PeopleSoft

EnterpriseOne AutoPilot 8.9 PeopleBook

September 2003

EnterpriseOne
AutoPilot 8.9 PeopleBook
SKU REL9EUA0309

Table of Contents

Scripting Actions 48

AutoPilot Overview

AutoPilot is an automated testing tool that you can use to create scripts to test the execution of J.D. Edwards ERP applications and to perform repetitive tasks, such as loading data, entering sales orders, or performing screen captures. You create the scripts by using AutoPilot to write commands that run essential functions and processes, such as:

- Launching applications
- Launching forms
- Executing form interconnections
- Running UBEs
- Setting processing options for interactive applications and for UBEs
- Entering data in header controls
- Entering data in grid columns
- Entering data in QBE lines
- Pushing toolbar buttons
- Clicking pushbuttons and bitmaps
- Selecting grid lines
- Performing database validations
- Selecting combo boxes
- Traversing tree paths

AutoPilot has the flexibility to run scripts on an NT platform, a Java platform, or in HTML. The tool has this flexibility because it reads and loads the specifications for each operation that you perform and passes the data through the operating system to J.D. Edwards ERP as a keyboard input. Therefore, you can use the script to test different operating systems, environments, and data mappings without making changes to the script.

AutoPilot's flexibility also allows you to do the following:

- Save scripts on your local drive or in a script repository that is shared by others.
- Run scripts in a stand-alone mode or with included scripts .
- Derive values for input in your scripts from a variety of sources, including literal values, valid value lists, visual assists, and variables, which you create to store values that you can easily change as needed.
- Pass variable values within a single script or among multiple scripts.
- Test the integrity of data that you add by performing a database validation.
- Capture data about J.D. Edwards ERP error and warning messages and about script playback events, including API calls.
- Send scripts to and receive scripts from others.

In summary, AutoPilot offers the following advantages to those wanting to write scripts that test key business processes:

- Decreases the time and effort required to create automated test scripts
- Ensures that the tool remains viable despite changes in J.D. Edwards ERP software because it reads and loads J.D. Edwards ERP specifications directly
- Allows users to write scripts that are compatible with future releases of J.D. Edwards ERP software
- Presents a user interface that disguises the complexity of the inner workings of the tool
- Provides the user flexibility to customize scripts by changing, for example, Object Configuration Manager (OCM) mappings
- Possesses the ability to work successfully with changing technologies

AutoPilot contains the following components, each of which helps you create scripts:

- Command pane, where you make choices to define the processes that you want to run in J.D. Edwards ERP software, such as launching an application
- Insert button, which allows you to insert a command to your script and create a script object that defines what the command does
- Script pane, which contains a running log of the commands that you have inserted into your script
- Tool bar, which allows you to navigate the AutoPilot form and to resize it to your specifications
- Menu bar, which contains the options that you need to run AutoPilot
- Caption bar, which identifies the script on which you are working
- Status bar, which displays information about an AutoPilot session, including processes that AutoPilot is running and brief definitions of AutoPilot commands

Note

All tasks in the *AutoPilot Guide* assume that you have opened AutoPilot and that you have either started a new script or opened an existing script.

Before You Begin

- ❑ You should have a working knowledge of common J.D. Edwards ERP software concepts, which you can find in the *Foundation Guide.*
- ❑ You should also have a good understanding of at least one J.D. Edwards ERP system, such as Accounts Payable or Sales Order Entry.

Upgrading from Service Pack 13 or Later

If you have a previous version of OneWorld Scripting Tool (also known as AutoPilot), and you upgrade to OneWorld Service Pack 13 or later, you should uninstall AutoPilot by deleting all of the files in your AutoPilot directory. Do not delete the directory itself, since it contains all of your scripts, valid value lists, and so on. After you upgrade to SP13 or later, change your desktop icon so that it points to *X: \b7\system\bin32\autopilot.exe*, where *X* is the drive on which OneWorld Xe is installed.

Note

Your Open Database Connectivity (ODBC) settings should remain valid after you upgrade. If you need to reconfigure your ODBC settings for AutoPilot, refer to the installation guide for your platform.

AutoPilot User Interface

You work with panes and bars in the AutoPilot form to write commands that make a script. The form consists of two panes: the command pane and the script pane. The command pane is the area in which you make choices that create commands. As you make the choices and insert them to create a script, AutoPilot displays the script as command lines in the script pane, where you can move, delete, and edit commands. The form also contains four bars: the caption bar, menu bar, tool bar, and status bar, all of which assist you in creating and identifying the script.

As you work in the AutoPilot form, you can also change its shape, size, and location on the desktop for ease of use. If you are working with more than one script, you can arrange child forms within the parent form by clicking options on the menu bar. Finally, you can move the tool bar to the most convenient position within the form, or you can detach the tool bar and move it to any position within the form or move it to your desktop.

Opening the AutoPilot Form

When you click the desktop icon for AutoPilot, a splash screen appears, followed by the AutoPilot form. Using this form, you create scripts to test J.D. Edwards ERP applications and carry out repetitive tasks. The AutoPilot form initially is blank.

► To open an AutoPilot form for scripting

1. From your desktop or the appropriate directory, launch AutoPilot.
2. Choose File from the AutoPilot menu bar.
3. Click New.

   The AutoPilot form, with some of the tool bar buttons activated, appears.

   The tool bar, which is located directly beneath the menu bar, contains buttons that represent the various commands, such as Application, that you can run in AutoPilot. When you pass the cursor over one of these buttons, or over one of the names in the drop-down menu under Command in the menu bar, words that identify the command appear in the status bar, which is located at the bottom of the AutoPilot form.

   When you place the cursor arrow on the splitter bar, you can change the size of the command or script pane by holding down the mouse button and pulling the bar up or down. Notice that when you initiate an AutoPilot session, the command pane is blank. You make the command pane active by initiating a command, such as Application. You find the names of the commands by clicking Command in the menu bar.

Panes in the AutoPilot Form

The two major components of the AutoPilot form are the command pane and the script pane.

The command pane is the top pane of the form and is divided into lists, from which you make choices that create commands that AutoPilot runs in J.D. Edwards ERP software. The

command pane also contains the Insert button, which you click to insert a command to the script.

The command pane allows you to make the choices that define a particular script of commands that AutoPilot runs in J.D. Edwards ERP software. The commands that you insert appear sequentially as command lines in the script pane. During or after script creation, you can move, edit, or delete the command lines that you have inserted to the script.

The script pane is the bottom pane of the form. It displays a running log and detailed description of the commands that you insert in the script. From the script pane, you can point the insertion cursor, which appears as a red arrow, to any spot in the script in which you want to insert a new command. You can also reorder the script using the mouse to drag and drop command lines, and you can edit command lines by using the mouse to highlight them.

The following table describes each of the panes and the specific components that you use to accomplish script-writing tasks:

Pane	Component	Purpose
Command	Lists	Make choices from or type entries in these populated and unpopulated areas to define a command that AutoPilot runs in J.D. Edwards ERP software.
Command	Insert button	Click this button to insert a command into the script.
Script	Insertion cursor	Point this red arrow to any spot in the script where you want to insert a new command.

The Command Pane

The command pane is the area in which you begin writing commands to create your AutoPilot script. You begin the command-writing process by clicking a command in the Command menu on the menu bar or by clicking buttons in the tool bar. After you have done so, distinct list areas appear in the command pane.

Note

Neither the Insert button nor lists appear until you click a command. You make choices from or entries to the command pane lists. When you click the Insert button, a command line appears in the script.

The command pane might also contain options. For example, when you click Application in the command menu, the command pane contains options for Use Default Form and Processing Options Only. When you click Select Grid Line in the command menu, the command pane contains options that allow you to script single-clicking or double-clicking a grid row in a form.

The two main components of the command pane are the command pane lists and the Insert button.

Command Pane Lists

Lists are distinct areas in the command pane, physically separated from one another and individually captioned. You make entries to or choices from these lists in order to write the commands that you insert in your script. A command pane list can be either populated or unpopulated. You make choices from populated lists and make inputs yourself to unpopulated lists. In either case, however, you use the lists to write script commands.

You work with both populated and unpopulated lists in the AutoPilot command pane. Populated lists in the command pane contain the specific items you choose to create a script command. For example, when you click Application in the command menu or click Launch Application in the tool bar, AutoPilot displays applications and descriptions of each application in an Applications list in the command pane.

AutoPilot populates a second list, the Menu list, when you click Launch Application. It displays menu text, or descriptions, of forms and interactive versions that are attached to the menu selections. Versions indicate that processing options exist for the application. The list also displays the Fast Path to the form.

As you write the script, the lists in the command pane change to reflect selections that you make in the menu bar or the tool bar. Other populated lists might include the following:

- Names of header controls, grid columns, forms, forms that appear next when you add a form or interconnect to another application, buttons, previously declared variables, previously declared validations, combo box items, or options (such as radio buttons and check boxes) found in forms
- Names of processing options for applications
- Sources of input to forms, such as literal values, UDC visual assists, valid values lists, variables, form interconnect visual assists, header controls, or grid columns
- Sources of row numbers in a form, such as literal values, valid values lists, or variables
- Values to be input to forms, which can be derived from an existing valid values list, variable, header control, or grid column
- Sources from which a repeat count value in the script can be defined

Unpopulated lists appear with a caption, but they are empty. You create or modify the script command by typing in the list words, numbers, special characters, spaces, or a combination thereof.

You can enter the following in unpopulated lists:

- Literal values to be input in header controls, grid columns, or a QBE line
- The name of a variable or validation that you are declaring
- The repeat count for a node in the script, which controls how many times the node, or tree control of commands, plays when you run a script
- The length of a wait period during script playback
- Comments to be inserted in the script
- A DOS command line message to the system
- A name for a screen capture
- A tree path that identifies a unique path to a node in a OneWorld form

Insert Button

When you make choices and entries to lists in the command pane and then click the Insert button, AutoPilot inserts a command line in the script pane. Each inserted command becomes a part of the script that AutoPilot runs in J.D. Edwards ERP software. The insertion cursor, which appears as an arrow in the script pane, follows the last command that you insert.

When you choose an application and version from the command pane and click the Insert button, you automatically launch J.D. Edwards ERP software if you have turned on the Playback button (identified by the initials PB) in the tool bar.

Clicking the Insert button for the first time starts the script. As you write the script by inserting new commands, AutoPilot continues to display all scripted commands, in the order of their insertion, in the script pane.

With the playback button turned on, as you make selections from the command pane in AutoPilot and click the Insert button, AutoPilot runs the scripted commands in J.D. Edwards ERP software.

The Script Pane

As you write your script, you can easily observe its progress because AutoPilot records each command that you insert in the script pane. The script pane contains visual expressions of each command that you insert.

The script pane consists of two components: command lines and the insertion cursor. The command lines express, in words and symbols, the selections that you make in the command pane. A command line does not appear in the script pane until you have clicked the Insert button. Command lines express either the context in which a command runs, or the action that is taken in the chosen context. In other words, context commands specify *where* you want all or a portion of the script to take place, while action commands specify *what* you want to take place in the script.

Each command line contains all or some of the following components:

- A symbol that designates the command as a script node.
- A symbol that identifies the specific type of context or action command that you wrote and inserted to the script, such as an Application command or a Press Toolbar Button command.
- A written description of the general context or action in J.D. Edwards ERP software, for example, Application.
- A written description of the specific context or action in J.D. Edwards ERP software, for example, {P0411 - A/P Standard Voucher Entry}.
- The source of the input into a form and its value. Value from "1," for example, means that you have inserted the literal value 1 to a header control, grid column, or QBE line.

Context commands and action commands make up the command lines in the script pane. A context command establishes the environment in which you write other commands. In other words, to click a button in a form, you first establish the context, which is a form in this example.

The following table summarizes the context commands that you write using AutoPilot and the results of writing those commands:

Context command	Result of writing command
Application	Launch J.D. Edwards ERP application
UBE	Launch UBE, application P98305 (Batch Versions) and form W98305D (Version Prompting)
Application Interconnect	With an application and form active, launch a different application or a form in the same application that is outside of the normal transaction sequence
Processing Options	Display processing options for a selected application in the AutoPilot command pane
Set Header Control Value	Specify the header control in which you want to input data
Set Grid Cell Value	Specify the grid cell in which you want to input data
Set QBE Cell Value	Specify the grid cell in the QBE line in which you want to input data
Form	Specify the form in which you want to take additional actions
UBE Selection	Launch Data Selection form
UBE Processing Options	Display processing options for a selection UBE in the AutoPilot command pane
UBE Print	Launch Printer Selection form

With a context established, you can write action commands. One function of action commands is to define the actions that you take within the context that you specify. If the context is a form, an action that you can take within that form is clicking a toolbar button. Therefore, `Press Toolbar Button` is an action command.

You can write other action commands independent of a specific context. For example, you can declare a variable (give it a name) and set and store a value for it before you launch an application. Likewise, you can declare a validation and associate it with a table and columns in the table independently of establishing a context. Still, you take these actions in order to accomplish something in a context. For example, you store the value of a variable in order to use it in a header control, grid column, combo box, or tree path.

The following table summarizes the action commands that you write using AutoPilot and the results of writing those commands:

Action command	Result of writing command
Select Grid Row	Select a grid row in the detail area of a form
Build Tree Path	Create a unique path to an item in a form that uses tree controls
Press Toolbar Button	Press standard buttons in a form, perform form and row exits, submit UBEs, select a grid tab, or press the grid scroll bar button

Action command	Result of writing command
Press Push Button	Press special buttons that do not reside on the toolbar of forms
Checkbox/Radio Button	Choose checkbox or radio button options in the header portion of a form
Select ComboBox Item	Choose item(s) in forms that use combo boxes instead of header controls
Press Clickable Text	Click a hyperlink on a form
Exit OneWorld	Exit J.D. Edwards ERP software
Command Line	Encapsulate a path to another program in the AutoPilot script
Comment/Wait	Write a comment about the script and insert it into the script pane; designate a command line and time period for AutoPilot to wait before proceeding with script playback
Variables	Declare a name for a variable, designate the source of its value, set the value, store the value
Declare New Validation	Declare a name for a database validation
Associate a Validation Column	Associate a table and a column with the declared validation; specify a value to be validated
Execute Validation	Write an SQL statement to validate whether an expected value is returned from the database
If <var> == <var>	Write a conditional (if/then) statement

See Also

See the following topics in the *AutoPilot Guide*:

- *Scripting the Context*
- *Scripting Actions*
- *Understanding the Script Pane Structure*
- *Modifying Scripts*

Bars in the AutoPilot Form

In addition to the command and script panes, the AutoPilot form includes four bars that assist you as you create a script. Bars in the AutoPilot form identify the script, contain options and buttons for scripting commands, and help identify the functions of buttons contained in the form.

Caption Bar

The caption bar is the horizontal bar that appears at the top of the AutoPilot form and identifies the name or title of the script that you are writing in AutoPilot. The text enclosed in the brackets in the caption bar is the name of the script.

Menu Bar

The menu bar appears horizontally beneath the caption bar. It is composed of options (beginning with File and ending with Help) that contain drop-down menus from which you can make selections that help you to write the script and to set up how AutoPilot runs.

File Option

Many of the choices in the drop-down menu of the File option represent essential Windows functions. You can create a new script, open an existing one, close a script, save it, or print it. However, the following choices reflect unique AutoPilot features: Send To, Properties, Repository, and Import.

Feature	Action
Send To	Enables you to send a script that you have written to another user who has access to AutoPilot.
Properties	Permits you to assign identifying features to the script, such as the function that the script tests.
Repository	Offers access to the script repository, which is a controlled storage location for completed scripts. This location is separate from your local drive and can be accessed by AutoPilot users to obtain examples of scripts that test particular functions.
Import	Allows you to import a previously saved XML script.

See Also

- *Property Pages for Scripts* in the *AutoPilot Guide*
- *Understanding the Script Repository* in the *AutoPilot Guide*

Edit Option

Currently, two functions are available for you to use when you need to modify existing scripts. These options allow you the flexibility to copy and paste command lines or branches of

scripts rather than deleting a command line or branch and then having to rewrite it. The following table lists the edit features and defines their actions.

Feature	Action
Copy	Copies a command line or branch of a script from one location to the clipboard of your system. Use the paste feature to insert the command line or branch of a script to another location. The command line or branch of a script that you copied remains in the original location.
Paste	Pastes a command line or branch of a script command into the insertion point. The paste feature does not replace the command line or branch of a script on which you are focused. Instead, it pastes the command line or branch of a script immediately below your insertion point.

View Option

You use the View option to set up the way in which the tool bar and status bar appear in the AutoPilot form.

Command Option

The drop-down menu that appears when you click Command in the menu bar contains the names of the commands that you can write to your script. These commands are the same as those that are represented by the tool bar buttons that you can click to write commands to the script.

The drop-down menu under the Command option also includes two choices that are not represented by tool bar buttons. `If <var> = = <var>` represents the command to write a conditional statement.

Play Option

The drop-down menu that appears when you click Play in the menu bar contains the names of the AutoPilot playback functions, each of which is represented by a tool bar button.

Clicking Playback on Creation toggles on or off the Playback button on the tool bar. When the playback button is turned on, AutoPilot plays the commands that you write in your script as soon as you insert them.

Tools Option

Using the Tools option allows you to fine-tune the way in which your script runs, to view the results of test scripts that you have run, and to generate data that you can use in your scripts. The following table summarizes the choices available from the Tools menu.

Tools Menu Choice	Purpose
Generate Valid Values List	Creates or selects data to store in a text file that you use in the script
Create a Script from Capture	Creates an AutoPilot script from the event stream that you capture
Include Local Script	Allows you to choose a script stored locally and then include that script within another script
Include Reposited Script	Allows you to choose a script stored in the repository and then include that script within another script
Results	Allows you to review the results of AutoPilot tests that you have run
Unwanted Windows	Allows you to close unwanted windows while executing a script
Select OneWorld Client	Allows you to choose from options to run a script to test Windows, Java, or HTML client
Options	Launches the Options form, which contains eight tabs that you use to set up the following options: • J.D. Edwards ERP and AutoPilot directories • Speed • Playback configuration • Sign-on parameters • Playback against OneWorld Java, playback against OneWorld HTML • Script generation • Specifications for script creation **Note Concerning HTML and MAF (Multiple Application Framework)** AutoPilot does not work with Interactive HTML if MAF (Multiple Application Framework) is turned on. Ensure that MAF is turned off.

Generate Valid Values List

You can create a text or numeric file that contains one or more values by clicking Generate Valid Values List under the Tools option. When you do so, a form appears that allows you to select data and to save it in a file. You can then use this file as a source of input for your script. When you click Generate Valid Values List, you use the Select Data File Type form to create a valid values list either by querying the database or by manually entering values of your own.

See Also

- *Valid Values List* in the *AutoPilot Guide*
- *Using a Valid Values List as a Source of Input* in the *AutoPilot Guide*

Create a Script from Capture

You can use the Create a Script from Capture option to create a script from the event stream. The Create a Script from Capture option can greatly increase your scripting productivity and shorten the learning curve for an inexperienced AutoPilot user. The option is designed to create a framework of an AutoPilot script. The option is not designed to create a script that you can run without some modification.

Note

When you use the Create a Script from Capture option to create a script, J.D. Edwards recommends that you use the fast paths instead of the menus when calling an application.

Include Local Script

You might want to write a script and include it with another script that tests a related function. To do so, you click Include Local Script from the Tools drop-down menu. This option enables you to choose a script that you have saved to your local directory and to include it with another script that you choose.

See Also

- *Including Scripts* in the *AutoPilot Guide*

Include Reposited Script

While a script is open, you can include a script that has been added to the script repository. To do so, you click Include Reposited Script from the Tools drop-down menu. This option enables you to browse through the scripts that have been checked in to the repository, and you can choose one or more of these scripts to include with a script that you choose.

See Also

- *Including Scripts* in the *AutoPilot Guide*
- *Understanding the Script Repository* in the *AutoPilot Guide*

Results

After you have played back a script, you have the option of saving the results of the test. AutoPilot collects those tests whose results you have elected to save and displays a summary of the test results in a Test Results form. To access the Test Results form, choose Results from the drop-down menu of the Tools option on the menu bar.

See Also

- *Understanding Script Reporting* in the *AutoPilot Guide*

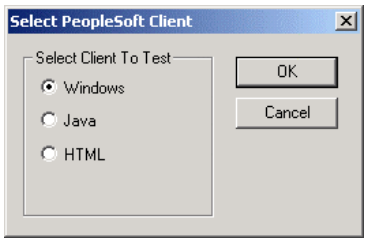
Unwanted Windows

You use the Unwanted Windows function to close windows that might appear as part of the system flow, but are not needed for your scripts' execution. Unwanted windows might include windows and message boxes such as Communications Failure, Confirm Delete, or Scheduled Packages.

