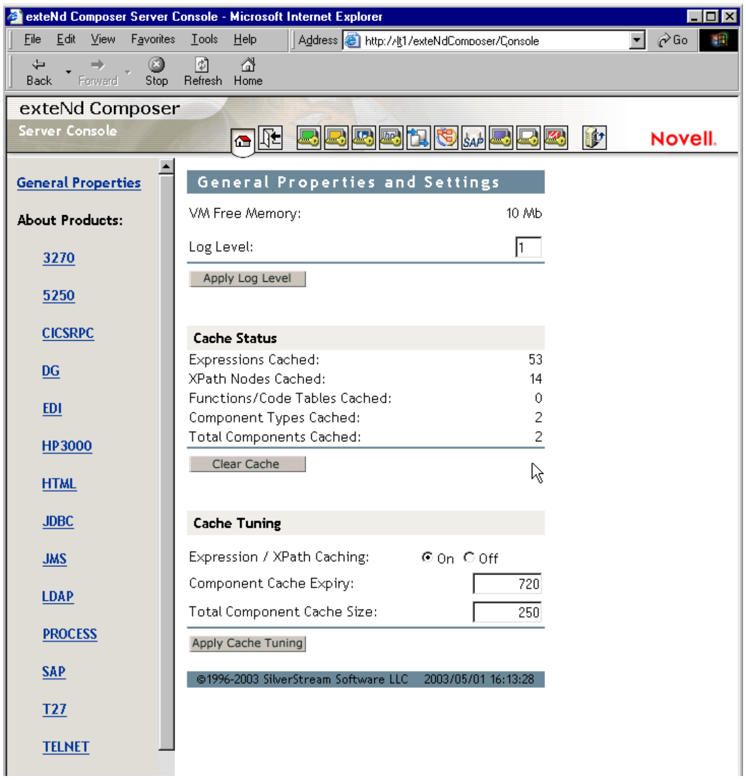
SilverMaster50

exteNdComposer
INDUSTRY
robots.txt
SilverMaster50
SilverStream
test

NOTE: If you are not using the exteNd app server, enter a URL of this form:

http://<hostname>:<port>/exteNdComposer/Console

2 Click on the **exteNd Composer** link. You should see the main console page:



General Properties and Settings

VM Free Memory:	10 Mb
Log Level:	1

Cache Status

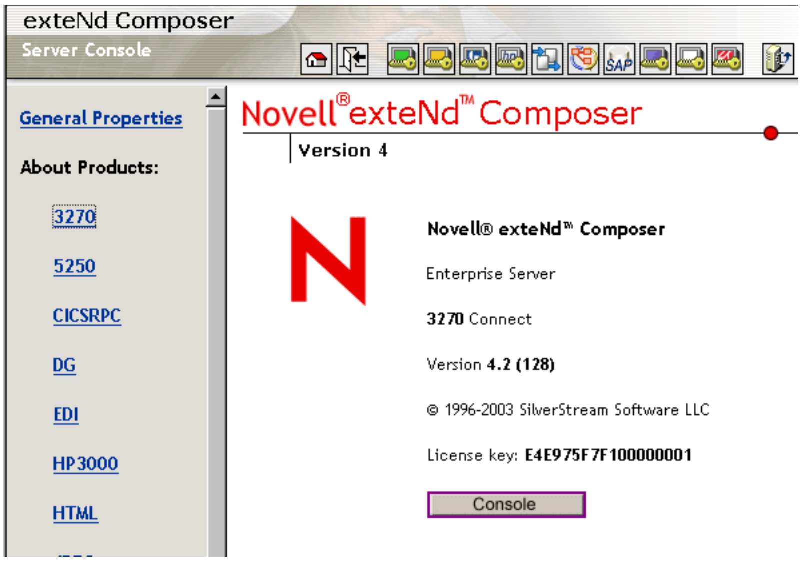
Expressions Cached:	53
XPath Nodes Cached:	14
Functions/Code Tables Cached:	0
Component Types Cached:	2
Total Components Cached:	2

Cache Tuning

Expression / XPath Caching:	[✓ On] Off
Component Cache Expiry:	720
Total Component Cache Size:	250

©1996-2003 SilverStream Software LLC 2003/05/01 16:13:28

3 Click on the 3270 link in the left (nav) frame. The 3270 Console General Properties Screen will come into view.


exteNd Composer

Server Console

General Properties

About Products:

3270

5250

CICSRPC

DG

EDI

HP3000

HTML

Novell® exteNd™ Composer

Version 4

Novell® exteNd™ Composer

Enterprise Server

3270 Connect

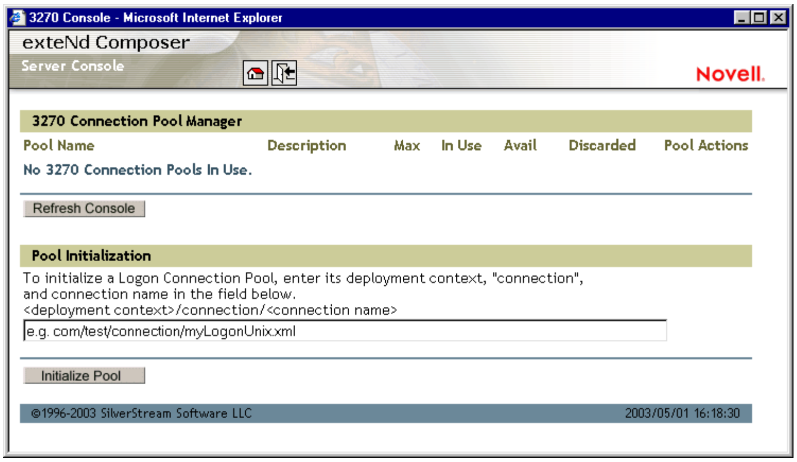
Version 4.2 (128)

© 1996-2003 SilverStream Software LLC

License key: E4E975F7F100000001

Console

4 Click the Console button. A browser popup window (the Connection Pool Management Screen) should appear:


3270 Console - Microsoft Internet Explorer

exteNd Composer

Server Console

Novell.

3270 Connection Pool Manager

Pool Name	Description	Max	In Use	Avail	Discarded	Pool Actions
No 3270 Connection Pools In Use.						

Refresh Console

Pool Initialization

To initialize a Logon Connection Pool, enter its deployment context, "connection", and connection name in the field below.
<deployment context>/connection/<connection name>

e.g. com/test/connection/myLogonUnix.xml

Initialize Pool

©1996-2003 SilverStream Software LLC 2003/05/01 16:18:30

5 To initialize a Logon Connection Pool, enter its deployment context, the word "connection", and the actual connection name in the text field near the bottom of the screen. (See illustration above.) Then click the **Initialize Pool** button.

NOTE: Refer to the appropriate Composer Enterprise Server guide for more information.

6 Optionally click the **Refresh Console** button to update the view.

Connection Pool Management and Deployed Services

The Connection Pool Management Screen displays the current state of the connection(s) with the 3270 Connect. The screen contains a table listing the Pool Name, Description of the connection, the maximum number of connections in the pool, the number of connections in use, the number of connections available, the number of connections discarded. It also contains several buttons allowing you to perform various actions related to connection pooling, which are outlined in the table below.

Button Name	**Action**
Reset Discarded	Resets the Discarded connections which are then reflected in the table
Reset (Pool)	Resets the Available and Discarded connections which are then reflected in the table
Refill (Pool)	Refills the pool with the maximum number of connections
Additional Buttons on 3270 Connection Pool Manager Console	
Refresh Console	Shows the current status of the connection pool
Initialize Pool	Initializes a Logon Connection Pool by entering a relative path to the deployed lib directory. This will not work unless the deployed jar is extracted. Click on the SUBMIT button when finished.

Connection Discard Behavior

The performance benefits of connection pooling are based on the ability of more than one user to access a resource, or set of resources, at once. The way a connection is established begins with the logon component picking the User ID and Password from the table. If the connection fails, then it is discarded for this User ID and Password *and another is tried until a connection is established.* The failure of *one* connection doesn't necessarily prevent a successful connection from being established.

The Connect for 3270 addresses the “one bad apple” problem by discarding any connection that can’t be established (for whatever reason: bad user ID, timed-out password, etc.) and reusing the others. When a connection is determined to be unusable, the Connect for 3270 will write a message to the system log that says: “Logon connection in pool <Pool name> was discarded for User ID <User ID>.”

Screen Synchronization

Screen synchronization has special ramifications for users of pools. If a situation arises in which a user leaves a connection without the screen returning to its original state, the next user will begin a session with the screen in an unexpected state and an error will occur. To prevent this, we have a screen expression which the user can specify in the connection pool. It is important that the last action in a 3270 Component be a Send Attention Key action that will result in the session ending with the correct logon screen active.

NOTE: The last action should be an empty Map Screen action so that the 3270 Terminal component waits until the launch screen arrives before giving up the connection. (This should happen automatically, when you create the Send Attention Key action, but nevertheless, the last action should be the Map Screen.)