You can enter, edit, and delete any windows or message boxes from a list that you create. You specify what should happen when the window appears. You can specify, for example, that AutoPilot should click OK on Confirm Delete and continue playing the script. The Unwanted Windows option works with any language, including Chinese, Japanese, and Korean.

Select OneWorld Client

You can run your scripts as a Windows, Java, or HTML client by choosing an option from the Select OneWorld Client form. If you choose the Java or HTML option, you must choose Options from the Tools menu and specify a server on the OneWorld Java tab or the OneWorld HTML tab.



Options for Configuring AutoPilot

Click Options to display a form with the following eight tabs, each of which contains controls and options to help you set up AutoPilot for testing:

- Directories
- Speed
- Playback
- Sign On
- OneWorld Java
- OneWorld HTML
- Script Generation
- Configure

The Directories tab allows you to specify where you start J.D. Edwards ERP software and where you store your local scripts, screen captures, and so on.

You can set the path for each directory by clicking the button next to each control. When you click the button, the Choose Directory form appears. You use this form to specify the path to each directory and the network drive on which that directory resides.

The Format combo box option allows you to choose a particular screen capture extension, such as .tif. If you do not want the option of adding scripts to the repository, you can disable it by choosing the Disable Repository option.

On the Speed tab, you can set how quickly AutoPilot types in a header control or grid cell in a form.

On the Sign On tab, AutoPilot displays your user ID, password, and the J.D. Edwards ERP software environment to which you sign on. If you are signed on to a different environment than the one that appears in the Environment control, AutoPilot displays a form that informs you that you must change the sign-on environment to match the J.D. Edwards ERP software environment.

On the OneWorld Java tab, you can specify a Java application server against which you can run an AutoPilot script.

On the OneWorld HTML tab, you can enter the universal resource locator for a OneWorld Web server, against which you can run an AutoPilot script.

On the Configure tab, you can do the following:

- Set how often you want your script to auto-save.
- Select J.D. Edwards ERP specifications, such as whether you want hidden edit and grid controls to appear, and whether you want to rebuild file specifications each time that you run an application or only when AutoPilot does not find the specifications.
- Set the threshold at which you want J.D. Edwards ERP software to idle.
- Click the Rebuild F9860.ATX button to load application names into AutoPilot.

The options on the Playback tab are divided into two sections. The top section of options allows you to configure script playback. The following table summarizes the purposes of the playback configure options.

Option	Purpose	Suggested Initial Setting
Play Back while Creating Script	AutoPilot plays back each command after you insert it in the script.	Off.
Save Results Data after Playback	AutoPilot writes data about script playback events to a table, where the results are stored.	On. You must choose this option if you choose any option other than None from the Events Stream Capture Level section.
Display Results Data after Playback	AutoPilot displays a Results form, which contains summarized information about each playback event.	On.
Ignore Breakpoints during Playback	During playback, AutoPilot ignores breakpoints that the user manually inserts into the script. If you do not choose this option, playback halts at a breakpoint until the user intervenes.	Off.
Accelerated Playback	AutoPilot communicates, through code, directly with the run-time engine to determine when a process is complete so that it can go on to the next command, thus speeding up playback.	Off. Choose this option only if you are certain that application launch is controlled by the run-time engine and not by a business function.

Cancel Playback on Comm Error	AutoPilot cancels playback if a communication error occurs between client and server. Choose this option when you are testing processes on a server.	Off
Log Variables on Script Failure	AutoPilot records the current value of variables when a script fails. This information can be useful when analyzing script failures. For example, the journal date variable value of "06/03/02" caused the script to fail because we are now working in 2003. J.D. Edwards highly recommends that you turn this option on.	On

The bottom section of options allows you to set up capture of script playback data. The chronological sequence of events that occurs during script playback is called an event stream. Using the options on the Playback tab, you specify how much of the event stream you want AutoPilot to capture.

You capture event stream data to accomplish two main purposes. You can import the data to the Analyzer Tool, which allows you to view data about each playback event in greater detail. For example, you can view the input values and return values of individual API calls, and you can see the time required to run each event. This information can aid in debugging applications.

You can also import the event stream to the Virtual Script Editor, where you can create a virtual script. You can run the virtual script on a single workstation to simulate many users, a process which helps you to test the scalability of your system.

The following table summarizes the purposes of the options on the Playback tab that allow you set up the capture of script playback data:

Option	Purpose
None	AutoPilot captures no data about script playback.
OneWorld warning and error messages	AutoPilot captures only data about warning and error messages.
Level 1 API calls	AutoPilot captures warning and error messages and captures API data about only those calls that initiate a business function or database call.
All API call levels	AutoPilot captures data about warning and error messages and about all API calls.

Window Option

The Window option provides several choices for changing the size and arrangement of AutoPilot forms. For example, you can choose to tile or cascade the forms if you have several open at once. In addition, the drop-down menu under Window displays the script or scripts that are currently open.

Help Option

Clicking About AutoPilot on the Help menu displays the version of AutoPilot that is installed on your machine, as well as the date on which you took the build.

Clicking Contents on the Help option displays the J.D. Edwards online help.

Tool Bar

The tool bar is composed of buttons that you click to perform the following actions:

- Script context and action commands
- Run script playback

You find the context and action commands represented by tool bar buttons in the Command option on the menu bar. You find the script playback commands represented by tool bar buttons in the Play option on the menu bar.

See Also

See the following topics in the *AutoPilot Guide*:

- *Scripting the Context*
- *Scripting Actions*
- *Playing Back the Script*

Status Bar

The status bar, located at the bottom of the AutoPilot form, provides information about your AutoPilot session. For example, after you have begun a session and are preparing to enter a new command, the status bar displays the message Ready, meaning that AutoPilot is ready to accept a new command. When you pass the cursor over a tool bar button, the status bar displays a written explanation of the function of that button.

Likewise, when you pass the cursor over any item that appears in a drop-down menu of the menu bar, the status bar displays a written explanation of the function of the item.

The status bar might also inform you that you need to wait before proceeding. For example, when you open a script for the first time in a session, the status bar asks you to wait while it loads the script specifications and reads the specifications for an application that it has not yet found.

Manipulating the AutoPilot Form

You can easily change the arrangement and size of the AutoPilot form and the panes within it. You can focus completely on one pane by manipulating the size of the form. On the other hand, if you are working with multiple scripts, you can keep each of them open and arrange them so that you can conveniently move between them as you work.

You might also want to resize the AutoPilot form and its panes when you are creating a script and playing it back. Adjusting the size of the form allows you to see both the AutoPilot form and the forms that are active in J.D. Edwards ERP as you work.

If you close AutoPilot and then open it again, the size of the AutoPilot form, the arrangement of the panes, and the position of the tool bar appear as they did when you closed the session. This feature might be useful when you set up the form as you like it and want to preserve the setup.

Changing the Size of the AutoPilot Form

You can easily change the size of the AutoPilot form by using the mouse. Moving the mouse within the form produces double-headed arrows. You then can size the form by holding down the mouse button and dragging the mouse in the direction that you desire.

► To expand the area of the AutoPilot form

1. On your desktop, with AutoPilot active, bring the cursor arrow to a corner of the screen.
2. When a diagonal doubled-headed arrow appears, click the mouse.
3. Holding down the mouse button, drag the arrow in the direction that is desired.

► To change the horizontal area of the AutoPilot form

1. On your desktop, with the AutoPilot form active, place the cursor arrow at one of the vertical edges of the AutoPilot form.
2. When a double-headed horizontal arrow appears, click the mouse.
3. Holding down the mouse button, drag the arrow to the left or to the right.

► To change the vertical area of the AutoPilot form

1. On your desktop, with the AutoPilot form active, place the cursor arrow over the caption bar or the bottom edge of the AutoPilot form.
2. When a double-headed vertical arrow appears, click the mouse.
3. Holding down the mouse button, drag the arrow up or down.

Arranging Multiple AutoPilot Forms

AutoPilot allows you to create and save multiple scripts during a single session or several sessions. You can open several scripts at once, resizing and rearranging them as you see fit.

You use the menu bar Window option to change the size and arrangement of AutoPilot forms when you want to work with more than one script. You can arrange multiple scripts so that you can view them simultaneously; likewise, you can easily move from one script to another.

If you decide to work with multiple AutoPilot scripts during a session, you can arrange the AutoPilot forms in either cascade or tile fashion, using the menu bar Window option. The Cascade command arranges the scripts in overlay fashion.

The top AutoPilot form is active. Click another form to make it active. To resize a form, place the cursor arrows on a vertical or horizontal edge, hold down the mouse button, and drag the form in the direction that you desire.

The Tile command divides the area of the AutoPilot form so that the existing AutoPilot forms appear simultaneously, adjacent to one another.

► To arrange multiple AutoPilot forms

1. Choose either Cascade or Tile from the Window menu.
2. Use the cursor arrow to change the size of the parent AutoPilot form or of any of the child forms.

Sizing Panes in the AutoPilot Form

You can change the size of panes in the AutoPilot form easily using the Split option, which you find under Window in the menu bar. The split option moves the cursor arrow to the splitter bar, which divides the command pane from the script pane. To resize the panes at any time, you can manually place the cursor arrow on the splitter bar.

► To size panes in the AutoPilot form

1. Choose Split from the Window menu.

   An arrow appears at the splitter bar, which divides the top pane from the bottom pane.

2. Drag the mouse up or down, expanding or shrinking the size of the panes, and click.

Manipulating the AutoPilot Tool Bar

You might work frequently with the tool bar during an AutoPilot session because you use many of its buttons to write context and action commands. To make your work easier, you can also move the tool bar and change its size and shape.

For instance, if you want more vertical space for the command pane, you can move the tool bar from near the top horizontal edge of the AutoPilot form to either the right or left vertical edge. You can also float the tool bar, moving it entirely out of the AutoPilot form and onto the desktop. Finally, after you have moved the tool bar from one position to another, you can return it to its original position by double-clicking the bar.

Relocating the Tool Bar

You can move the tool bar using the grabber, which is represented by a vertical bar. Note that two tool bars actually exist. One contains the buttons that represent action and context commands that you use to write your scripts. The other contains buttons that you use to play back your scripts. Each bar contains a grabber, so you can move one, the other, or both.

► To relocate the tool bar

1. In the AutoPilot form, place the cursor arrow on the grabber, which is represented by a vertical bar in the AutoPilot form.
2. Holding down the mouse button, drag the tool bar, which you can now place either vertically along the right or left edge or horizontally along the bottom of the AutoPilot form.

You can divide the two sections of the tool bar and place one along a vertical edge and one along a horizontal edge of the AutoPilot form, or you can place them together.

Resizing the Tool Bar

You might want to change the size and shape of the tool bar. You can do so easily by using the cursor arrows and the mouse button.

► To resize the tool bar

1. In the AutoPilot form, place the cursor arrow at the edge of the form.
2. When a double-headed horizontal arrow appears, click and hold down the mouse button.
3. Drag the mouse up, down, or across.

   As you resize the AutoPilot form, the tool bar resizes along with it.

Floating the Tool Bar

You might prefer to drag the tool bar completely outside of the AutoPilot form and work with it from your desktop. To do so, use the mouse to grab the bar and drag it to the position that you desire, or you can double-click the bar. When the tool bar is in a floating position, you can use the mouse to resize it.

► To float the tool bar by dragging

1. In the AutoPilot form, place the cursor arrow on the grabber in the tool bar, which is represented by a vertical bar.
2. Click and hold down the mouse button.
3. Drag the tool bar to any position desired.

   An outline of the tool bar appears as you drag it.

4. When the outline of the tool bar assumes the shape that you want, release the mouse button.

► To float the tool bar by double-clicking

1. Place the cursor arrow anywhere within the tool bar.
2. Double-click the mouse.
3. To return the tool bar to its original position, place the cursor in the bar that runs along its top and double-click the mouse again.

► To resize and reshape the tool bar from the floating position

1. In the AutoPilot form or on your desktop, with the tool bar in a floating position, place the cursor arrow on one of its corners or edges.
2. When the double-headed vertical, horizontal, or diagonal arrow appears, click the mouse.
3. While holding down the mouse button, drag the arrow away from the bar until a resized, reshaped, outline of the bar appears.

4. Release the mouse.

   The tool bar in its new configuration appears.

5. Double-click the top of the tool bar to return it to its original configuration.

Scripting the Context

To create a script, you choose options from lists in the command pane. These selections create the commands that you insert in the script, and you then play back these commands to test J.D. Edwards ERP applications.

You can insert two kinds of commands in an AutoPilot script: context and action. You use context commands to establish the setting that you want to test. These settings include applications, UBEs, interconnected applications, processing options, forms, headers, grid columns, and QBE lines. After you establish a context, you can write action commands, which accomplish specific tasks that you perform in J.D. Edwards ERP software, such as pressing a button or typing in a header control.

Context commands can depend on other context commands. For example, you write an application command to launch an application and form. You write a header command so that you can input data in one or more header controls in the form. Although applications, forms, and header controls are all contexts, you cannot type inputs to the header controls until you have established the application and form contexts in the script.

Understanding Context Commands

You write context commands during an AutoPilot session to establish the context in which you work. Each of these commands establishes a unique environment, and you write each command according to the AutoPilot functions that you want to test. Remember that, in general, you must write context commands before you can decide which actions you want to take.

The lists in the command pane change to reflect the context command that you choose. For example, the lists that appear in the command pane when you are writing an Application command are different from the lists that appear when you are writing a Form command. Therefore, you should become familiar with the concepts for each of the context commands.

Context commands also establish a hierarchy in the script pane. For example, you typically begin a script by writing an Application command. In writing this command, you also choose another context, a form, and both of the Form command lines appear in the script pane. However, AutoPilot indents the Form command line beneath the Application command line. This indentation indicates that the Application command gave rise to the Form command and, therefore, is its parent and is superior to it in the hierarchy of commands.

The most important point about this hierarchy for script-writing purposes is that it affects script playback. Changes that you make to a parent command affect the commands that are subordinate to it. For example, if you delete a parent command from the script, the system automatically deletes all of the commands that are children of that command.

The following is a simple hierarchy of AutoPilot commands:

- Primary context commands are Application, Application Interconnect, UBE, and Processing Options. These commands always provide the context for other context and action commands. They appear as parents to other commands in the script.
- Secondary context commands are Form, Header, Grid Column, and QBE. These commands generally are subordinate to primary context commands, but they provide the context for action commands. They appear as both parents and children of other commands in the script.

- Action commands, such as clicking a toolbar button, are nearly always subordinate to a primary or secondary context command. They nearly always appear as children of other commands in the script.

However, remember that variations exist for these generalizations. For example, a Form command, when it gives rise to a Header, Grid Column, or QBE command, is primary to these commands, but secondary to the Application or Application Interconnect command.

See Also

- *Understanding the Script Pane Structure* in the *AutoPilot Guide* for further discussion of the hierarchy of commands

Application Command

You use the Application command to launch interactive versions of applications. Clicking the Application command allows you to choose an application, the menu text for that application, and the Fast Path for the application.

The Application command is a primary context command. You must script it in order to script inputs to header controls, grid columns, or QBE lines in forms. An Application command also is often necessary if you want to interconnect to another application.

UBE Command

You use the UBE command to launch previously created UBE versions when you want to submit a UBE to J.D. Edwards ERP software for processing. AutoPilot allows you to launch UBE versions from a menu, from a row or report exit, from an application that calls for a blind UBE submission, or from another UBE. After you write a UBE command, you can write other commands. You can select data for your report, set UBE processing options, submit UBE versions to the printer, and instruct AutoPilot to wait for the UBE to complete processing before executing additional commands in the script. If necessary, you can also write a command that instructs AutoPilot to automatically exit the Batch Versions program (P98305) when you have completed scripting the UBE submission.

Options for the UBE Command

You can write the UBE command at various points in the script. The decision to do so depends on the process that you are testing. When you click UBE in the Command menu, the command pane lists that appear resemble the lists that appear when you click Application. You can choose from the lists a UBE, a menu Fast Path to the UBE, and a version.

The command pane also contains two options:

- Execute FASTPATH
- Create "Work With Batch Versions" commands

AutoPilot automatically turns on both of these options when you click UBE. The lists in the command pane change to reflect whether you have turned on one, both, or neither of these options. For example, if you choose a UBE, but turn off the Execute FASTPATH option, AutoPilot removes the Menu Item list.

The Execute FASTPATH Option

If you want to launch a UBE from a menu, you turn on the Execute FASTPATH option and then choose an option from each of the three lists in the command pane: UBE, Menu Item, and Version. When you click the Insert button, AutoPilot sends the Fast Path command to J.D. Edwards ERP software.

In some cases, you might not want to launch a UBE that uses a Fast Path. For example, double-clicking a grid row or clicking a button might launch a blind submission of a UBE. You might also access a UBE from the Batch Versions program (P98305). Finally, you might launch a UBE that is coded to automatically submit another UBE. In any of these cases, you turn off the Execute FASTPATH option after you choose a UBE, and AutoPilot removes the Menu Item list that contains the Fast Paths.

The Create "Work With Batch Versions" commands Option

When you complete your choices from the command pane and click the Insert button, AutoPilot automatically declares and sets a variable that stores the UBE version that you chose or that is automatically submitted. When you turn on the Create "Work With Batch Versions" commands option, AutoPilot performs the following tasks without your intervention:

- Interconnects to the Batch Versions program (P98305)
- Launches the form Work With Batch Versions – Available Versions
- Writes a QBE context command to the script
- Inputs the stored variable value to the QBE line
- Runs a Press Toolbar Button {Find} command
- Selects and double-clicks a row
- Confirms the Version Prompting form

If you turn off this option, AutoPilot writes no script lines to exit to the Work With Batch Versions – Available Versions form. Turn off this option when the UBE that you are launching is submitted automatically from a menu, an application, or another UBE. When you launch a UBE from a menu that is hard-coded to submit the version automatically, AutoPilot removes the Version list from the command pane and disables both of the options. When you click the Insert button, AutoPilot automatically submits the UBE.

Option Combinations

Depending on the operation that you are testing, you can launch UBEs from different locations. The location dictates the combination of options that you choose.

The following table shows five different scenarios for launching a UBE and the combination of options that you choose.

UBE Launch	**Execute FASTPATH Option**	**Create "Work With Batch Versions" commands Option**
From a menu	On	On
From a Reports menu in an interactive application	Off	On

From a Row menu in an interactive application	Off	Off
From a menu that is hard-coded to submit the UBE as a blind execution	Disabled	Disabled
From another UBE	Disabled	Off

UBE Submission

When you write a script that includes the Batch Versions program (P98305), you must write the command to submit the UBE. You do so by writing a Press Toolbar Button {Submit} command. This command presses the Submit button in the Version Prompting form. If you want to select data for your report, choose the data selection option, and then submit the report. If the UBE is a blind execution, you do not work with the Version Prompting form. J.D. Edwards ERP software automatically submits the UBE, and you can write the command to print it.

UBE Data Selection

If you launch your UBE with the Create "Work With Batch Versions" commands option, you can also use the Criteria Design Aid feature in AutoPilot to select the data for your report. The UBE context command and the UBE data selection action command work together when you script in AutoPilot. After you have launched a UBE and written, either automatically or manually, a series of commands that runs through the Version Prompting form, you can use the Checkbox/Radio Button command in AutoPilot to choose the Data Selection option, and then submit the form by writing a Press Toolbar Button command. AutoPilot then allows you to script the data selection criteria by making entries to and choices from the command pane.

You script data selection by clicking UBE Selection in the command menu or by clicking the CDA button in the tool bar. When you do so, the command pane appears with five lists:

- Line Number
- Operator
- Left Operand
- Comparison
- Right Operand

In addition, the command pane contains an option that allows you to press the OK button on the Data Selection form.

Note that the command pane lists mirror the functions of the Data Selection form. After you enter a line number for data, you determine the logic for the data selection that you want to enter to formulate your criteria. Note that the Operator and Comparison lists contain a SKIP option. If, for example, you have completed your entries to the Data Selection form for one line and you want to make entries on another line, choose the SKIP option to cause the Operator and Comparison entries for the new line to duplicate the entries for the previous line.

While the data selection feature in AutoPilot is essentially the same as the Criteria Design Aid feature, the AutoPilot feature has some limitations. For example, you can choose a left

operand in the Data Selection form by clicking a selection in a drop-down menu. No such drop-down menu currently exists in AutoPilot. The name of the object that populates the left operand in OneWorld differs from the name that appears in the drop-down list. You must manually enter the name of the object, exactly as it appears in the J.D. Edwards ERP list.

Likewise, you must manually enter information in the Right Operand list as they appear in the drop-down menu, or you can enter one or more literal values. You can enter multiple and range values to the right operand. You separate multiple values with commas, such as 1,2,5; you separate a range value with two hyphens, such as 1--4.

In addition, while you can declare a variable, set its value, and then use that value in the right operand, you must enter the name manually and enclose it in angle brackets, such as <batchno>. In contrast, when you use a variable as a source of input to a header control or grid column, AutoPilot presents in the value selection list the names of all variables that you have declared and allows you to choose one.

After you enter the name of a declared variable, AutoPilot displays options in the command pane that prompt you to designate the value of the variable as a literal, a range, or a list.

Creating and using variables can make the process of selecting data for your UBE more efficient. For example, you might want to write a script that enters transactions, then launches a UBE and extracts particular data for the report. Creating a variable allows you to store the data, such as a list of particular cost centers, that you need for your report. When you are ready to select the data, you enter the name of the variable in the right operand of the Data Selection form. You can store in the variable a single value, a series of discrete values, or a range of values.

When you complete one set of criteria, you click the Insert button. With playback turned on, you can observe the way in which AutoPilot enters your criteria in the Data Selection form. If you click UBE Selection again, you can enter selection criteria on another line. When you have completed your data selection, choose the Press OK option in the AutoPilot command pane and click the Insert button. If processing options exist for the UBE version that you launched, they appear next, and you script processing options commands, as necessary.

See Also

See the following topics in the *AutoPilot Guide*:

- *Source of Input List*
- *Using a Variable as a Source of Input*
- *Using the Value Selection List*
- *Creating Variable Links*
- *Selecting Data for a UBE*

UBE Processing Options

After you submit a UBE, some versions ask you to set processing options. You set these options in much the same way that you set processing options for interactive applications. However, the command menu entries and tool bar buttons that you choose to set UBE processing options are separate and distinct from those that you choose to set processing options for interactive applications.

When you set processing options for a UBE version, you click UBE Processing Options in the command menu. You then choose options from the Processing Options list in the command

pane, and then write a Press Toolbar Button {OK} command. AutoPilot inserts the chosen processing options in the script and runs them during playback.

See Also

- *Processing Options Command* in the *AutoPilot Guide*
- *Setting the Context as a Processing Option* in the *AutoPilot Guide*

UBE Print Command

You can send your UBE to print after you have submitted it or after J.D. Edwards ERP has automatically submitted it. You do so by clicking UBE Print in the command menu or the stoplight button on the tool bar.

AutoPilot offers three options in the command pane after you click UBE Print: Wait for UBE to complete before continuing, Expect no "Printer Selection" window, and Create exit "Work With Batch Versions" commands. At this point, AutoPilot cannot send UBEs to the screen in Adobe Acrobat format. When you submit a version, AutoPilot automatically chooses the To Printer option on the Version Prompting form.

Wait for UBE to Complete Option

If you turn on this option, AutoPilot submits the UBE to the default printer and waits for it to finish before resuming the script. If the UBE that you submit launches additional UBEs, AutoPilot chooses the printer queue. When the printer completes all of the submitted UBEs, AutoPilot resumes playing the script.

If you turn off this option, AutoPilot submits any UBEs for printing, but resumes the script without a waiting period. You can submit your UBE from either a local or a server environment, but you cannot override the location after you choose it. In either environment, AutoPilot handles, without your intervention, all print windows that appear.

Expect No "Printer Selection" Window Option

You use this option if the UBE that you are running does not require a printer. This option prevents AutoPilot from waiting for a printer window to appear before AutoPilot continues running the script. If you turn off this option and a printer window does not appear, AutoPilot continues to wait, and the script fails to advance. Turning this option on tells AutoPilot not to expect and wait for a print window and to continue running the script after you submit the UBE.

Create Exit "Work With Batch Versions" Command Option

If you launch a UBE from the Batch Versions program (P89305), you can choose the Create exit "Work With Batch Versions" command option. When you do so, AutoPilot automatically writes a Form command line for Work With Batch Versions - Available Versions and writes a Press Toolbar Button {Close} command. These commands confirm and close the form and display the menu item. If you do not launch a UBE from the Batch Versions program, do not turn on this option.

Application Interconnect Command

The Application Interconnect command allows you to script the exit from one application to another, which might occur, for example, when you press the Add button.

Clicking Application Interconnect in the command menu lets you choose and insert in the script pane in AutoPilot the new application and form command lines that mirror the application and form that are active in J.D. Edwards ERP software.

You script the Application Interconnect command *reactively*; that is, you script it after you have already exited to a new application. You must script an Application Interconnect command so that the AutoPilot script Application and Form commands match the application and form that are active. If you do not script the Application Interconnent command, you cannot continue scripting because the Form command line in the script pane will not match the form and application that are active.

Remember that AutoPilot also automatically writes an Application Interconnect command to the script when you launch a UBE from a menu or from a Reports menu in an interactive application. In each of these cases, you choose the Create "Work With Batch Versions" commands option in the command pane when you choose a UBE. AutoPilot launches the UBE and then automatically writes a series of commands to the script, including an Application Interconnect to the Batch Versions application (P98305).

You can also script an Application Interconnect command by clicking Press Toolbar Button in the command menu and choosing the Press Custom Button option. However, you cannot use the two scripting approaches interchangeably.

In deciding to choose Press Custom Button to script an Application Interconnect command, you *initiate* in AutoPilot the exit to a new application or form. AutoPilot inserts the Application and Form commands in the script and launches the application and form.

Suppose that you launch the Companies application (P0010) and the Work With Companies form (W0010C), and then you want to exit to a new application, so you click Press Toolbar Button in the command menu. When you click Press Custom Button, a tree node expands.

You choose Form or Row, and then, by clicking one or the other, choose from various form or row menu selections, which you use to script an Application Interconnect command. These menu selections match the lists that appear when you click Form or Row in the menu of the active form.

When you choose a form or row selection in AutoPilot, new lists appear in the command pane. You choose an application and form, and then press the Insert button. AutoPilot runs the form or row selection and interconnects to the application that you chose.

You might close the interconnected application and return to the previous form. In that case, you must write another Application Interconnect command and Form command in AutoPilot to ensure that the command lines in the script pane match the application and form that are active.

When you move to a new form within the same application and click Form in the command menu, the Form list displays only the forms that are included in that application. However, if you move to a form that is in a different application or is outside of the normal cycle of transactions for the application, the name of that form does not appear in the list when you click Form in the command menu. When you click Application Interconnect in the command menu, you can choose from the command pane lists the new application and form that are active.

See Also

See the following topics in the *AutoPilot Guide*:

- *Form Command*
- *Setting the Context as a Form*
- *The Custom Button Option*
- *Clicking a Custom Button*

Processing Options Command

You can use AutoPilot to set processing options for interactive versions of applications that you want to run. During playback, AutoPilot determines whether the processing options are set as you scripted them. You script the processing options for an application and the interactive version that is attached to the menu item for the chosen application. To do so, you click Application in the command menu, choose an application and menu item, and then click the Processing options only option in the command pane.

Note Concerning Identical Processing Option Labels

AutoPilot does not distinguish between multiple processing option labels on the same template if they are spelled identically. Even if the identically spelled processing options occur on different tabs, AutoPilot cannot distinguish between them during script execution. AutoPilot does not report any errors, and it operates on the last instance of any identical processing options.

When you click the Insert button, the command pane displays the tabs with processing options for the application version that you have chosen. With playback turned on, you can view the J.D. Edwards ERP Processing Options form and its tabs.

AutoPilot serializes the processing option IDs when you create the script. When you load the script for playback, AutoPilot finds the matching processing option IDs in J.D. Edwards ERP software and displays processing option text that is consistent with the release for which you play your script.

If a new release changes the processing option ID and the text, AutoPilot displays an error message in the processing options command line of the script pane when you play back the script. You can correct the processing option text in the command pane.

Form Command

When you script an Application or an Application Interconnect command in AutoPilot, you select from both the Application list and the Menu list in the command pane. Your selection from the Menu list specifies the form and version that appears in J.D. Edwards ERP software when you click the Insert button. The Form command line appears automatically in the script pane anytime that you choose an application and form from these lists and click the Insert button.

Scripting commands in AutoPilot requires that the Form command line in the script pane mirror the form that is active. You can verify that the two mirror one another by choosing a form from the Next Form list in the command pane, by clicking Form in the command menu, or by clicking the Form button in the tool bar.

Next Form List

The Next Form list helps you ensure that the Form command line in the script pane matches the active form. For example, you might decide to script pressing the Add button in a form, such as Work With Addresses, to move to another form, such as Address Book Revision. To do so, click Press Toolbar Button in the command menu, choose Standard Button in the command pane, and choose Add from the tree. In the Next Form list, you can choose Address Book Revisions. When you click Insert, AutoPilot inserts the Form command {Address Book Revision}.

Form List

You might not know the form that appears next in the software. Suppose you do not know that the form Address Book Revision appears when you click Add on the Work With Addresses form. In this case, you can choose Unknown/None from the Next Form list.

However, when you choose Unknown/None, the Form command line in the script pane still shows {Work With Addresses}, while the active form is Address Book Revision. If you attempt to continue scripting at this point, the script fails.

To ensure that the Form command line mirrors the form that is active, you click Form in the command menu or click the Form button on the tool bar. In the command pane, a Form list displays the names of the forms that are included in the application that you selected. In this case, you choose Address Book Revision from the Form list.

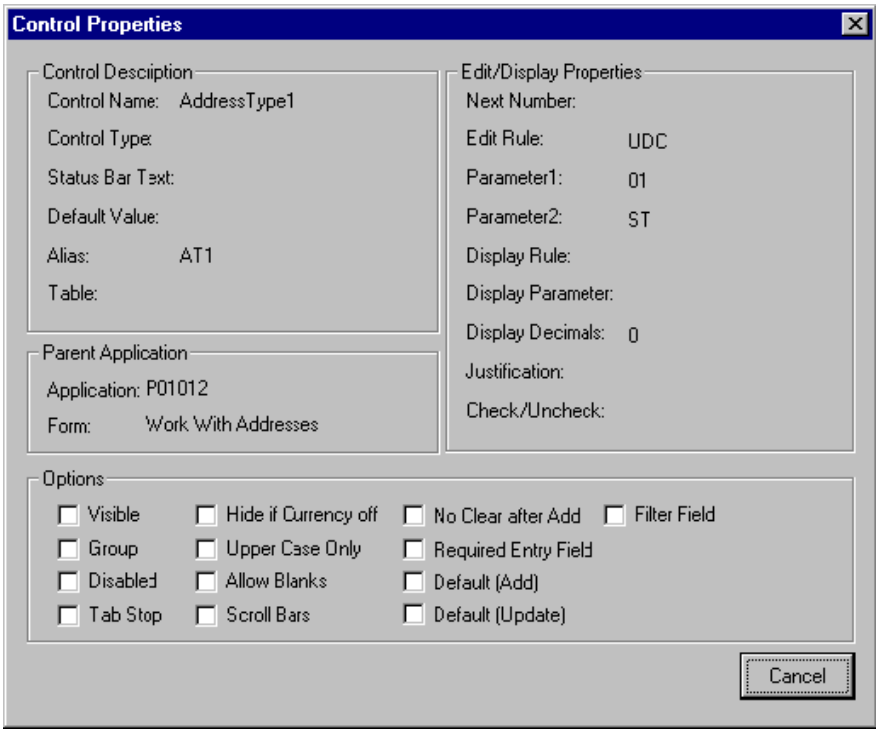
When you insert the command, the Form command line matches the form that is active, and you can proceed with scripting. You have confirmed that the active form matches the form name that appears in the script pane of the AutoPilot form.

Header Command

The Header command establishes the header portion of a form as the context in which additional commands, such as clicking buttons, entering control inputs, and choosing options, can take place. You begin scripting the Header command by clicking Set Header Control Value in the Command menu.

The header control list that appears in the command pane includes all of the controls that are in the active form. You can choose to display hidden controls by clicking Tools in the menu bar, then clicking Options, then clicking the Configure tab, and then choosing the Display Hidden Edit Controls option in the Spec Selection Options.

To review the properties of any header control, right-click the name of the control in the Header Control list, and then click Control Properties. The system displays Control Properties. This form includes four sections: Control Description, Parent Application, Edit/Display Properties, and Options. To exit the form, click Cancel.



Note Concerning the Header Command

After you choose a header control, you can choose additional options in the command pane, including a source of input for the control and the value of the input. When you click the Insert button, AutoPilot inserts two command lines in the script. The context command line is Header. However, by choosing a control, a source of input, and the value for the input, you have written an additional command. This command is the action command Type to, which appears in the script pane as a command line that shows the name of the control, as well as the source of input and the value.

Grid Column Command

The Grid Column command establishes the grid column in a form as the context in which additional commands, such as pressing grid buttons and entering inputs to grid columns, can take place. You begin scripting the Grid Column command by clicking Set Grid Cell Value in the Command menu.

The grid column list that appears in the command pane includes all of the columns that are in the active form. You can display hidden columns by clicking Tools in the menu bar, then clicking Options, then clicking the Configure tab, and then choosing the Display Hidden Grid Columns option in the Spec Selection Options.

To review the properties of any grid column, right-click the name of the control in the Grid Column list, and then click Control Properties. The system displays the Control Properties form. This form includes four sections: Grid, Column, Column Edit/Display Properties, and Column Properties.

Note Concerning the Grid Column Command

After you choose a grid column, you make additional command pane choices, including a source of input for the control and the value of the input. When you click the Insert button, AutoPilot inserts two command lines in the script. The context command line appears containing the words Detail Information. By choosing a grid column, a source of input, and a value of the input, you have written an additional command. This command is the action command Type to, which appears in the script pane as a command line that shows the name of the grid column, as well as the source of input and the value.

QBE Command

The QBE command establishes the QBE line in a form containing a grid as the context in which additional commands, such as entering inputs in the QBE line and pressing the Find button, can take place. You begin scripting the QBE command by clicking Set QBE Cell Value in the Command menu. You then choose a grid column in which you want to type inputs.

Note Concerning the QBE Command

After you choose a grid column, you make additional command pane choices, including a source of input for the control, and the value of the input. When you click the Insert button, AutoPilot inserts two command lines in the script. The context command line appears containing the words QBE Information. By choosing a grid column, a source of input, and a value of the input, you have written an additional command. This command is the action command Type to, which appears in the script pane as a command line that shows the name of the grid column, as well as the source of input and the value.

See Also

- *The Type to Command* in the *AutoPilot Guide*
- *Scripting the Type to Command* in the *AutoPilot Guide*

Creating a Script from Event Rule Capture

You can use AutoPilot to translate events in the software into an AutoPilot script. Using the Create a Script from Capture option in AutoPilot allows you to automatically create a script from actions that you perform in the software without having to understand the internal data relationship or the intricacies of creating an AutoPilot script manually. Generating a script from the event stream can also increase the productivity of scripting and shorten the learning curve for an inexperienced user.

When you create a script from event rule capture, you create a script framework, not a finished script. AutoPilot captures all the events in the stream. You might not need all those events. You will have to decide how to modify your scripts generated from the tool.

Although the script might not be fine tuned to suit your needs, these unmodified scripts have many uses. For example, you can capture a process that is causing an error in an AutoPilot script and then e-mail the script to your support organization.

Note

When you use the Create a Script from Capture option to create a script, J.D. Edwards recommends that you use the fast paths instead of the menus when calling an application.

► To create a script from event rule capture

1. From your desktop or the appropriate directory, launch J.D. Edwards software and sign on.
2. In AutoPilot, from the File menu, choose New.
3. From the Tools menu, choose Create a Script from Capture.
4. On Create a Script with Event Capture, complete the following field:
   - Script Name
5. Click Start Capture.
6. On J.D. Edwards Solution Explorer, complete the following field and press Enter:
   - Fast Path

     Enter a command in the Fast Path, such as 3/G11. You cannot enter an abbreviation of a program, such as OMW, UDC, OL and so on. If the fast path command does not contain a menu selection, the AutoPilot script will fail.

     You can capture multiple applications in a sequence as long as you always start an application by entering a command in the fast path.
7. Perform your task.

   AutoPilot records every event capture. Ensure that you are performing actions deliberately in order to create the most accurate script. For example, if you click the OK button twice, AutoPilot records two events.
8. Click Stop Capture when you have finished your task, and then click Generate Script.

   Your new script loads in the AutoPilot script view pane.
9. From the File menu, choose Save to save your script.
10. Modify the script as needed.

Writing the Script Using Context Commands

You can begin scripting context commands in one of three ways: by clicking Command in the menu bar, by clicking a hot key on the keyboard, or by clicking a tool bar button. When you do so, lists appear in the command pane. You make selections from populated lists and enter information in unpopulated lists. When you click the Insert button, AutoPilot inserts one or more command lines into the script pane. The context command is identified in the script pane with words and symbols.

In general, the sequence that you follow to write primary context commands is as follows:

- Choose a general context, such as an interactive application or UBE, by clicking the Command menu, a hot key, or a tool bar button.
- Specify a context, such as a particular application and menu item, by making choices from or entries in lists.
- Click the Insert button to write the command to the script pane.

Some context commands depend on other context commands. For example, Header is a context command, but you set the header as the context only after you have set an application and a form as the context for the script.

The general sequence that you follow to write secondary commands is as follows:

- Choose a general context, such as a header, grid, or QBE line.
- Specify a context, such as a control or grid column. Available controls are determined by the application and form that you previously chose.
- Choose a source of input for the specific context.
- Choose a value to be input in the specific context.

Setting the Context as an Application

You often begin a script by launching an application. This process establishes both the application and the form that you work with in your script.

Note Concerning the Menu List and Setting Context

The Menu list includes the text of the menu item in Explorer, the Fast Path, and the application version. You can launch different versions of the same application from different Explorer menus. Be sure to choose the menu item that is associated with the version and processing options that you want to test.

► To set the context as an Application command

1. Choose Application from the Command menu.
2. From the Application list in the command pane, click a J.D. Edwards 5 application.
3. From the Menu list in the command pane, click the name of a menu item.
4. Click the Insert button.

AutoPilot inserts the Application, Fast Path, and Form command lines into the script pane. In the playback mode, AutoPilot launches the specified version of an interactive application. The form appears on the screen with the AutoPilot form, and you can navigate between the two.

Setting the Context as a UBE

There are several ways that you can set the context as a UBE. You can begin the script by launching the UBE from the Batch Versions program (P98305), or you can launch an interactive application and then perform a report exit to the Batch Versions program. You can launch an interactive application, then perform a row exit that launches a blind execution. You can launch a UBE from a menu that is hard-coded to submit the version automatically. Finally, you can launch a UBE that launches another UBE. In this case, J.D. Edwards ERP software launches any subsequent UBEs and then blindly submits them without any further intervention by AutoPilot.

When you click UBE in the Command menu, options for executing a Fast Path and for creating a Work With Batch Versions command appear. You can use these options to establish the way that AutoPilot submits the UBE, except when a menu is hard-coded to automatically submit it.

If you choose a UBE that is not automatically submitted, you must write a command to click the Submit button on the Version Prompting form. Before you write that command, however, you can click the option that allows you to select data for your report. In some cases, after you submit the UBE and select data, you can set processing options. Doing so requires you to write a UBE Processing Options command to the script and set the options by making choices from the lists in the command pane.

Finally, you can choose the way to print the UBEs that you submit. You can instruct AutoPilot to wait for the UBE to print before resuming running the script, or you can send the UBE to print, but tell AutoPilot to continue running the script. If it is appropriate to the function you are testing, you can also write a command to close the Batch Versions program and return to Explorer.

Launching a UBE

You can use AutoPilot to launch a UBE from a variety of contexts. You might begin your script by launching a UBE from a menu. On the other hand, you might launch a UBE after you launch an interactive application. In this case, you might launch the UBE from a report menu, or you might launch it after you perform a row exit. You might also choose to launch a UBE that is automatically submitted. Finally, you can launch a UBE that in turn launches one or more additional UBEs.

Launching a UBE from a Menu

If you want to launch a UBE from a menu, you must make choices from each of the three lists that appear in the command pane when you click UBE in the Command menu: Application, Menu Item, and Version. You also turn on both of the options in the command pane: Execute FASTPATH and Create "Work With Batch Versions" commands. The first option establishes the Fast Path AutoPilot uses to access the UBE; the second option commands AutoPilot to automatically perform a QBE search in the Work With Batch Versions - Available Versions form for the UBE version that you chose. When you turn this option off, you write the command to submit the UBE from the Version Prompting form.