If you want to check, at runtime, for the presence of a bad screen at the end of a user session, include a Function Action at the end of your component’s action model that executes a function similar to the one shown below:

```
function checkValidReleaseScreen(ScreenDoc)
{
       var screenText = ScreenDoc.XPath("SCREEN").item(0).text
       if((screenText.indexOf( "MENU") != -1 ||
screenText.indexOf("APLS") != -1) &&
          (screenText.indexOf("COMMAND UNRECOGNIZED") == -1 ||
screenText.indexOf("UNSUPPORTED FUNCTION") == -1))
        {
            return true;
        }
        else // Write error messages to
             // System.out and System.err:
        {
java.lang.System.out.println("Warning - Releasing logon
connection at bad screen");
java.lang.System.err.println("Warning - Releasing logon
connection at bad screen");
            return false;
        }
}
```

In this particular example, this function checks the screen text and returns *true* if the screen contains “MENU” or “APLS” and does *not* contain “COMMAND UNRECOGNIZED” nor “UNSUPPORTED FUNCTION.” Otherwise it returns *false.*

A Testing

Environmental Differences between Animation Testing and Deployment Testing

There are significant environmental differences between Animation testing in Composer and Deployment testing. Both types of testing are needed to adequately verify the components and services you build. The differences are detailed in the table below.

Table A-1

	Testing in Composer	Deployment Testing
OS	Win98 or WinNT or Win 2000.	WinNT or Sun Solaris.
Platform	JRE (Java Runtime Environment).	Application Server complete with JRE support for Failover, Security, Connection Mgt., etc.
Component or Service Startup	Directly from Composer.	By Service Triggers only (i.e., deployment Servlets or EJBs).
xObject access	From disk files.	From a JAR file in Application Server.
Runtime Context	Test individual components or components running within a service.	Always from within a service.

	Testing in Composer	Deployment Testing
Service and Component Inputs	Input documents frequently come from sample XML documents on the local machine as well as DOMs from other services or components.	Input documents are passed into the services and components via Service Triggers, or DOMs from other services or components.
Project Variables for: * Log File Paths * DTD URIs * XSL URIs * Send Mail Server * XML Inter-change URIs	Usually point to locations on local machine (but could be on Servers or Web).	Should point to locations on production Servers and Web.
Testing Tools	In addition to Log actions, you can use dialog boxes (ECMAScript alert() function) to display runtime values.	No dialog boxes can be used.
JDBC Connection	Doesn't use Server Connection Pools – May be using Test Databases.	Uses Server-provided Connection pools – Should be using Production Databases.
HTTP Connections	May be pointing to local machine or test servers.	Should be pointing to test or production servers.
3270 Connections	Usually point to test systems and test UserIds and Passwords.	Should point to production systems and production UserIds and Passwords.
5250 Connections	Usually point to test systems and test UserIds and Passwords.	Should point to production systems and production UserIds and Passwords.
CICS(ECI) - RPC Connections	Usually point to test systems and test UserIds and Passwords.	Should point to production systems and production UserIds and Passwords.

B Java Code Pages

About Encodings

exteNd Composer’s ability to perform character encoding conversions is tied directly to the Java VM in use. The supported encodings vary between different implementations of the Java 2 platform. Sun's Java 2 Software Development Kit, Standard Edition, v. 1.2.2 for Windows or Solaris and the Java 2 Runtime Environment, Standard Edition, v. 1.2.2 for Solaris support. The encodings can be found at the Sun web page:

http://java.sun.com/products//jdk/1.2/docs/guide/internat/encoding.doc.html

Sun's Java 2 Runtime Environment, Standard Edition, v. 1.2.2 for Windows comes in two different versions: US-only and international. The international version (which includes the lib\i18n.jar file) supports all encodings in both tables.

C 3270 Glossary

AID Key

Any of the following 3270-supported keys:

Table C-1

3270 Key	PC Key
Enter	Enter
Clear	ESC
PF1 through PF12	F1 through F12
PF13 through PF24	Shift F1 through Shift F12
PA1 through PA3	Ctrl F1 through Ctrl F3

Field

A unit of data contained in a TDS. A field may be a label to display on the screen, an item of data, or special blank non-display fields. Each field has its own attributes that determine how it is displayed and if the area can be modified.

Native Environment Pane

A pane in the 3270 Component Editor that provides an emulation of an actual 3270 terminal session.

Map Screen Action

A special non-editable action that indicates the location in an Action Model where a new terminal data stream (TDS) screen is received. Any actions intended to interact with this screen must be placed subordinate to the Map Action's Screen Actions line in the Action Model.

Multi Row Transform

A special action for mapping repeating rows of data within a 3270 screen between a DOM and the 3270 screen. This action also transforms the ScreenDom and creates a hierarchical view of the screen. This action along with a Multi Row Group action, a Decision action, and additional Repeat actions are created by the Multi Row Wizard. Special care should be taken when editing any of these actions.

ScreenDoc

A special DOM in the 3270 (and 5250) component editor windows representing the current 3270 screen display as an XML document.

Send Attention Key

An action that appears in the Action Model whenever an AID key is pressed.

TDS

Terminal Data Stream

D Reserved Words

The following terms are reserved words in exteNd Composer 3270 Connect and should be avoided in any user-created labels or scripts.

- USERID
- PASSWORD
- PART
- MENU
- AID key
- SEND ATTENTION KEY
- MULTIROW

Index