► To launch a UBE from a menu

1. From the Command menu, choose UBE.
2. In the command pane, make a selection from each of the available lists:
   - UBE
   - Menu Item
   - Version
3. Turn on both the Execute FASTPATH and Create "Work With Batch Versions" commands options, if available.
4. Click the Insert button.

   When you click the Insert button, AutoPilot automatically inserts a series of command lines in the script. The command sequence ends at the Form {Version Prompting} command line.

Launching a UBE from a Report Menu

You might want to launch a UBE from the Reports menu in an interactive application. In this case, you begin the script by launching an interactive application. You use the Press Custom Button option to choose a report. You choose the UBE without the Execute FASTPATH option, and the Menu Item list containing Fast Paths to the UBEs disappears. You choose the Create "Work With Batch Versions" commands option, which means that you write the command to submit the UBE from the Version Prompting form.

► To launch a UBE from a report menu

1. From the Command menu, choose Application.
2. In the command pane, choose options from the following lists:
   - Application
   - Menu
3. Click the Insert button.
4. From the Command menu, choose Press Toolbar Button.

   Note
   Ensure that your INSERT line in the script is always at the child level.

5. In the Button list, expand Custom Button, and then expand Report.
6. Choose a report.
7. Click the Insert button.
8. From the Command menu, choose UBE.
9. In the command pane, choose a UBE.

   Do not turn on the Execute FASTPATH option. If it is turned on, turn it off.

   Note
   When you turn off the Execute FASTPATH option, the Menu Item list disappears. Do not turn off this option until you have chosen the UBE. If you turn off the option before

you choose the UBE, AutoPilot turns the option on again after you have chosen the UBE, and you must turn it off again.

10. Choose a version from the Version list.
11. Turn on the Create "Work With Batch Versions" commands option.
12. Click the Insert button.

    Because you turned on the Create "Work With Batch Versions" commands option, AutoPilot automatically writes a series of script commands that ends at the Form {Version Prompting} command line.

Launching a UBE from a Row Menu

You can launch a UBE from a Row menu in an interactive application. To do so, you begin by launching an interactive application. After you have written a row exit command using the Press Custom Button option, you choose the UBE command. You click neither of the command pane options. The Menu Item list disappears, and you choose a version. If you turn off the Batch Versions option, AutoPilot blindly submits the UBE.

► To launch a UBE from a Row menu

1. From the Command menu, choose Application.
2. In the command pane, choose options from the following lists:
   - Application
   - Menu
3. Click the Insert button.
4. From the Command menu, choose Set QBE Cell Value.
5. In the command pane, choose options from the following lists:
   - Grid Column
   - Source of Input
   - Value selection
6. Click the Insert button.
7. From the Command menu, choose Press Toolbar Button.
8. In the Button list in the command pane, under the Standard Button heading, choose Find.

   Caution
   Do not choose options from the Next Form list. If you choose these options and insert the command, AutoPilot launches the new interactive application that you chose.

9. Click the Insert button.
10. From the Command menu, click Select Grid Row.
11. In the command pane, click the following options:

- Click by row number
- Single click

12. In the Source of Row Number list, choose a value source.
13. In the value selection list, enter a row number or choose a variable or valid values list.
14. Click the Insert button.
15. From the Command menu, click Press Toolbar Button.
16. Click Custom Button.
17. Click Row.
18. Choose a Row selection.

Caution
Ensure that you do not make choices from the Application or Next Form lists because they launch applications. You need to launch a UBE.

19. Click the Insert button.
20. From the Command menu, click UBE.
21. In the command pane, choose a UBE from the UBE list.

Caution
Turn off both the Execute FASTPATH and Create "Work With Batch Versions" commands options.

22. Choose a version from the Version list.
23. Click the Insert button.

    AutoPilot automatically submits the UBE.

Launching a UBE That Is Automatically Submitted

If you launch a UBE that is hard-coded to submit the version automatically, you cannot click the options in the command pane. When you choose the UBE, AutoPilot disables both of the options and the Version list disappears. You choose from the Menu Item list and click the Insert button, and then AutoPilot launches the UBE and blindly submits it.

► To launch a UBE that is automatically blindly submitted

1. From the Command menu, click UBE.
2. In the command pane, choose a UBE that will be blindly submitted from the UBE list.

   When you choose the UBE in this scenario, AutoPilot disables both options and the Versions list disappears.

3. Choose a menu item from the Menu Item list.
4. Click the Insert button.

Launching a UBE from Another UBE

You might launch a UBE from a menu or from an application that in turn launches one or more subsequent UBEs. In this case, J.D. Edwards ERP software automatically launches any UBEs that are called by the first one and blindly submits them. You do not choose versions or printing options, or set processing options. Without further direction from the AutoPilot script, J.D. Edwards ERP software completes all of the processes that are associated with the UBEs that the first UBE launches.

► To launch a UBE from another UBE

1. In the Command menu, follow the steps for creating a script that launches a UBE from a menu, a Report menu from an interactive application, or a Row menu from an interactive application.
2. In the command pane, choose a UBE from the UBE list.

   If the UBE is coded to launch another UBE, AutoPilot disables the Execute FASTPATH option and removes the Menu Item list from the command pane.
3. Choose a version from the Version list.
4. Turn off the Create "Work With Batch Versions" command option.
5. Click the Insert button.

Submitting a UBE

You write a command to submit the UBE only when you have turned on the Create "Work With Batch Versions" commands option in the command pane. When you turn on this option, AutoPilot automatically writes commands that culminate with the Form {Version Prompting} command line. You then use the Press Standard Button option to write a command to press the Submit button on this form.

► To submit a UBE

1. Write commands through the command line Form {Version Prompting}.
2. From the Command menu, choose Press Toolbar Button.
3. In the Button list in the command pane, click Standard Button.
4. Choose Submit.
5. Click the Insert button.

Selecting Data for a UBE

After you launch a UBE, you might want to refine the data that appears in your report. If so, you can use the Criteria Design Aid feature in AutoPilot, which you access either by clicking UBE Selection in the command menu or by clicking the CDA button on the tool bar. This feature allows you to script entries to the Data Selection form.

You can use the Criteria Design Aid feature when you launch a UBE with the Create "Work With Batch Versions" commands option turned on. If you launch the UBE from a menu, AutoPilot automatically inserts a series of commands that ends at the Form {Version Prompting} command line. If you launch the UBE from a Report menu, you write a series of commands that culminates at the same point. In either case, however, when your script

reaches the Form {Version Prompting} command line, you can write a command to click the Data Selection option and a command to submit the report for data selection.

At this point, you can click the UBE Selection command and use the AutoPilot command pane to script entries to the Data Selection form. When you are finished, you can click the OK option in the command pane. If you stored values in a variable earlier in your script, you can use these values in the right operand of the Data Selection form. You must, however, type the name of the variable in the Right Operand list of the command pane. In addition, the variable name must be enclosed in angle brackets (<>).

After you enter a variable for the right operand, AutoPilot displays options that you use to designate the value of the variable as a single value, a range of values, or a list of values.

You use Criteria Design Aid in conjunction with writing a UBE command, not as a stand-alone command. In addition, some UBEs allow you to set processing options. You use AutoPilot to set the processing options for the UBE after you have selected the data that you want to appear in your report. You complete the UBE submission process by sending the report to the printer.

Caution Concerning Object Names

Enter the object name in the left operand list exactly as it appears in the drop-down menu of the list on the Data Selection form. Likewise, enter an object name in the right operand list exactly as it appears in the drop-down menu of the list in the Data Selection form, unless you enter a literal value. If you enter a literal value, you can enter a single value, multiple values, or a range of values. You separate multiple values with commas; you separate a range of values with a hyphen.

► To select data for a UBE

1. From the Command menu, choose UBE.
2. In the command pane, choose options from the following lists:
   - UBE
   - Menu Item
   - Version
3. Turn on both the Execute FASTPATH and Create "Work With Batch Versions" commands options.
4. Click the Insert button.
5. From the Command menu, choose Checkbox/Radio Button.
6. In the command pane, click DataSelectionYN in the Radio Button or Check Box list.
7. In the Source of Input list, click Check.
8. Click the Insert button.
9. From the Command menu, choose Press Toolbar Button.
10. From the Button list, choose Standard Button, and then Submit.
11. Click the Insert button.
12. From the Command menu, choose UBE Selection.

13. In the command pane, complete the following fields:
    - Line Number
    - Operator
    - Left Operand
    - Comparison
    - Right Operand
14. If you enter a variable in the Right Operand list, click one of the following options that appear in the command pane in order to specify the type of value:
    - Single value
    - Range of values
    - List of values
15. Click the Insert button.
16. After you write as many UBE Selection commands as you need, click the Press OK option and then click the Insert button.

Setting UBE Processing Options

After you submit a UBE version, you might see the Processing Options form appear. In this case, you must set processing options for the UBE before you can run the report. To set the processing options for the UBE, you choose UBE Processing Options in the command menu, and then choose options from the command pane.

► To set UBE processing options

1. Submit a UBE.
2. From the Command menu, choose UBE Processing Options.
3. In the Processing Options list of the command pane, click the node of a processing options tab.
4. Choose a processing option.
5. Choose a source of value from the Source list (if applicable).
6. If the value is literal, enter it in the unpopulated Literal Value field. If you choose Variable as the value source, AutoPilot populates the Variables list, which contains the UBE version with which you are working, as well as the names of any variables for which you have set values.
7. Click the Insert button.
8. In the command pane, click the Press Toolbar Buttons node in the Processing Options list.
9. If you are satisfied with the processing options that you set up, click OK. If you are not, click Cancel.
10. Click the Insert button.

See Also

- *Using a Variable as a Source of Input* in the *AutoPilot Guide*
- *Setting the Context as a Processing Option* in the *AutoPilot Guide*

Printing a UBE

Clicking UBE Print in the Command menu produces three options in the command pane. If you turn on the option Wait for UBE to complete before continuing, AutoPilot submits the UBE to the printer and waits for it to complete before it resumes the script playback. If you do not turn on this option, AutoPilot continues playing back the script without waiting for the UBE to run.

If the UBE you run does not print, click the Expect No "Printer Selection" Window option. This option ensures that AutoPilot does not wait for a printer window to appear before it resumes the script.

Turning on the option Create exit "Work With Batch Versions" commands, allows you to automatically write a Form command line for Work With Batch Versions - Available Versions and a Press Toolbar Button {Close} command to return to the form that was active before launching the UBE. You turn on this option only if you launched your UBE with the Create exit "Work With Batch Versions" commands option turned on.

► **To print a UBE**

1. Submit the UBE and set any necessary processing options, or after AutoPilot blindly submits the UBE, from the Command menu, choose UBE Print.
2. In the command pane, click one or more of the following options:
   - Wait for UBE to complete before continuing
   - Expect no "Printer Selection" window
   - Create exit "Work With Batch Versions" commands

   Note
   Click the Create exit "Work With Batch Versions" commands option only if you launched your UBE from the Work With Batch Versions – Available Versions form.

3. Click the Insert button.

Setting the Context as an Interconnected Application

In scripting a command to press a standard button, such as Add, you might exit from one J.D. Edwards ERP application to another. When this occurs, you must script an application interconnection by clicking Application Interconnect in the Command menu.

For example, you might want to write a script using the Customer Ledger Inquiry application (P03B2002). If you launch the application, choose the menu item Work With Customer Ledger Inquiry, then script pressing the Add button and Unknown/None from the Next Form list, J.D. Edwards ERP software exits to a new application, Standard Invoice Entry (P03B11).

If you click Form in the Command menu, the Standard Invoice Entry form does not appear in the Form list in the command pane. This tells you that by pressing the Add button, you exited

to another application. You cannot continue scripting until the Application and Form command lines in AutoPilot mirror the application and form that are active.

Using the Application Interconnect command, you can ensure that your script includes the new application and form in the script pane so that you can continue scripting. Remember that you use the Application Interconnect command *after* you exit to a new application.

► To set the context as an interconnected application

1. From the menu bar in the form to which you have exited, click Help.
2. Choose About J.D. Edwards.
3. Note the application ID and form name and click OK.
4. In AutoPilot, in the Command menu, Click Form.

   Note that Standard Invoice Entry does not appear in the Form list.
5. In the Command menu, click Application Interconnect.
6. In the command pane, choose from the lists that appear:
   - Application (choose the application that is active)
   - Menu (choose the form that is active)
7. Click the Insert button.

   AutoPilot interconnects to the new program or form, and the Application and Form command lines in the script pane now mirror the program and form that are active. You can now script additional commands.

See Also

- *The Custom Button Option* in the *AutoPilot Guide*
- *Clicking a Custom Button* in the *AutoPilot Guide*

Setting the Context as a Processing Option

You might want to set processing options for a particular application before you begin writing secondary commands for the application. To do so, you choose an application and menu item from the command pane as if you are launching an application. However, before clicking the Insert button, you click the Processing options only option in the command pane. This option allows you to choose processing options from lists in the command pane.

► To set the context as a processing option

1. From the Command menu, choose Application.
2. From the Application list in the command pane, click an application.
3. From the Menu list in the command pane, click the name of a menu item.
4. In the command pane, turn on the Processing options only option.
5. Click the Insert button.

   The command pane now displays a list of the processing options tabs for the application version you have chosen. In the script pane, the command line shows the Launch Processing Options symbol, the template for the application, and the version of the application that you chose.

6. Expand the node of one of the tabs.
7. Choose a processing option.

   AutoPilot populates the Source list in the command pane with two sources of input: literal and variable.

8. Choose a source of value.

   When you do so, a value selection list appears in the command pane.

9. If the value is literal, enter it in the unpopulated Literal Value field. If the value is a variable, AutoPilot populates the Variables list with the names of any variables whose value(s) you have set.
10. Click the Insert button.

    AutoPilot enters to the script pane a command line that summarizes the processing option(s) you have chosen.

Note
With playback turned on, when you insert a processing option value in AutoPilot, AutoPilot inserts the value to the corresponding control in the Processing Options form.

---

11. In the command pane, click the Press Buttons node in the Processing Options list.
12. If you are satisfied with the processing options you have set up, click OK. If you are not, click Cancel.
13. Click the Insert button.

Note
You can insert as many processing options to the script as you wish. You can then launch the application, if you desire. When you do so, be sure not to turn on the Processing options only option in the command pane.

See Also

- *Using a Variable as a Source of Input* in the *AutoPilot Guide*

Defining Unwanted Windows

When you create scripts, windows and message boxes such as *Communication Fail*, *Confirm Delete*, or *Scheduled Packages* might appear. These windows and message boxes are not routinely recognized by AutoPilot scripts and their unexpected appearance might cause your scripts to fail. To prevent script failure, you can specify what you want AutoPilot to do when certain windows and message boxes appear.

You can enter, edit, and delete any windows or message boxes from a list that you create. You can also designate a subscript for Autopilot to play when a particular window or message box appears. The Unwanted Windows option works with any language, including Chinese, Japanese and Korean.

► To define unwanted windows

1. From the Tools menu, choose Unwanted Windows.
2. On Close Unwanted Windows, click New to define a new unwanted window.

   Alternatively, click a row in the Window To Search For And Close pane, and then click Edit.

> **Note**
> To delete an entry, click a row in the Window To Search For And Close pane, and then click Delete. The system *does not* prompt you with a confirm delete message. Once you delete the selection, you cannot recover it. You will have to enter a new unwanted window entry if you need to replace the deleted one.

3. On New Unwanted Window Entry, complete the following field:
   - If This Window Is Found

     Enter the exact name of the form. Ensure that you enter the exact form name the way it appears in language (for example, English, French, German, and so on).

4. Choose one of the following options:
   - Then Click Button

     Choose this option if you know that the form contains a button to click. Ensure that you enter a command in the combo box (for example, OK, Cancel, and so on).

   - Or Send Numeric Command

     Choose this option to bypass the actual button. In Windows, when you click buttons on forms, numeric messages are sent to the window message handler. Typically the OK button sends "1" and Cancel sends "2." This method is more flexible, but not as intuitive (You can actually see what these messages are by using DevStudio's SPY++ utility.).

   - Or Play Script

     Choose this option to play your designated subscript if AutoPilot finds the form that you specified in the If This Window Is Found field.

   - Fail Script

     Choose this option to make the script fail whenever the unexpected window appears. The Fail Script option overrides both the Then Click Button and the Or Send Numeric Command options.

5. Click OK.
6. On Close Unwanted Windows, click Apply, and then click OK.

Setting the Context as a Form

When you insert an Application command in the script, AutoPilot inserts in the script pane the name of the form that you selected from the menu list. When you move to a new form, you must script the Form command to establish the context in AutoPilot. You can script the Form command either from the Next Form list or from the Command menu.

► To script the Form command using the Next Form list

1. Insert an application into a script.
2. From the Command menu, choose a command that allows you to switch forms, for example, Press Toolbar Button.
3. In the Button list, click a button-pressing option, such as Add, that takes you to another form.
4. In the Next Form list, click the name of the form that appears next.

   The Next Form list contains the names of the forms that are included in the current application.

5. Click the Insert button.

► To sccript the Form command using the command menu

1. Insert an application into a script.
2. From the Command menu, choose Form.

   The Form list appears in the command pane. It displays the names of all forms that are included in the current application.

   Note
   You can also display the Form list in the command pane by clicking the Form button in the tool bar.

3. Choose a new form to be confirmed that matches the active form in the software.
4. Click the Insert button.

   In the script pane, the new Form command line contains the name of the active form in the software.

Setting the Context as a Grid Column

After you launch an application and choose a form, you can establish a grid column in the form as a context for further scripting. When you click Set Grid Cell Value in the Command menu, a Grid Column list appears in the command pane if the active form has a grid detail area. From this list, you can choose a specific column to further refine the context.

► To set the context as a grid column

1. From the Command menu, choose Set Grid Cell Value.
2. In the Grid Column list, choose a grid column in which you want to enter data.

3. Choose a source of input from the Source of Input list.
4. In the value selection list, choose or enter a value and click the Insert button.

   When you click the Insert button, AutoPilot writes both a Grid (or Detail Information) context command line and a Type to action command line in the script pane.

Setting the Context as a Header

After you launch an application and choose a form, you can establish the header portion of the form as the context for further scripting. When you click Set Header Control Value in the Command menu, a Header Control list appears in the command pane from which you can choose a specific control to further refine the context.

► To set the context as a header

1. In the Command menu of the AutoPilot form, click Set Header Control Value.
2. In the Header list, choose a control to which you want to input data.
3. Choose a source of input from the Source of Input list.
4. In the value selection list, choose or enter a value and then click the Insert button.

   When you click the Insert button, AutoPilot writes both a Header context command line and a Type to action command line in the script pane.

Setting the Context as a QBE Line

The QBE line provides another context in which you can script commands after you launch an application and choose a form.

When you click Set QBE Cell Value on the command menu, a Grid Column list appears in the command pane, if the active form has a grid detail area. You can choose a specific column to further refine the context.

► To set the context as a QBE line

1. Insert an application into a script.
2. From the Command menu, choose Set QBE Cell Value.
3. In the Grid Column list, choose a grid column in which you want to enter data.
4. Choose a source of input from the Source of Input list.
5. In the value selection list, choose or enter a value and click the Insert button.

   When you click the Insert button, AutoPilot writes both a Grid (or Detail Information) context command line and a Type to action command line in the script pane.

See Also

- *The Type to Command* in the *AutoPilot Guide*
- *Scripting the Type to Command* in the *AutoPilot Guide*

Scripting Actions

Action commands designate the specific actions that the script performs, such as pressing buttons, choosing options, and entering data, within a context, such as an application or form. Action commands require a context, yet they are essential in that they specify the unique steps that you want to take within the chosen context. For example, you must write action commands in order to move between forms; enter data in header controls, grid columns, or QBE lines; select lines in a grid; perform database queries and updates; and so on.

Action commands also allow you to move within a script to a non-J.D. Edwards application, such as Microsoft Excel. You do this by sending a message to your system via a command line in AutoPilot. You can also use the Command Line command to capture screens and store the images in a file for later use.

You also use action commands when you want to enhance scripts that you have written. For example, you can write an action command to include a previously-created script within another script. For example, in a script that requires the entry of dates and then goes on to test other functions, you might include a stand-alone script that tests the entry of dates.

When you want to play back a script that you have created, you can use action commands to customize the playback. For example, you can insert a Wait command in the script. This command tells AutoPilot to wait the amount of time that you specify, at the point in the script that you specify, before it proceeds with playback. In addition, you can insert comments in the script that identify what it is testing or that describe what occurs at a particular point during playback.

After you script the entry of data in forms, you might want to verify that AutoPilot entered the information in the database that you specified. AutoPilot offers another action command, database validation, which enables you to do that.

Action commands help you do the following:

- Build scripts that test a specific set of processes.
- Test whether the data that you entered during the course of scripting has been properly entered in the database.
- Modify and comment on scripts that you have already created.
- Customize the way in which your script runs.
- Use applications other than J.D. Edwards to run your script or assist you in other tasks, such as writing documentation.

The Type to Command

You use the Type to command to script inputs for header controls, grid columns, or QBE lines in a form. Unlike some action commands, such as Press Toolbar Button, no Command menu item or tool bar button represents the Type to command. To write it, you use lists to specify the context as a header, grid, or QBE line; choose a specific header control or grid column; designate a source of input for the control or column; and choose a value to input into the control or column. The value can be either a literal or you can derive it from a variable, valid values list, UDC visual assist, or a form interconnect visual assist.

The lists that you use to write the Type to command appear in the command pane of the AutoPilot form.

The Header Control or Grid Column List

The Header Control and Grid Column lists are populated with alphabetic descriptions of the data dictionary items that are located in the header, grid, and QBE areas of the form, which is the context you have set. You click a control or column in which you want to script an input.

Source of Input List

After you choose a header control or grid column, use the Source of Input list to choose one of the following sources from which to get a value to input in the header control or grid column:

- Literal value
- Valid values list
- Variable
- UDC visual assist value
- Form interconnect visual assist
- Clear source of input

Literal Value

When you choose Literal Value as a source of input, you specify that an entry in a control, grid column, or QBE line of a form appears exactly as it appears in the value selection list. For example, the literal value of a NameAlpha control entry might be *Jane Meade*, which is exactly the text that AutoPilot enters in the header control of a form when the script runs.

Valid Values List

When you choose a literal value as a source of input, AutoPilot assigns only one value to a header control, grid column, or QBE line in a form. Choosing Valid Values List as a source of input enables you to create a text or numeric file that can contain multiple values, any of which you can enter in the header control, grid column, or QBE line. You can create a valid values list either by assigning your own values or by choosing a database and querying it for values to include in the list.

You might choose Valid Values List as a source of input when you want to run a script multiple times and, each time, enter a different value in a specified header control, grid column, or QBE line. As AutoPilot loops through the script that you created, the value it enters in the control or column changes to reflect the values that you included in the list. Alternatively, you might want to run a script once, but enter five different values in a grid column. Again, creating a single valid values list that contains five items enables you to do this.

If you exit from a script, exit AutoPilot, and then open the script again, AutoPilot resets the valid values list so that, when you play back the script, the first value that you entered in the list appears first. If you exit a script without exiting AutoPilot, and then open the script again, AutoPilot uses the value that was next in order on the list when you exited from the script.

Variable

You might want to choose a value, store it, and then use it at a subsequent place in the script. You might also want to use the value more than once in the script. In this case, use a variable as a source of input and store its value anywhere in the script. You declare the variable to assign a name to it. Then you set and store its value, which you can get from a valid values list, header control, grid column, another variable, or from your literal input.

The command pane in the AutoPilot form displays the following six components when you write a variable command:

- New variable list. You enter the name of the variable in this list, thereby declaring the variable.
- External variable option. By choosing this option, you specify that the variable can be linked to a variable in another script so that its value can be passed between scripts.
- Default variable list. You can enter a value that AutoPilot uses even if you do not set a value for the variable.
- Existing variable list. AutoPilot displays the names of any existing variables that you have declared in the script.
- Source of value list. You choose a source of value for the variable, such as a literal value, a valid values list, a header control, a grid cell, or another variable. In addition, you can choose variable manipulations, such as adding or subtracting a literal value or a variable value from the variable.
- Value selection list. You enter a literal value or choose the object that contains the value that you store in the variable. For example, if you chose a header control as the source of value, you choose from the value selection list the specific control that contains the value. You can also obtain variable values from valid values lists, from variables to which you have previously assigned a value, or from J.D. Edwards ERP sources, such as error and warning messages or grid row counts.

The following table explains other key terms that are related to variables in AutoPilot scripts:

Variable Term	Meaning/Application
Variable scope	The range of commands within a script in which the value for a variable can be used.
Global variable	A variable for which the value can be used throughout an entire script.
Local variable	A variable for which the value can be used only within a portion of a script.
External variable	A variable that can be linked to a variable in another script so that a variable value can be passed between scripts.
Default value	A value that you assign to a variable that AutoPilot uses when you do not set the value of the variable elsewhere in the script.
Conditional statement	An if/then statement that you write by comparing the values of two variables. The statement stipulates that if a condition exists in the script, then the script should run other commands.
Variable concatenation	The practice of stringing together two or more variables to create a new variable.
System variable	A variable for which the value is derived from J.D. Edwards data, such as error and warning messages.
Valid values count	A variable for which the value is derived from the number of items in a valid values list.
Variable watch list	A list that tracks variable values that are used during script playback.
Validation success	A variable for which the value indicates the success or failure of a database validation.

Variable Scope

Variable scope refers to how broadly you can use the value of the variable within your script. You create a node each time you write a context command. The node in which you declare a variable determines its scope. For example, if you declare a variable within an Application command node, the scope of the variable extends to that node only, and you can use a value that you set for the variable only within that node. If, for example, you declare a variable within an Application command node, and then you launch another application, you cannot use the value that you set for this variable within the new Application command node. If you declare a variable within a Form command node, its scope extends only to that form.

See Also

See the following topics in the *AutoPilot Guide*:

- *Changing the Scope of a Variable*
- *Indented Nodes*
- *Drag and Drop*

Global Variables

The scope of a variable is global when you can use its value throughout the entire script. To establish global scope for a variable, you must make the Declare variable command a child of the Begin Script node, which is always the first node in the script.

Local Variables

You can use the value of a local variable only within a portion of the script, specifically the node to which you attached it. For example, you might declare the variable immediately after you launch an application and a form. In this instance, you can set the value of the variable and use this value only for any command lines that you script within the Form command node because the Declare variable command that you write is a child of the Form command node.

You can expand the scope of the variable by dragging it to another node that is higher in the script. For example, you might drag the Declare Variable command from the Form command node to the Application command node, which makes it a child of the Application command. This action broadens the scope of the variable, and you can use the value that you set for it anywhere within the application.

However, the scope of the variable is still local. If you launch another application later in the script, you cannot use the value of the variable within that new Application command unless you make this command a child of the first Application command.

External Variables

A variable with global scope allows you to pass a value to header controls, grid columns, and QBE lines throughout a single, stand-alone script. AutoPilot also allows you to declare a variable as external, which means that you want to link the variable to a variable in another script. You use external variables when you want to pass variable values between scripts. For example, you might store a batch number in one script. If you declare the variable that stores the batch number as external, you can link that variable to external variables in one or more other scripts, and pass the batch number value to other scripts.

See Also

See the following topics in the *AutoPilot Guide*:

- *Variable Linking between Scripts*
- *Creating Variable Links*
- *Script Includes*
- *Including Scripts*

Default Values for Variables

You might want to create a script that you can play both in stand-alone mode and with other scripts. To do so, you assign a default value to your variable. For example, you might create a Script B that links to Script A, which passes along a batch number value. Suppose, however, that you want to play Script B by itself. If you set a default value in Script B, AutoPilot uses that value when you play Script B in stand-alone mode.

You can also assign a default value to a variable that you do not declare as external. AutoPilot uses the default value of the variable each time that you use it as a source of input for a header control, grid column, or QBE line. If you write a command to set a value for the variable, that value overrides the default value.

Conditional Statements

Conditional statements allow you to write If/Then/Else commands that compare the values of two variables for which you have declared names and set values. If the script meets the conditions that appear in the statement, then AutoPilot runs an additional branch of the script. Conversely, you can write commands that are connected to an Else branch in the script or you can allow the script to end if it does not meet the condition that appears in the statement.

You might write a conditional statement to ensure that your script tests an application even if the script does not meet the conditions that you expected to exist. For example, you might want to test making revisions to an existing Address Book number. If the Address Book number exists, then AutoPilot selects the grid line in the Work With Addresses form, double clicks the line, and then revises the existing Address Book number in the Address Book Revision form. However, if the Address Book number does not exist, your script fails unless you write a conditional statement that stipulates that if the Address Book number does not exist, AutoPilot should add a form and run commands to create a new Address Book entry.

Another conditional statement might test the converse: if the Address Book number does not equal the number that AutoPilot returns to the QBE line of the Work With Addresses form, the Address Book number does not exist. If that condition is met, AutoPilot clicks Add and creates a new entry. If the Address Book number does exist, AutoPilot double-clicks the grid line and revises the Address Book entry. This is the Else portion of the statement.

You can also compare variables between scripts by declaring a variable as external in one script, including the script with a parent script, and linking the external variable to a variable in the master script. You then build your conditional statement on these two variables.

You can use AutoPilot to set conditional statements of data equality or inequality, but the tool does not allow you to develop compound conditional statements that link together.

Variable Addition

Variable addition enables you to scroll through a grid from top to bottom. For example, you can write a command that adds one to the row number of the grid each time AutoPilot plays back the node of the script. If you set the repeat count of the node to match the number of lines in the grid, AutoPilot scrolls through the entire grid, one line at a time, from top to bottom.

Variable Subtraction

Variable subtraction enables you to scroll through the entire grid, one line at a time, from bottom to top. You write a command that subtracts one from the row number of the grid each time AutoPilot plays back the node of the script.

Variable Concatenation

Variable concatenation enables you to string together existing variable values to create a new variable. For example, you might have created two variables with values of 10 and 25, respectively. You can concatenate them to create a new variable that has a value that you use to select a range of values for a UBE.

System Variables

System variables obtain their values from J.D. Edwards ERP software, rather than from the information that you enter in AutoPilot. To use a system variable, you do not need to declare a variable or set its value because its value is determined during script playback.

The following table names the system variables and presents examples of how they might be used in script writing:

Name of system variable	Meaning	Possible use in script
Errors	AutoPilot records the number of J.D. Edwards ERP error messages that occur during script playback.	Use in conjunction with a conditional statement. For example, set a condition that specifies that if the number of error messages returned is greater than 0, then AutoPilot should run the Exit OneWorld command.
Warnings	AutoPilot records the number of J.D. Edwards ERP warning messages that occur during script playback.	Use in conjunction with a conditional statement. For example, set a condition that specifies that if J.D. Edwards ERP sends a warning message when the OK button is clicked, then AutoPilot should click the OK button twice.
Grid Row Count	AutoPilot records the number of completed rows in a grid.	Set a repeat count for a node to ensure that AutoPilot accesses each line in the grid during script playback.

Note Concerning the Value of the Grid Row Count

The value of the Grid Row Count system variable is determined by the number of rows that J.D. Edwards ERP completes after you click Find, not by the total number of rows in a completed grid. If you want the Grid Row Count value to reflect the actual number of rows in the grid, use the Select Grid Row command to go to the bottom of the grid. After you script this command, the Grid Row Count value is the total number of completed rows in the grid.

See Also

- *Performing Grid Row Operations* in the *AutoPilot Guide*

Valid Values Count

If you choose valid values count as a source of input, you also choose from the value selection list a valid value that you or someone else created. The list can be stored either on your local drive or on a server. AutoPilot counts the number of items in the list and stores that number as a variable. You might use the valid values count as a source of input and establish the repeat count for a node by setting the value of the grid row count from the valid values count. The number of times AutoPilot plays back the script node matches the number of items in the valid values list, and you can write a command for AutoPilot to enter the values from the valid values list to a cell in each line of the grid.

See Also

- *Valid Values List* in the *AutoPilot Guide*

Variable Watch List

The watch list, which you can select from the AutoPilot View menu, is a separate form that displays the values of variables in the script during playback. Because the watch list is not in the AutoPilot form, you can display it at all times, even as you exit and open scripts.

The watch list contains two column headers: Variable and Value. During script playback, each time AutoPilot sets the value of a variable, it adds the variable name and its value to the list. If the variable value that AutoPilot enters in a header control or grid column is invalid and script playback stops, AutoPilot stops adding values to the watch list. Each time that you stop the script and replay it, AutoPilot clears the watch list.

Validation Success

The Validation Success variable enables you to quickly verify whether the data that you expected to exist as a result of running your script actually does exist. After you declare a validation, you choose Validation Success as a source of input, and choose from the value selection list the name of the validation that you want to verify.

After you associate and run the validation and run the script, AutoPilot displays in the watch list the name and value of the validation success variable that you created. A value of 1 indicates that the validation was successful; a value of 0 indicates that the validation failed. In the rare instance that you declare and associate a validation, but do not run the validation, AutoPilot returns a value of 2 when playback completes.

See Also

- *The Database Validation Command* in the *AutoPilot Guide*

UDC Visual Assist Value

You can also use as a source of input a value from a UDC visual assist. Choosing UDC Visual Assist Value from the Source of Input list populates the UDC Visual Assist Value list with the same UDC values that appear when you click the flashlight button on the form. When you insert the value in your script in the playback mode, AutoPilot tests the code that opens the form and chooses the value that you specified for your script.

UDC Visual Assist Value appears in the Source of Input list only if a header control or grid column in the form contains a visual assist.

Form Interconnect Visual Assist

You might need to use a visual assist that is not associated with a UDC. For example, you might want to run a company master search. In J.D. Edwards ERP software, when you click the flashlight icon for company master search, a new application appears.

You script this form interconnect in your AutoPilot script by clicking Form Interconnect Visual Assist in the Source of Input list. When you choose Form Interconnect Visual Assist and insert the command in the script, AutoPilot tests the code that triggers a series of events in J.D. Edwards ERP software. AutoPilot clicks the flashlight button, runs an Application Interconnect command, and confirms the new form. AutoPilot inserts both the Application Interconnect and Form commands in the script pane, and you can script any additional commands that you need.

The Form Interconnect Visual Assist choice appears in the Source of Input list only when a header control or grid column contains a visual assist that requires an exit to a new application.

Note Concerning Valid Value Selection

You do not use the value selection list when you choose Form Interconnect Visual Assist. This source of input requires that you choose a value after you exit to the new application.

Clear Source of Input

The Clear command in the Source of Input pane allows you to remove an entry to a header control, grid column, or processing option form control. Each time you choose a control or grid column from the command pane and choose Literal Value from the Source of Input list, AutoPilot provides a reminder in the value selection list that you can clear the content of the control or column.

The Value Selection List

After choosing a Source of Input, you must specify the value or values to enter in the header control, grid column, or QBE line. You can use any of the following methods to supply the value:

- Enter a literal value of numbers, letters, spaces, or a combination of these.
- Choose a valid values list that you created.
- Choose a variable that you declared and set.
- Choose a system variable.
- Choose a UDC or form interconnect visual assist value.

Remember that the caption of the value selection list changes to reflect the choice that you make in the Source of Input list.

Scripting the Type to Command

You work with the Header Control, Grid Column, Source of Input, and value selection lists to create a Type to command. You create this action command when you create the context commands Header, Grid (or Detail Information), and QBE. When you click the Insert button, AutoPilot writes the context command and the Type to command and indents the Type to command beneath the context command to reflect the script pane command hierarchy.

Using the Header Control or Grid Column List

You begin writing the Type to command when you click Set Header Control Value, Set Grid Cell Value, or Set QBE Cell Value in the command menu. When you click one of these commands, the Header Control or Grid Column list appears, populated by the controls or columns from the active form. You click the control or column in which you want to type data.

► To use a header control or grid column list

1. From the Command menu, choose Set Header Control Value, Set Grid Cell Value, or Set QBE Cell Value.
2. In the Header Control or Grid Column list of the command pane, choose a header control or grid column in which you want to enter data.
3. Click the Insert button.

Using the Source of Input List

After you choose a header control or grid column, you must choose a source of input for it. Depending on the process that you are testing and how you want your script to run, you choose from any of four possible sources of input: Literal Value, Valid Values List, Variable, or UDC Visual Assist Value.

The documentation for the Source of Input list includes the step that follows choosing a source of input, which is entering or choosing a value from the value selection list. The value selection list is also documented separately.

Using a Literal Value as a Source of Input

When you run a script, a literal value appears in the form exactly as you type it in the unpopulated field Literal Value.

► To use a literal value as a source of input

1. From the command pane in the AutoPilot form, choose Literal from the Source of Input list.
2. In the Literal Value field, type an input as you would normally enter it in the header control, grid column, or QBE line of the form. The entry can be letters, numerals, special characters, spaces, or a combination thereof.
3. Click the Insert button.

Using a Valid Values List as a Source of Input

After you create a valid values list, you can use it as a source of input for a header control, grid column, or QBE line. You use a valid values list when you want to enter values multiple times, or when you want to run a script multiple times and input a different value each time.

For example, if a valid values list contains five items, you can enter 5 for the repeat count for the node that contains the list. During one script playback, AutoPilot loops through the node five times and inserts a different item from the list each time. Conversely, you can leave 1 in the repeat count for the node that contains the list, but change the repeat count at Begin Script to 5. During each of the five playbacks of the entire script, AutoPilot inputs a different value from the list.

If you close the script, leave AutoPilot open, and then open and run the script again, AutoPilot inputs the next value in the list in the appropriate context. After you create the valid values list, it contains stored values that you can use as a source of values in subsequent scripts.

You must create a valid values list before you choose it as a source of input for a form. To create a valid values list, choose the type of valid values list that you want to create, enter values in the list, and name the list.

Creating a List of Literal Values

A list of literal values is a list that contains values that you choose. Before you create the list, you should verify that the values that you create are valid for the application that you want to test.

► To create a list of literal values

1. From the menu bar of AutoPilot, click Tools.
2. Choose Generate Valid Values List.
3. On Select Data File Type, choose List of Literal Values and click Next.
4. On Enter File Name & Date, type a file name in the File Name field.

   If you type the name of an existing file, the values in it automatically fill the list.
5. Type one or more values in the Enter Values list, separating them by pressing Enter. The values should be stacked vertically in the box.
6. Click Finish.

Creating a Valid Values List from a Simple Database Query

A simple database query produces a valid values list that contains values that AutoPilot retrieves from the database and includes in the list, based on the table and column that you choose. You can limit the number of records in the list, and you can specify the method that AutoPilot uses to sort the records, such as in ascending, descending, or random order.

► To create a valid values list from a simple database query

1. From the Tools menu, choose Generate Valid Values List.
2. On Select Data File Type, choose Simple Database Query and click Next.
3. On Select Table, double-click a table.
4. In the Column Name window, choose a table column and click Next.

5. Specify format options and sort options by clicking the appropriate options.

Note
To view the contents of the valid values list, click the Preview button. Advanced users can also enter SQL statements.

6. Click Preview.
7. Click Next.
8. In the Finish window, assign a file name to the valid values list by typing in the control.
9. Click Finish.

Using a Valid Values List in the Script

After you create a valid values list, you return to the Command menu and click the context in which you want to write commands. You choose a header control or grid column, and then choose Valid Values List from the Source of Input list. When you choose Valid Values List, the name of the list that you created appears in the value selection list.

▶ To use a valid values list as a source of input

1. From the header control or grid column list in the command pane of the AutoPilot form, choose a header control or grid column.
2. Choose Valid Values List from the Source of Input list.
3. Choose the name of a valid values list that you have created.
4. Click the Insert button.

Updating the Repeat Count in a Node

The valid values list can contain multiple values. If you want to use all the values as inputs for a header control, grid column, or QBE line, you change the repeat count in the Form command node of the script to match the number of items in the list. This ensures that, during playback, AutoPilot successively types each value in the control, grid column, or QBE line until it exhausts the list.

▶ To update the repeat count in a node

1. In the script pane of the AutoPilot form, click the Form line for the node in which you scripted the valid values list.
2. In the command pane, type in the Repeat Count list the number of times that you want the node to play back.
3. Click the Update button.

Using a Variable as a Source of Input

To use a variable as a source of input, you must first declare it, or give it a name. You can declare the variable at any point in the script. However, if you make the variable global, the value that you assign to it can be used at any subsequent point in the script. After you have declared the variable, you can set it, or assign a value to it, which you store for later use in

the script in the variable that you declared. After you have set the value of the variable, you can change it at any point in the script. If you declare and set the value of more than one variable, you can write conditional statements to compare their values. For example, you might use a conditional statement to verify that a value exists in the database. If the conditional statement shows that the value does not exist, you can modify the script with commands to add the value.

Declaring a Variable

When you declare a variable, you give it a name that indicates the place in which a value that you set can be stored. You can insert the Declare variable command at any point in the script. Where you declare it determines its scope. However, you can change the scope of a variable by dragging it from one point in the script to another.

► To declare a variable

1. From the Command menu, choose Variables.

   Note
   You can perform this step at any point in the script.

2. In the command pane, type a name for the variable in the New Variable field.
3. In the Source of Value list, click Unknown/None.
4. Click the Insert button.

Note

After completing these steps, you have given the variable a name; but you have not yet assigned a value to it.

Changing the Scope of a Variable

The scope of a variable is the context within which you can use a value that you assign to the variable. The scope of the variable can extend locally, to a form or a single application, or globally, throughout the entire script, regardless of how many applications you launch. If you make the variable global, you can choose any point in the script to set its value, and you can use the value at any point in the script.

You can declare the variable at any point in the script. To change its scope, move it up or down in the hierarchy of script pane commands by clicking the Declare command and dragging it to the point that you choose.

As you drag the mouse, an arrow appears over the Declare command line. An arrow that points up indicates that the command line that you are dragging will be placed above the line that is highlighted when you release the mouse button. An arrow that points down indicates that the command line that you are dragging will be placed below the line that is highlighted when you release the mouse button.

► To change the scope of a variable

1. In the script pane of the AutoPilot form, click the Declare command line.
2. Drag the Declare command line to another context by clicking and holding the mouse button.

   To make the variable global, drag the Declare command to the top of the script.

3. When the Declare command line is on top of the Application command line and the arrow is pointing up, release the mouse button.

Setting the Value of a Variable

After you have declared the variable, you set its value. You store the value in the declared variable so that you can use it at points in the script that you determine by establishing the scope of the variable.

► To set the value of a variable

1. From the Command menu, choose Variables.
2. In the Existing Variable list of the command pane, choose the name of the declared variable to which you want to assign a value.
3. In the Source of Value list, choose one of the following sources for the value:
   - Literal Value
   - Valid Values List
   - Variable
   - Header Control Data
   - Grid Cell Data
4. If you assign a literal value, type that number or word into the Literal Value field in the command pane. If you created a list of valid values or declared and set the value of another variable, click the name of one of the values in the list. To derive the value from a header control or grid column, click the name of the control or column that populates the list.

   Note
   If you choose Grid Cell Data as the source of value, AutoPilot displays an unpopulated Row Number list and the grid columns for the form in which you are working. If you enter a grid row number in this list and click the Insert button, AutoPilot stores the value from the row that you specified.

5. Click the Insert button.

   AutoPilot sets the value that you chose and stores it in your declared variable.

Using a Variable in the Script

After you have declared a variable and set its value, you can use the value as a source of input. The scope that you established for the variable determines where you can use its value in the script.

► To use a variable in the script

1. In the header control or grid column list of the command pane, click a header control or grid column of a form to establish the context in which you want to input the value of the variable.
2. In the Source of Input list, choose Variable.
3. In the Variable list, choose the name of the variable that you declared and for which you set a value when you began the scripting process.
4. Click the Insert button.

   AutoPilot enters the variable that you set in the header control, grid column, or QBE line.

Updating the Value of an Existing Variable

You might need to change the value of an existing variable. If you have declared the variable and set its value, you can change the value at any point in the script.

► To update the value of an existing variable

1. In the script pane of the AutoPilot form, click the Set command line.

   You assigned a value to the declared variable on this line.

2. In the Variable list of the command pane, choose the name of the variable that you want to update.
3. Choose an option from the Source of Value list.
4. Choose or enter a value.
5. Click the Update button.

Setting Conditional Statements

To use AutoPilot to compare the values of two variables, click `If <var > == <var >` in the Command menu. When you do so, the command pane displays three populated lists that permit you to write a conditional If/Then statement, which also includes an Else statement. AutoPilot populates two of the lists with the names of the variables that you have declared. You write the left and right side of your conditional statement by choosing from each of these lists. To compare the left variable to a literal on the right, rather than a variable, enter a literal value in the Right Literal list.

The third list contains conditional operators such as equal to, not equal to, greater than, and so on. The command pane also includes a check box option, Numeric comparison. When you turn on this option, AutoPilot converts the variables that you have declared in words to numeric values before it compares them. To write a conditional statement that uses a string, rather than a numeric value, choose the option Is Not In from the list of conditional operators.

Note

You are not required to write any commands as part of the Else branch of the script. You write commands that are part of the Else branch when you want AutoPilot to run a series of commands only if the first part of your conditional statement is not true.

► **To set a conditional statement**

1. In the AutoPilot form, declare and set the value of two variables.

   Note
   You can declare the variables and set their values at any point in the script before you write the conditional statement.

2. From the Command menu, choose `If <var > == <var >`.
3. In the command pane, choose options from the following lists:
   - Left Variable
   - Operator
   - Right Variable
   - Right Literal
4. If the values of the variables are numeric, click the Numeric option.

   This step is optional.

5. Click the Insert button.

   AutoPilot enters the *If* portion of the conditional statement in the script pane. The *Then* and *Else* portions are blank.

6. Write and insert in the script the commands that constitute the *Then* branch of the conditional statement.
7. Click the Insert button.
8. If you want to add an *Else* condition, drag the insertion cursor beneath the *Else* command line in the script pane.
9. Write and insert in the script the commands that constitute the *Else* branch of the conditional statement.
10. Click the Insert button.

You can include branches of script for which the execution depends on the conditional statement.

Adding a Value to a Variable

You can add to a variable the value of another variable or a literal value. You can add a value to a variable whether you have declared a variable or both declared a variable and set its value. However, if you declare a variable with the intention of setting its value by adding a value later in the script, you should enter a default value so that AutoPilot knows the value to add to. If you want to add the value of one variable to another variable, you must have first declared and set the value of the variable that contains the value that you want to add.

► To add a value to a variable

1. From the Command menu, choose Variables.
2. Choose a variable from the Existing Variable list.
3. Choose one of the following from the Source of Value list:
   - Add Variable
   - Add Literal
4. To add a variable value, choose the name of a variable from the Variable list.
5. To add a literal value, enter a literal value in the Literal Value field.
6. Click the Insert button.

Subtracting a Value from a Variable

You can subtract from a variable either the value of another variable or a literal value. You can subtract a value regardless of whether you have set the value for the variable, provided that you set a default value when you declared the variable.

► To subtract a value from a variable

1. From the Command menu, choose Variables.
2. Choose a variable from the Existing Variable list.
3. Choose one of the following from the Source of Value list:
   - Subtract Variable
   - Subtract Literal
4. To subtract a variable value, choose the name of a variable from the Variable list.
5. To subtract a literal value, enter a literal value in the Literal Value field.
6. Click the Insert button.

Concatenating a Variable

Concatenating a variable enables you to create alphanumeric strings from two or more variables. You can construct a concatenated variable from other variables or from literal values. The following table illustrates the principle of variable concatenation:

Variable Name	Variable Value
X	1
Y	1
X concatenated from Y	11

Before You Begin

- Declare at least one variable, and either set its value or enter a default value.

► To concatenate a variable

1. From the Command menu, choose Variables.
2. Choose a variable from the Existing Variable list.
3. Choose one of the following from the Source of Value list:
   - Concatenate Literal
   - Concatenate Variable
4. If you want to concatenate a literal value, enter a literal value in the Literal Value field.
5. If you want to concatenate a variable, choose a variable from the Variable list.
6. Click the Insert button.

   AutoPilot creates a concatenated value for the variable that you chose from the Existing Variable list.

Creating a Variable to Confirm Validation Success

To confirm the success or failure of a validation, you declare a variable and choose Validation Success as a source of value. When you choose Validation Success, AutoPilot displays in the value selection list the names of the validations that you declared. When you run the validation, AutoPilot sets the value of the variable to 1 if the validation succeeds. If the validation fails, AutoPilot sets the value of the variable to 0. If you declare the validation and assign values, but do not run it, AutoPilot sets the value of the validation variable to 2.

With Watch List selected from the AutoPilot View menu, you can easily determine whether your data validation was successful following playback. The watch list displays the name of the validation variable in the Variable column. If the validation was successful, AutoPilot displays 1 in the Value column.

Before You Begin

- Declare and associate a validation. For AutoPilot to indicate success or failure by returning a value of 1 or 0, you must also run the validation. See *Scripting the Database Validation Command* in the *AutoPilot Guide*

► To create a variable to confirm validation success

1. From the Command menu, choose Variables.
2. Enter the name of a variable in the New Variable list or choose the name of a variable from the Existing Variable list.
3. Choose Validation Success (0/1) from the Source of Value list.
4. Choose the name of a validation from the value selection list.
5. Click the Insert button.

   If the validation runs successfully, AutoPilot displays in the watch list a value of 1 for the validation variable.

Creating a Variable to Store a Valid Values List Count

To capture and store the value that equals the number of items in a valid values list, you choose Valid Values Count as a source of value for a new or existing variable. After you choose this source of value, you choose a valid values list. AutoPilot stores the number of items in the list as the value for the variable.

To use the Valid Values Count as a source of value, you must create a valid values list, either by generating a simple database query or by creating a list of literal values.

See Also

- *Using a Valid Values List as a Source of Input* in the *AutoPilot Guide*

► To create a variable to store a valid values list count

1. From the Command menu, choose Variables.
2. Enter the name of a variable in the New Variable list or choose the name of a variable from the Existing Variable list.
3. Choose Valid Values Count from the Source of Value list.
4. Choose the name of a valid values list from the value selection list.
5. Click the Insert button.

   The watch list displays the number of items in the valid values list as the value of the variable.

6. To use the valid values count value to set the repeat count for a node, select the node in the script pane that contains the valid values list that you used to determine the value of the Valid Values Count variable.
7. In the Define repeat count list, choose Variable.
8. In the Repeat Count list, choose the variable that stores the valid values count value.
9. Click the Update button.

   AutoPilot updates the repeat count value, which now matches the item number value in the valid values list. When you play back the script, AutoPilot enters a value from the valid values list in a grid cell, one grid row at a time, until it has used each value in the valid values list.

Using a UDC Visual Assist Value as a Source of Input

Some header controls, grid columns, and QBE lines in forms contain UDC visual assists. When you choose UDC Visual Assist Value as a source of input, the UDCs in the value selection list in the command pane correspond to the codes in the visual assist forms.

The flashlight icon identifies the UDC visual assist, but J.D. Edwards ERP software also identifies other visual assists in this way. UDC Visual Assist Value appears in the Source of Input list only if a header control or grid column in the active form has a UDC visual assist.

► To use a UDC visual assist value as a source of input

1. From the Command menu, choose Set Header Control Value or Set Grid Cell Value.
2. In the command pane of the AutoPilot form, choose a header control or grid column from the Header Control or Grid Column list.

Note
These values correspond to the UDC values that appear when you click the visual assist icon for a control or column in a form. If your selection does not have a visual assist icon in the software, you will not see the UDC Visual Assist Value in the Source of Input list (step 3).

3. Choose UDC Visual Assist Value from the Source of Input list.
4. In the UDC Visual Assist Value list, choose a UDC value.
5. Click the Insert button.

   AutoPilot runs the code path in the software and inserts the UDC value in the script.

Using a Form Interconnect Visual Assist as a Source of Input

Some header controls and grid columns contain visual assists that require you to exit from the current application to a new one. If you have chosen a header control or grid column that has this type of visual assist, such as Address Book Master Search, AutoPilot displays Form Interconnect Visual Assist in the Source of Input list. If you choose this source of input, AutoPilot clicks the flashlight button in the form and writes an Application Interconnect command and Form command. You can then write any additional commands that you need.

Note

You do not use the value selection list when you choose this option. After you exit to a new application and form, you can write the additional commands that you need as part of your script.

► **To use a form interconnect visual assist as a source of input**

1. From the Command menu, choose either Set Header Control Value or Set Grid Cell Value.
2. In the command pane of the AutoPilot form, choose a header control or grid column from the Header Control or Grid Column list.
3. Choose Form Interconnect Visual Assist from the Source of Input list.
4. Click the Insert button.

   AutoPilot automatically writes a command to click the flashlight button in the active form, and then runs an Application Interconnect command and a Form command.

Clearing an Input from a Header Control or Grid Column

You can clear the contents of a header control or grid column. AutoPilot allows you to do this by choosing Clear from the Source of Input list in the command pane.

▶ To clear an input from a header control or grid column

1. From the Command menu, choose Set Header Control Value, Set Grid Cell Value, or Set QBE Cell Value.
2. In the AutoPilot command pane, choose the name of a header control or grid column from the Header Control or Grid Column List.
3. In the Source of Input list, choose Clear.
4. Click the Insert button.

   AutoPilot clears the value from the chosen header control or grid column.

Using the Value Selection List

After you choose a header control or grid column and a source of input for it, you complete the Type to command by choosing a value from the value selection list. AutoPilot displays this list when you choose a source of input and captions it according to the source of input that you choose.

If you want to input a literal value in the header control or grid column, type that value in the unpopulated value selection list. If you want to input the values that you assigned to a valid values list or variable, you choose the name of the list or variable from the value selection list. If you want to input a UDC visual assist value, you choose one from the value selection list. Finally, you might want to delete an entry in a control in a header or a processing option form or a grid column. AutoPilot allows you to do this by choosing Clear from the Source of Input list.

Assigning a Literal Value

The literal value that you assign as an input in a header control or grid column can be numbers, letters, special characters, or a combination of these. Verify that the literal value that you assign is a valid input.

▶ To assign a literal value

1. In the command pane of the AutoPilot form, choose a header control or grid column name from the Header Control or Grid Column list.
2. Choose Literal Value from the Source of Input list.
3. In the value selection list labeled Literal Value, input the value that you want to assign to the control or grid column.
4. Click the Insert button.

Assigning a Valid Values List Value

For input for a header control or grid column, you can assign a value from a valid values list. AutoPilot chooses the first value in the list the first time that you play back the script. If you choose to play back more than once, AutoPilot chooses the second value in the list on the second loop. This pattern continues until the loop ends or the item list is exhausted.

► To assign a valid values list value

1. In the command pane of the AutoPilot form, choose a header control or grid column name from the Header Control or Grid Column list.
2. Choose Valid Values List from the Source of Input list.
3. In the value selection list labeled Valid Values List, choose the name of a valid values list that you previously created.
4. Click Insert.

Assigning a Variable Value

You can declare a variable, set its value, and assign that value to a header control or grid column. After you set the value, AutoPilot stores it; it is available for you to use at any point in the script.

► To assign a variable value

1. In the command pane of the AutoPilot form, choose a header control or grid column name from the Header Control or Grid Column list.
2. Choose Variable from the Source of Input list.
3. In the value selection list labeled Variable, choose the name of a variable that you previously declared and for which you set a value.
4. Click the Insert button.

► To assign a UDC visual assist value

1. In the command pane of the AutoPilot form, choose a header control or grid column name from the Header Control or Grid Column list.
2. Choose UDC Visual Assist Value from the Source of Input list.
3. In the value selection list labeled UDC Visual Assist Value, choose the name of a UDC value.
4. Click Insert.

Assigning a Form Interconnect Visual Assist Value

AutoPilot enables you to choose a form interconnect visual assist value for those header controls or grid columns in forms that contain a visual assist that exits to a different application. You do not use the value selection list when you choose Form Interconnect Visual Assist as a source of input. Instead, after you exit to the different application, you script an input in a header control or grid column in the active form by using the Press Toolbar Button command to find and select a value.

► To assign a form interconnect visual assist value

1. In the command pane of the AutoPilot form, choose a header control or grid column name from the Header Control or Grid Column list.
2. Choose Form Interconnect Visual Assist from the Source of Input list.
3. Click the Insert button.

   AutoPilot presses the visual assist button on the header control or grid column that you have chosen, and then writes an Application Interconnect command and a Form command to the script pane.

4. From the Command menu, choose Set QBE Cell Value.
5. In the command pane, choose the name of a grid column from the Grid Column list.
6. Choose a source of input from the Source of Input list.
7. Enter or choose a value from the value selection list.
8. Click the Insert button.
9. From the Command menu, choose Press Toolbar Button.
10. In the command pane, choose Standard Button.
11. Choose Find.
12. Click the Insert button.
13. From the Command menu, choose Press Toolbar Button.
14. In the command pane, choose Standard Button.
15. Choose Select.
16. Click the Insert button.

    AutoPilot enters the value from the form interconnect visual assist in the header control of the form that you originally chose.

17. From the Command menu, choose Press Toolbar Button.
18. In the command pane, choose Standard Button.
19. Choose Close or Cancel.
20. Click the Insert button.
21. From the Command menu, choose Form.
22. In the command pane, choose the name of the form from which you exited when you scripted pressing the form interconnect visual assist button.

    Note
    This command confirms for AutoPilot that you have returned to the previous form. If you do not confirm the form, you cannot continue scripting.

23. Click the Insert button.

Scripting the Type to Command

After you choose from each of the three command pane lists, you click the Insert button to script the Type to command. AutoPilot uses the information that you chose to write two command lines in the script pane. One command line contains the context, either header, grid, or QBE, and the repeat count for the node. The other contains the name of the header control or grid column that you chose; a symbol that indicates whether you chose a literal value, a valid values list, a variable, or a UDC visual assist value as your source of input; and the value that you assigned.

Typing Data in a Header Control

You use the command pane lists to script inputs for the header portion of a form. Clicking Set Header Control Value in the Command menu establishes the header as the context in which you type data. The options that you choose from the lists in the command pane create the Type to action command. When you click the Insert button, the command line that specifies the header control as the context appears as a node. The command line identifies the control that you choose and the value that you type in it. AutoPilot inserts subsequent Type to commands subordinate to the node.

► To type data in a header control

1. From the Command menu, choose Set Header Control Value.
2. Choose the name of a control from the Header Control list.

   When J.D. Edwards software is running, a BlueCue highlights the control in the form that corresponds to the control that you chose in AutoPilot.
3. Choose a source of input from the Source of Input list.
4. In the value selection list, enter a literal value or choose the name of a valid values list, variable, or UDC visual assist value.
5. Click the Insert button.
6. Script inputs to additional header controls by clicking Set Header Control Value in the Command menu and repeating steps 1-5.

Clicking Options in a Header

Some forms contain options that you can click. You click Check Box/Radio Button in the Command menu when you want to write commands for these options in your script. The command to click an option is a different command than the Type to command, which you use to type data in header controls, grid columns, or QBE lines. However, when you work with a form that contains these options in its header, AutoPilot inserts the command in the Header node along with any Type to commands that you write.

► To script clicking options in a header

1. From the Command menu, choose Checkbox/Radio Button.
2. In the command pane, choose an option from the Radio Button or Check Box list.
3. If the option is a check box, click Check or Uncheck in the Source of Input list.

If the option is a radio button, the Source of Input list is unpopulated. AutoPilot clicks the radio button when you insert the command.

Typing Data in a Grid Column

You use the command pane lists to script inputs in the grid area of a form. Clicking Set Grid Cell Value in the Command menu establishes the grid area as the context in which you type data. The options that you choose from the lists in the command pane create the Type to action command. When you click the Insert button, the command line that specifies the grid detail area as the context appears as a node. The Type to command is indented beneath and attached to the node. The command line identifies the column that you chose and the value that you typed in it. AutoPilot inserts subsequent Type to commands subordinate to the node. You can script different inputs to multiple rows of the grid, or you can use a playback loop in AutoPilot to script the input in one row multiple times.

► To type grid column inputs

1. From the Command menu, choose Set Grid Cell Value.
2. In the command pane, choose a grid column from the Grid Column list.

   A BlueCue appears in the appropriate grid column in the form.

3. Choose a source of input.
4. Enter a literal value or choose the name of a valid values list, variable, or UDC visual assist value.
5. Click the Insert button.
6. Script inputs for additional grid columns by clicking the name of another column in the Grid Column list and repeating steps 1-5.

► To type inputs to additional grid rows

1. From the script pane in AutoPilot, click a Detail Information command line that you inserted in the script.

   The insertion cursor is connected to the node that you chose.

2. From the Command menu, choose Set Grid Cell Value.
3. In the command pane, choose a grid column from the Grid Column list.
4. Choose a source of input.
5. Enter a literal value or choose the name of a valid values list, variable, or UDC visual assist value.
6. Click the Insert button.
7. Repeat steps 1-6 each time that you want to script commands in a new row.

► To script a playback loop

1. Follow the steps for typing inputs in a grid row.
2. Turn off the Playback button in the tool bar (optional).
3. In the script pane, click the Detail Information command line that you want to play multiple times.

4. In the command pane, choose a source of input, such as literal value or variable, in the Define repeat count from list.
5. Enter a literal value or choose the name of a variable that you created.

   The value specifies the number of times that you want the inputs to loop, that is, to be entered in successive grid rows.

6. Click the Update button.
7. Click the Playback button (optional).

   During script playback, AutoPilot enters the inputs for a single row of the grid as many times as you specified in the Repeat Count list. For example, if the repeat count for Row 1 is 3, the system completes Rows 1 through 3 with the inputs that you originally scripted for Row 1.

Assigning a UDC Visual Assist Value

AutoPilot enables you to choose a UDC value for those header controls or grid columns in forms that contain UDC visual assists. The value selection form displays the valid UDC values for the control or column that you chose. If the control or column that you chose does not contain a UDC value, UDC Visual Assist Value does not appear as a choice in the Source of Input list.

Typing Data in a QBE Line

You can script commands to enter data in the QBE line of a form that has a QBE line. Clicking Set QBE Cell Value in the Command menu establishes the QBE line as the context in which you type data. The options that you choose from the lists in the command pane create the Type to action command. When you click the Insert button, the command line specifying the QBE line of the grid as the context appears as a node. The Type to command is indented beneath and attached to the node. The command line identifies the grid column that you chose and the value that you typed in it. AutoPilot inserts subsequent Type to commands subordinate to the node. You can script different inputs in multiple rows of the grid, or you can script the input in one row multiple times, using a playback loop in AutoPilot.

► To type data in a QBE line

1. From the Command menu, choose Set QBE Cell Value.
2. In the Grid Column list in the command pane, choose the name of a grid column.
3. Choose a source of input.
4. Enter a literal value, or the name of a valid values list, variable, or UDC visual assist value.
5. Click the Insert button.
6. To script an input in the QBE line of another grid column, choose another name in the Grid Column list and repeat steps 1-5.
7. From the Command menu, choose Press Toolbar Button.
8. Choose Press Standard Button.
9. Choose Find.
10. Click the Insert button.

The Select Grid Row Command

The Select Grid Row command allows you to perform and test several important functions. You use it to work within a detail area of a form. You can select records, delete them, add to a grid row, or edit the entries in a grid row. You can also script an exit to another form in another application. You use this command, therefore, in conjunction with several other action and context commands, including Press Toolbar Button, Application Interconnect, and Grid Data.

AutoPilot allows you to click either a row or a grid cell, specify the row number or grid column, and perform a specific action, such as double-clicking the row or editing the content of the cell.

Note

When you choose the Select Grid Row command, AutoPilot also populates the command pane with a value selection list that allows you to enter a literal value or choose the name of a previously-created valid values list or variable. The caption of this list changes to reflect the option that you choose in the Source of Row Number list.

Operation Type List

The Operation Type List in the AutoPilot command pane allows you to choose a row in a grid area either by specifying a row number or by specifying the value of a particular cell in a particular grid column.

Click by Row Number Option

The Click by Row Number option allows you to choose a grid row by number. Because you specify a single number, this option works particularly well for grid areas with a large number of rows. AutoPilot finds the designated row and performs the action that you chose in the Action on grid row list. You can designate the grid row either by entering a literal value, a value from a valid values list, or a value from a variable in the value selection list.

Click by Cell Content Option

To choose a grid row that contains a particular value, such as an item number, use the Click by Cell Content option and choose a grid column and an action that you want to perform on the cell. AutoPilot selects the grid cell rather than the entire grid row.

Action on Grid Row List

The Action on grid row list allows you to specify the purpose for choosing the row. In AutoPilot, you can script the following types of grid row operations:

- Single-click a grid row.
- Single-click a grid row button.
- Double-click a grid row.

- Double-click a grid row button.
- Position grid row for add or edit.

Single-Click a Grid Row

You write a command to single-click a grid row when the form that is active does not have a row button. Although you can write this command when the active form has a row button, J.D. Edwards does not recommend it because single-clicking the grid row sometimes selects only a cell.

Single-Click a Grid Row Button

You write a command to single-click the grid row button when the form that is active contains rows with button. Clicking the grid row button selects the row, rather than a cell. You do not script this grid row operation in forms that do not contain a button.

Double-Click a Grid Row

If you want to move from the detail area of one form to another form, exit to a new application, or double-click a row to perform a process, you write a command to double-click the grid row.

Double-Click a Grid Row Button

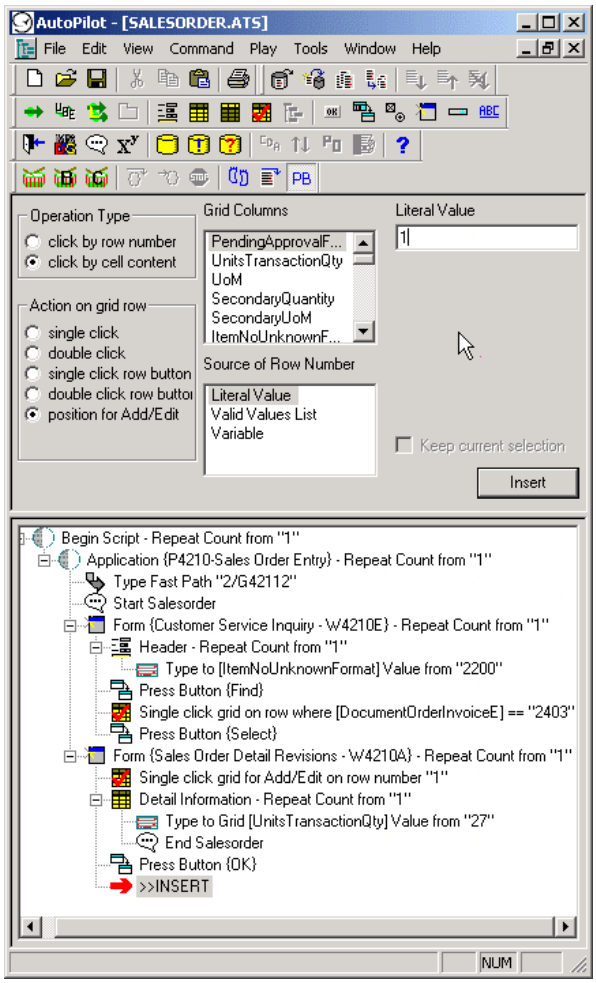
You write a command to double-click the grid row button when the form that is active contains grid rows with buttons. You do not script this grid row operation in forms that do not contain grid row buttons.

Position Grid Row for Add/Edit

To enter changes in the detail area of a form, you write a command to position the grid row for an add or an edit. You use the command to add an entry to a grid row or to edit an existing value in a grid cell.

Grid Columns List

You use the Grid Columns list only when you have chosen the Click by Cell Content option. When you choose this option, AutoPilot populates the Grid Columns list with the names of all the columns in the grid that is active. You can scroll through this list to find the name of a column.



After you choose Click by Cell Content, you can specify in the Source of Row Number list a value that might exist in a particular cell of that column. AutoPilot searches for the value and selects the first row that contains that value in the specified column.

Source of Row Number List

In the Source of Row Number list, you choose the source of the value that you use to select the grid row. You can choose Literal Value, in which case you type a row number or a grid cell value in the value selection list. You can choose Valid Values List, in which case you choose from the value selection list a valid values list that you previously created. AutoPilot uses the first value in the list to select a grid row either by row number or by cell content. You can choose Variable, in which case you choose from the value selection list a variable for which you have declared a name and set a value. AutoPilot uses this value to select a grid row either by row number or by cell content.

Scripting the Select Grid Row Command

To begin the scripting, open a form. Populate the detail area by writing a command to click Find. When the grid is populated, choose in the AutoPilot command pane the row that you want to target for your script and the operation that you want to perform on that row. You can select a grid row either by row number or by cell content.

In either case, you choose the type of operation that you want to perform on the grid row. Operations include single-clicking the row, single-clicking the row button, single-clicking to perform an add/edit, double-clicking the row, or double-clicking the row button.

To finish scripting the Select Grid Row command, choose a source of value for row selection. These sources include a literal value, a valid values list, or a variable. You then either enter a literal value in the value selection list or choose from the value selection list a valid values list or variable.

Clicking by Row Number

After you populate the grid, you can use AutoPilot to select a row by searching for a row number that you designate. To complete this command, you must also choose an action that you want AutoPilot to perform on the grid row, choose a source of value that AutoPilot uses to select the row, and choose or enter the value of the row.

► To click a grid row by row number

1. From the Command menu, choose Press Toolbar Button.
2. Choose Press Standard Button.
3. Choose Find.

   AutoPilot fills the detail area in the active form.

4. From the Command menu, choose Select Grid Row.
5. In the AutoPilot command pane, choose Click by Row Number from the Operation Type list.
6. Choose a grid row action from the Action on Grid Row list.
7. Choose a value source from the Source of Row Number list.
8. In the value selection list, enter a literal value or choose the name of a valid values list or variable.
9. Click the Insert button.

Clicking by Cell Content

You might want AutoPilot to search the detail area for a particular value, and then select the row after it finds a cell that contains that value. In this case, you can write a command to click the grid row using cell content rather than row number.

Writing a command to select a grid row based on cell content involves the same steps that you use to write a command to select the row based on row number. However, when you select a row based on cell content, you must also choose a grid column as a search criterion.

► To click a grid row by cell content

1. From the Command menu, choose Press Toolbar Button.
2. Choose Press Standard Button.
3. Choose Find.

   AutoPilot fills the detail area in the active form.

4. From the Command menu, choose Select Grid Row.
5. In the AutoPilot command pane, choose Click by Cell Content from the Operation Type list.
6. Choose a grid row action from the Action on Grid Row list.
7. In the Grid Columns list, choose a grid column in which you want AutoPilot to search for a value.
8. Choose a value source from the Source of Row Number list.
9. In the value selection list, enter a literal value or choose the name of a valid values list or variable.
10. Click the Insert button.

Performing Grid Row Operations

After you select a row, you can perform the following operations on it:

- Single-click a grid row.
- Single-click a grid row button.
- Double-click a grid row.
- Double-click a grid row button.
- Position a grid row for add or edit.

Note

Choose the single-click and double-click grid row operations when you are writing commands to test a form that contains a detail area that does not have row buttons.

You can use these options before you write other AutoPilot commands.

For example, single-clicking on a grid row or grid row button enables you to write a command to press the Select or Delete button. Double-clicking a row enables you to move to another form or application. Clicking on a row and positioning for add or edit allows you to edit the selected grid row

► To single-click a grid row

1. From the Command menu, choose Select Grid Row.
2. Choose an operation from the Operation Type list.
3. Click the single click option in the Action on Grid Row list.
4. Choose a value source from the Source of Row Number list.

5. In the value selection list, enter a literal value or choose a valid values list or variable.
6. Click the Insert button.

► To double-click a grid row

1. From the Command menu, choose Select Grid Row.
2. Choose an operation from the Operation Type list.
3. Click the double-click option in the Action on Grid Row list.
4. Choose a value source from the Source of Row Number list.
5. In the value selection list, enter a literal value or choose a valid values list or variable.
6. Click the Insert button.
7. From the Command menu, choose either Form or Application Interconnect.
8. In the command pane, choose the next form or application that appears when you double-click the row in the form.
9. Click the Insert button.

► To single-click a grid row button

1. From the Command menu, choose Select Grid Row.
2. Choose an operation from the Operation Type list.
3. Click the single click row button option in the Action on Grid Row list.
4. Choose a value source from the Source of Row Number list.
5. In the value selection list, enter a literal value or choose a valid values list or variable.
6. Click the Insert button.

► To double-click a grid row button

1. From the Command menu, choose Select Grid Row.
2. Choose an operation from the Operation Type list.
3. Choose the double click row button option in the Action on Grid Row list.
4. Choose a value source from the Source of Row Number list.
5. In the value selection list, enter a literal value or choose a valid values list or variable.
6. Click the Insert button.
7. From the Command menu, choose either Form or Application Interconnect.
8. In the AutoPilot command pane, choose the next form or application that appears when you double-click the row button on the form.
9. Click the Insert button.

► To position a grid row add or edit

1. From the Command menu, choose Select Grid Row.
2. Choose an operation from the Operation Type list.
3. Choose Position for Add/Edit in the Action on Grid Row list.

4. Choose a value source from the Source of Row Number list.
5. In the value selection list, enter a literal value or choose a valid values list or variable.

Note
If you are adding a row, enter a high row number, such as 999, to ensure that you reach the bottom of the grid when AutoPilot inserts the command.

6. In the Command menu, click Set Grid Cell Value.
7. Choose a grid column in the Grid Colum pane.
8. Choose a source of input in the Source of Input pane.
9. Enter a literal value or UDC value, or choose a valid values list or a variable from the value selection list.
10. Click the Insert button.

    AutoPilot adds to or edits the column in the grid row that you selected.

The Press Toolbar Button Command

The Press Toolbar Button command allows you to script many of the important actions that you can take in a form, including the following:

- Moving from one form to another within the same application
- Moving to a form in a new application, using either the Form menu or the Row menu
- Populating a grid on a form
- Choosing a row in a grid
- Updating the database
- Deleting a record
- Exiting a form
- Choosing a grid tab
- Clicking the grid scroll button
- Submitting UBEs that are not hard-coded to run automatically

The Standard Button Option

The Standard Button option in AutoPilot contains button-clicking choices that match the buttons in the tool bar of forms. For example, the tool bar in some forms features 10 button-clicking options. When one of these forms is active, the Standard Button tree in the Button list in AutoPilot contains the same options.

When you script one of these button-clicking options, AutoPilot runs the command exactly as it would be run in J.D. Edwards ERP software. For example, you might script clicking OK to update the database after you have entered new data on a form.

When you choose the Standard Button option, the Next Form list also appears. The option that you choose from this list indicates the form that becomes or remains active when you click the Insert button.

Note

Clicking a standard button, such as Add, occasionally takes you to another application. In this case, you must write an Application Interconnect command instead of completing the Next Form field.

When you launch a UBE, the standard button choices in AutoPilot also match the buttons in the menu bar of the form. For example, when you need to submit a UBE using the Version Prompting form, choose Submit from the Standard Button options in AutoPilot.

See Also

- *UBE Submission* in the *AutoPilot Guide*
- *Submitting a UBE* in the *AutoPilot Guide*

The Custom Button Option

Use the Custom Button option to scripting making a choice from the Row menu or the Form menu. Making a choice from the Row menu or the Form menu usually results in moving to a different application, in which case you must also script an Application Interconnect.

Although you use both the Custom Button option and the Application Interconnect command to interconnect applications, they perform slightly different functions. You script an Application Interconnect command *after* you have exited to a new application, for example by clicking the Add button. You can use the Custom Button option *before* you exit to a new application. The Custom Button option allows you to choose the application and form in the command pane and insert the commands. AutoPilot interconnects the applications, and the form in the new application appears.

When you choose Custom Button from the Button list in the command pane, the following options appear in the tree:

- Form
- Row

Clicking either or both of these options further expands the tree and displays the available menu choices.

Note Concerning Custom Buttons and the Command Line

The options under Form and Row correspond to the options that appear in the drop-down menu when you click Form or Row in the menu bar of the active form. When you insert a command to press a standard button and to press a custom button, the same symbol appears in the command lines of the script. However, the command line for clicking a standard button describes the command as Press Toolbar Button, whereas the command line for clicking a custom button describes the command as Select Menu Exit. When you press a standard button, you usually travel between or within forms in the same application. When you press a custom button, you usually change applications.

The command line for clicking a custom button contains the type of menu exit, either Form or Row, and the name of the menu item that you select. The Select Menu Exit command line

should be followed by the Application Interconnect command line, which records the application to which you have exited.

Form Exit

AutoPilot allows you to script making a choice from the Form menu, just as you might do directly in J.D. Edwards ERP software. Typically, you perform a form exit when you want to move to a form that is related to the one on which you have been working. However, that form might be in a different application, and the exit represents a change in the standard sequence of forms that you access when you perform a transaction.

Row Exit

AutoPilot allows you to script making a choice from the Row menu, just as you might do directly in J.D. Edwards ERP software. Typically, you perform a row exit when you want to move from a chosen grid row in a form to a related form. That form might be in a different application, and the exit represents a change in the standard sequence of forms that appear when you perform a transaction.

You might also use the Custom Button option to perform a row exit that launches a UBE version. In this case, you write a UBE command after you script the row exit, and AutoPilot automatically submits the UBE.

See Also

- *Launching a UBE from a Row Menu* in the *AutoPilot Guide*

The Select Grid Tab Option

You use the Select Grid Tab option in AutoPilot to test whether J.D. Edwards ERP software moves to customized grid tabs that you created. In J.D. Edwards ERP software, you customize the grid by choosing Preferences, Grid, and New Format. To create a new tab, you customize fonts, the number of grid columns, the width of grid columns, and so on. Each grid customization creates a new tab. After you create as many custom tabs as you need, you can use AutoPilot to script selecting a tab or tabs. Not only can you determine whether the tab is chosen, you can also determine whether your customized changes appear.

The Grid Scroll Button Option

You use the Grid Scroll Button option in AutoPilot to script clicking the up and down arrows in the detail area of a form. AutoPilot moves the arrows up or down by line or by page.

Scripting the Press Toolbar Button Command

You use the Press Toolbar Button command in AutoPilot to script many important functions. For example, clicking Add allows you to move to a new form, either in the same application or a new one. By choosing the Standard Button option for the Press Toolbar command, you can write a script command to click Add. Other Press Toolbar Button options allow you to perform form and row exits, select a grid tab, or scroll through a grid.

Clicking a Standard Button

In general, you choose the Standard Button option when you want to click a toolbar button in a form. When you choose this option, the choices in the command pane match the toolbar buttons on the form that is active.

See Also

- *Submitting a UBE* in the *AutoPilot Guide*
- *Selecting Data for a UBE* in the *AutoPilot Guide*

► To move from one form to another using the Add button

1. From the Command menu, choose Press Toolbar Button.
2. Choose Standard Button, then choose Add.
3. Choose the name of a form from the Next Form list or choose Unknown/None for another application.

   Caution
   If you choose Unknown/None and then click the Insert button, ensure that the Form command line in AutoPilot matches the form that is now active in the software. From the Command menu, choose Form, choose the name of the active form, and then click the Insert button.

4. Click the Insert button.

   The next form appears.

   If you choose Unknown/None and the form that appears in the software is part of a different application, complete steps 5 through 8 to script an Application Interconnect command.

5. From the Command Menu, choose Application Interconnect.
6. From the Application list in the command pane, choose the name of the active application.
7. From the Form list in the command pane, choose the name of the active form.
8. Click the Insert button.

► To move from one form to another using the Select button

1. From the Command menu, choose Select Grid Row.
2. In the Source of Row Number list in the command pane, choose literal value, valid values list, or variable.
3. To specify the row number to select, enter or choose a value from the value selection list.
4. Click the Insert button.
5. From the Command menu, click Press Toolbar Button.

6. Choose Standard Button, and then choose Select.
7. In the Next Form list, choose the form that will appear next.
8. Click the Insert button.

Note
If you choose Unknown/None from the Next Form list, use the Form command to confirm the new form after it appears. If the form that appears in the software is part of a different application, you must script an Application Interconnect command.

► To update the database

1. From the Command menu, choose Press Toolbar Button.
2. Choose Standard Button, and then choose OK.
3. If you are moving to another form, choose it from the Next Form list. If you are staying on the same form, do not choose from the list.
4. Click the Insert button.

► To fill a grid

1. From the Command menu, choose Press Toolbar Button.
2. Choose Standard Button, and then choose Find.
3. Click the Insert button.

► To delete a record

1. From the Command menu, choose Select Grid Row.
2. In the Source of Row Number list in the command pane, choose literal value, valid values list, or variable.
3. To specify the row number to be selected, enter or choose a value from the value selection list.

   The row number should contain the record that you want to delete.
4. Click the Insert button.
5. Choose Press Toolbar Button from the Command menu.
6. Choose Standard Button, and then choose Delete.
7. Click the Insert button.

► To exit a form

1. From the Command menu, choose Press Toolbar Button.
2. Choose Standard button, and then choose either Cancel or Close, depending on the button that is available on the form.
3. Click the Insert button.

Clicking a Custom Button

You use the Custom Button option to script making choices from the Row menu and the Form menu. When you choose this option, the choices in the command pane match the form and row exits in the form that is active.

Remember that a form or row exit might result in an application interconnect. To script the application interconnect, choose from the command pane the menu choice, the next form, and the application before you click the Insert button.

► To script a form exit

1. From the Command menu, choose Press Toolbar Button.
2. Choose Custom Button, and then choose Form.
3. Choose a form exit.
4. If the form exit results in the system launching a new application, choose the new application from the Application list.
5. From the Next Form list, choose a form, if necessary.
6. Click the Insert button.

► To script a row exit

1. From the Command menu, choose Press Toolbar Button.
2. If the detail in the active J.D. Edwards software form is empty, choose Find.
3. Click the Insert button.

   In the J.D. Edwards software form, the detail area loads.

4. From the Command menu, choose Select Grid Row.

   Note
   To display the new form, you can choose Row from the menu bar in OneWorld, click one of the forms in the list, or you can click Select.

   You can determine the name of the newly active OneWorld application and form by clicking About OneWorld in the menu bar.

5. Choose single-click row button from the Actions on grid row pane.
6. In the Source of Row Number list in the command pane, choose literal value, valid values list, or variable.
7. To specify the row number to be selected, enter or choose a value from the value selection list.
8. Click the Insert button.
9. From the Command menu, choose Press Toolbar Button.
10. From the Button list in the AutoPilot command pane, click Custom Button.
11. Click Row.

12. Choose a Row exit.

Note
If you run a row exit to launch a UBE, you do not choose from the Application or Next Form lists that appear. You choose the row exit only, click the Insert button, and then write a UBE command.

13. If you are exiting to an interactive application, choose that application from the Application list.
14. Choose a form name from the Next Form list.
15. Click the Insert button.

Selecting a Grid Tab

You can choose customized grid tabs by using the Select Grid Tab command. AutoPilot chooses the grid tab number that you script and, in playback mode, displays the grid with your customized changes.

► To script the Select Grid Tab option

1. In a form, create as many tabs as you need.
2. From the Command menu, choose Press Toolbar Button.
3. In the Button list, click Select Grid Tab.
4. In the Literal Value list, enter the number of the grid tab that you want to choose.

   The first grid tab to the left is 1. The second grid tab to the left is 2, and so on.

5. Click the Insert button.

Clicking the Grid Scroll Button

When you are working in a form with a populated detail area, you might want to scroll through the detail area from top to bottom or from page to page.

► To script the Grid Scroll Button option

1. From the Command menu, choose Select Grid Row.
2. In the command pane, under Operation Type, choose click by row number.
3. Under Action on grid row, choose single click.
4. Choose a source of row number and enter a literal value or choose a variable or valid values list.
5. Click Insert.
6. From the Command menu, choose Press Toolbar button.
7. Choose Grid Scroll Button.

8. Choose one of the following scrolling options in the tree:
    - Page Up
    - Page Down
9. Click Insert.

The Press Push Button Command

You can use the Press Toolbar Button command to script clicking standard toolbar buttons, performing custom functions, selecting grid tabs, and selecting grid scroll buttons. However, some applications contain special buttons that do not appear on the toolbar or menu bar.

You use the Press Push Button command to script clicking these oversized push buttons and clickable bitmaps. AutoPilot displays in the command pane the push button options and clickable bitmaps in the active form.

Push Button Options

Some applications, such as System Setup (P0000), contain forms that use oversized push buttons. Pushing these buttons allows you to choose forms on which you can set up, for example, constants for general accounting, accounts payable, accounts receivable, and so on.

Note Concerning the Press Toolbar Button Command

You cannot click these buttons using a Press Toolbar Button command because they do not appear on the toolbar. Therefore, if you click Press Toolbar Button in the AutoPilot command menu, none of these push button options populates the command pane. Alternatively, if you click Press Push Button, the command pane displays in the Select Button to Press list the push button options that appear on the form.

When you write a command to press a push button, you often move to a new form. You might need to write a Form command for the new form so that your AutoPilot script matches the actions that you took in J.D. Edwards ERP software. Sometimes, when you press a push button, you exit to a new application. In that case, you must write an Application Interconnect command in AutoPilot. You can verify whether you have exited to a new application by clicking Help and About J.D. Edwards in the tool bar of the active form.

Clickable Bitmap Options

Some applications, such as Cross Reference (P980011), use clickable bitmaps that enable you to move to a new form. Because you cannot click these options by using a button on the tool bar, you cannot use a Press Toolbar Button command.

Instead, to click a bitmap option, you use the AutoPilot Press Push Button command. If a form that contains clickable bitmaps is active, the AutoPilot command pane displays the available bitmaps.

Note

AutoPilot displays the system-assigned name, such as Bitmap 184, for each clickable bitmap, rather than the descriptive label that appears next to each one on the form. In addition, the control properties for the bitmaps do not indicate their identities. Therefore, you must identify the particular bitmap that you want AutoPilot to click. To help you identify the name of the bitmap, a BlueCue appears around the descriptive label on the form when you click the corresponding name in the AutoPilot command pane.

When you script pressing a clickable bitmap, you typically move to another form and must write a form command in AutoPilot to match the actions that you took. If you exit to a new application when you press a clickable bitmap, you must write an Application Interconnect command in AutoPilot so that the current application in your script matches the active application.

Scripting the Press Push Button Command

Writing a Press Push Button command allows your script to perform actions that you cannot script with the Press Toolbar Button command. You need to script a Press Push Button command when you are working on a script that tests applications and forms that contain push buttons and clickable bitmaps that do not appear on the tool bar of the form. Writing a command to press a push button or to click a bitmap enables you to move to another form in the application. In some cases, this command enables you to exit to another application.

Pressing a Push Button

You might want to test an application, such as System Setup (P0000), which contains forms that display oversized push buttons that do not appear on the tool bar. In most cases, you push these buttons to move to a new form in the same application or to exit to a new application.

Scripting a Press Push Button command in AutoPilot enables you to push the button in the form. You cannot write a Press Button command to perform this action because the Press Button command applies only to buttons that appear on the tool bar.

► To press a push button in a form

1. From the Command menu, choose Press Push Button.

   AutoPilot populates the command pane with push button choices only when the active form contains push buttons.

2. In the AutoPilot command pane, choose a push button from the Select Button to Press list.
3. Click the Insert button.
4. If pressing the push button results in a move to another form in the same application, from the command menu choose Form.
5. In the Form list of the command pane, choose the name of the active form.

6. Click the Insert button.
7. If pressing the push button results in exiting to a new application, from the Command menu, choose Application Interconnect.
8. In the command pane, choose one of the following:
   - Application (choose the active application)
   - Form (choose the active form)
9. Click the Insert button.

Clicking a Bitmap

Some applications contain forms that use clickable bitmaps that do not appear on the tool bar. To test one of these applications, you must script a Press Push Button command to click one of the bitmaps. AutoPilot displays in the command pane the system-assigned name for each bitmap, rather than the descriptive text that appears next to each bitmap. If you click the system-assigned name in the command pane, AutoPilot identifies the corresponding clickable bitmap on the form by enclosing it in a BlueCue.

Scripting a Press Push Button command to click a bitmap enables you to move to another form or to exit to another application. You cannot click the bitmap by writing a Press Button command because the bitmaps do not appear on the tool bar.

► To click a bitmap in a form

1. From the Command menu, choose Press Push Button.
2. From the Select Button to Press list, choose the name of a tab in the active form.
3. Click the node next to the tab name.
4. From the drop-down menu, choose the system-assigned name of a clickable bitmap.

   Note
   When you click the bitmap name, the BlueCue that appears in the active form identifies the corresponding bitmap.

5. Click Insert.
6. If pressing the push button results in the system calling another form in the same application, from the Command menu, choose Form.
7. In the Form list of the AutoPilot command pane, choose the name of the active form.
8. Click Insert.
9. If pressing the push button results in exiting to a new application, click Application Interconnect in the Command menu.
10. In the command pane, choose one of the following:
    - Application (choose the active application)
    - Form (choose the active form)
11. Click Insert.

The Select ComboBox Item Command

Some applications, such as Object Management Workbench (P98220) and Expense Report Review / Entry (P09E2011), use combo box controls. These controls can appear on forms as edit text fields, pop-up menus, or scrolling lists. AutoPilot provides the text from the combo box in the command pane. After you write a Select ComboBox Item command and play back the script, AutoPilot locates the combo box and sends a message to the form to select the text string that you specified in the command pane.

When you choose the command, AutoPilot populates the Combo box list in the command pane with either a list or a tree control. A tree control appears in the command pane only when the combo box is under tab controls in the active form.

In the Combo box list, when you click the name of a control or an item in a scrolling list or pop-up menu, AutoPilot populates the Choices list with the text names that appear in the combo box, along with the user defined system codes, which it retrieves from the User Defined Codes table (F0005).

Forms in some applications have hidden combo box controls that are not used. Nonetheless, AutoPilot displays these controls in the Choices list, just as it displays hidden header controls and grid columns in the command pane. You cannot choose a default value, such as None, to enter in the combo box.

Scripting the Select ComboBox Item Command

When you write scripts that use forms that contain combo box lists, you choose the Select ComboBox Item command from the Command menu of the AutoPilot form and choose a combo box from the Combo box list in the command pane.

Select ComboBox Item is active in the Command menu of the AutoPilot form only if you launch an application and form that use combo boxes.

After you choose an option from the Combo box list, AutoPilot populates the Choices list with the available items in the combo box.

Some applications, such as Object Management Workbench (P98220), use more than one combo box list, and the lists depend on one another to establish, for example, search criteria. In this case, the choice that you make in the Combo box list in the AutoPilot command pane changes the items in the Choices list.

► **To script the Select ComboBox Item command**

1. From the Command menu, choose Application.
2. In the command pane, choose an application and Fast Path, and then click Insert.
3. From the Command menu, choose Select ComboBox Item.
4. From the Combo box list, choose a combo box.

   The system populates the Choices list with the items in the combo box.
5. From the Choices list, choose a combo box item.
6. Click Insert.

   AutoPilot enters the item from the combo box list to the control in the form.

The Build Tree Path Command

Some applications use tree path controls. To write script commands for forms in these applications, you must use the Build Tree Path command to create a unique path that uses the tree path in a form.

You use the Build Tree Path command using any combination of literal text or variables. For example, the first node in a tree might consist of a parent, one child, and one grandchild. To use AutoPilot to write the Build Tree Path command, you first designate the data type that represents the first node in the tree path. Available data types include the following:

- Literal values, which are the precise text that designates a node in a tree control
- Variable values, which you set as the text that designates a node in a tree control
- Ordinal values, which represent the order in which a node appears in a tree path, such as first, second, third, and so on

You choose ordinal as the data type when you want to build a path to the first node in the tree (the parent), the first child of the parent, and the first grandchild of the parent. Clicking the Add button populates the tree path list with a leaf node, which is a node without children. You can create parent-child relationships by clicking the Add button to add nodes.

During playback, AutoPilot uses the search string that you create to identify the coordinates of each node in the tree in the active form. You can modify the tree path as necessary using the Add and Remove buttons. Removing the parent node also removes all children from the path. If you want to add a node, you must add it to a leaf node.

If you attempt to add a child node to a parent that already has a child, AutoPilot displays a dialog box that indicates that you that cannot add a child to the node. You must click a leaf node to create a new node with a child.

Scripting the Build Tree Path Command

You use the Build Tree Path command to write scripts that test applications that use tree controls. You can use any combination of variables or literal text to build a unique path to nodes that exist in the tree path in an active form. You can also modify the tree path by adding and removing nodes from the path that you build.

Building a Tree Path Using Variable Values

You can choose one of the following methods to use variable values to build a tree path:

- Declare a variable and set its value as the text that represents a node in the tree. In this case, you choose Variable as the data type for the node.
- Set a numeric value for the variable. If you choose 3 as the value, AutoPilot chooses the third node in the tree or the third child of a parent. In this case, you choose Ordinal as the data type for the node.

Whether you choose Variable or Ordinal as the data type for the node, the names of the variables you declared appear in the Select Variable list of the command pane.

Before You Begin

- ❑ Declare a variable and set its value. See *Using a Variable as a Source of Input* in the *AutoPilot Guide*

► To build a tree path using variable values

1. From the Command menu, choose Variables.
2. Declare a variable and set its value.

   Note
   You can assign any value to the variable. However, remember that if you assign a numeric value, that value represents the position of a parent node or a child in the tree path. For example, if you set the value of the variable to 3, the value represents the third node in the tree control.

3. When a form that uses a tree control is active, from the Command menu, choose Build Tree Path.

   The AutoPilot command pane contains a Tree Path list and a Data Type list.

4. Choose one of the following from the Data Type list:
   - Text Variable

     Choose Text Variable if the variable that you want to use has a text value.

   - Ordinal Variable

     Choose Ordinal Variable if the variable that you want to use has a numeric value.

5. From the Select Variable list, choose a variable for which you have set a value.
6. Click Add.

   AutoPilot inserts the variable as a node in the Tree Path list.

Building a Tree Path Using Literal Values

You can build a tree path using a literal value as the data type to represent a node. You type the name of the node in the Enter Node list exactly as it appears on a form. To create the tree path, you can use literal values as a data type in combination with variable and ordinal values.

► To build a tree path using literal values

1. In the AutoPilot form, launch an application and form that uses tree controls.
2. From the Command menu, choose Build Tree Path.
3. Choose one of the following from the Data Type list:
   - Text Literal

     Choose Text Literal if you want to use a text value.

   - Ordinal Literal

     Choose Ordinal Literal if you want to use a numeric value.
4. In the Enter Node list, type a literal value.
5. Click Add.

   AutoPilot inserts the literal value as a node in the Tree Path list.

   When you play back the script, AutoPilot finds the node with the literal value, based on the tree path that you built.

Adding a Parent Node or Child to a Tree Path

After you have chosen a data type and entered a literal value or selected a variable, you add a tree path node by:

- Clicking the Add button with the Tree Path list in the command pane unpopulated
- Choosing a node in the tree path list that does not have a child and clicking the Add button.

Remember that you cannot add to a node that already has a child.

► To add a parent node or child to a tree path

1. In the Tree Path list of the AutoPilot form command pane, click a node that does not have a child.
2. Choose one of the following from the Data Type list:
   - Text Literal
   - Text Variable
   - Ordinal Literal
   - Ordinal Variable
3. Choose a variable from the Value Selection list or enter a literal value in the Enter Node list.
4. Click Add.

► **To remove a parent node or a child from a tree path**

1. In the Tree Path list of the AutoPilot form command pane, click a parent node or a child.
2. To remove a parent node and its child, click the parent node, and then click Remove.
3. To remove the child only, click the child, and then click Remove.

Removing a Parent Node or a Child from a Tree Path

You can modify your tree path by selecting a node or a child and clicking the Remove button. Remember that removing a node also removes the child of the parent.

The Database Validation Command

Use the data validation action commands to verify the data that you entered in the database using commands in your AutoPilot script.

The data validation process in AutoPilot consists of the following steps:

- Declaration, in which you give the validation a name.
- Association, in which you pair the values that you enter in header controls or grid columns with columns in a database table.
- Execution, in which you use record selection criteria to verify whether the records that you entered during scripting actually were entered in the database.

You can also verify that the system deleted records from the database.

Validation Definition

You script validation commands to compare a data set that you created in AutoPilot and ran in J.D. Edwards ERP software with a data set that was written to the database. These commands confirm that the system has entered records in the database as you expected. You also use validation for process testing, to verify that data moved as you intended it to move through a sequence of applications that are included in a transaction cycle. Finally, you use validation to validate values that cannot be accessed through AutoPilot, such as century. In most transaction fields, you make no entry for century. Using validation commands, you can verify that century data actually appears in the database.

Validation Declaration

You script a new database validation command each time that you declare a validation. You can declare a validation at any point in the script.

The declared validation includes:

- A validation name that you choose
- A table to be validated, which you choose from a list

Declaring the validation is similar to declaring a variable in that it provides only a name for the validation. In essence, it provides the room in which you later store values.

Validation Association

Associating a variable enables you to store data. You can gather this data from many different points in the script. No action is taken when you associate a validation. Rather, you open the room that you created when you declared the validation and store in that room data of your choosing.

In associating a validation, you define the values to be validated against chosen columns in the database. You choose a column, which cannot be associated more than once in a script, such as ABALPH (NameAlpha in the Address Book program (P01012)) and associate that column with a value, which can be derived from a literal value, variable, header control, or grid column. You then choose a database value type that AutoPilot uses to validate the data that you enter in the database when you run your script.

You must specify the following information to script the validation association:

- Validation name (declared validation)
- Database column identification, which you choose from a list
- Source of expected data, such as a header control
- Database value type, either key selection or validation value

Validation association can occur at as many different points in the script as you choose, but you must use both a key selection value and a validation value.

Key Selection Value

The key selection value specifies the database column that contains the specific records that you want. When you mark a validation association as a key selection value, a database record that matches an associated column must exist or the validation fails. When AutoPilot runs the validation, it uses the key selection value to verify whether the system has updated the correct database column with the values that you stored during association.

When you choose a key selection value, you select a value that all records that you are validating have in common. For example, to validate data for the Address Book Revision form, you might choose the database column A5AN8 or ABAN8 as a key selection value because all address book records have an address book number.

Validation Value

When you mark a validation association value, you choose a specific record set to be validated. However, while the key selection value indicates the database where the records that are to be validated exist in the database, the validation value specifies the values, such as names, that you expect to find in the database.

Validation Execution

You run the validation using the record selection criteria that you established in validation association. AutoPilot retrieves the specified data from the database through structured query language (SQL). You can then compare the retrieved data with the data that you expected to return. Running the validation indicates whether the data that you entered and stored during association actually updated the database in the condition that you specified.

The following actions occur when you script running the declared and associated validation:

- An SQL statement is generated
- The system queries the database for the specified data
- The system compares the returned data to the expected data

The generated SQL statement contains the table that you chose when you declared the variable, the validation value columns that you chose during association, and the key selection that you chose during association.

Thus an SQL statement that you generate when you run the validation might contain the identity of the table (F0101) that contains the expected data, the columns (ABALPH, ABAT1 and ABAN8) that contain the expected data, and the key column (ABAN8) that joins all of the columns.

The SQL statement queries the database and retrieves data that conforms to the statement elements. You compare the results of that query against the results that you expected, based on the data that you stored during validation association.

If an error exists in associating the data to be validated, AutoPilot displays an error message. For example, if you do not choose a key column during association, AutoPilot notifies you that no record selection criteria have been chosen. If you associate the key column with an incorrect header control, grid column, or variable, AutoPilot indicates that the SQL statement contains an error.

In either of these cases, you can make any necessary corrections to the script. In addition, running a validation that fails does not stop the script from playing through to completion. The test results compare the data that you expected to return with the data that is actually returned.

Expect No Matching Records Option

If you enter a record, successfully validate it, and then write a command to delete the record, you might want to validate that the system successfully deleted the record from the database. To do so, you run the validation again and turn on the Expect No Matching Records option. When you run the script, AutoPilot again checks the database. The validation runs successfully if AutoPilot indicates that the record that you deleted from the database no longer exists. When you choose this option, you tell AutoPilot to expect to find no records that match the criteria in the SQL statement.

Scripting the Database Validation Command

You enter database validation action commands as you write your script. No formula exists to determine where the commands must occur. However, each of the three phases—declaration, assignment, and execution—must occur for the validation to take place.

Declaring a Validation

You can declare one or more validations as soon as you begin a script. You do not have to place the validation declaration command line at the top of the script if you want the validation to be effective within any node in the script, as you do when you declare a variable. However, declaring the validation early enables you to easily store data through association as you write the script.

▶ To declare a validation

1. From the Command menu, choose Declare New Validation.
2. In the command pane, enter a name in the Validation Name list.
3. Choose a selection in the Database Table list.

   This selection identifies the database table against which you validate data from AutoPilot.

4. Click Insert.

Associating a Validation

After you declare the validation, you have a place in which you can store values. During validation association, you choose values that you want to validate and pair, or associate, those values with values in the database.

You can write scripts that test scenarios that involve multi-currency with both accounting methods Y and Z. For example, you can use divisors instead of multipliers for exchange rate calculations.

The two types of validation associations are Key Selection Value and Validation Value:

Key Selection Value	Associations that determine which database record (row) you verify
Validation Value	Associations that specify columns within that row whose values you wish to verify

Each validation must include at least one Key Selection Value association and at least one Validation Value association.

► To associate a validation

1. From the Command menu, choose Associate a Validation Column.
2. Choose a validation name that you created using the Declare New Validation option by scrolling through the list and clicking the name you want.
3. Choose a column name from the Database Column list.
4. In the Value Type group box, choose Validation Value.

   This option associates the chosen column with a particular value.

5. In the Currency Type group box, choose one of the following currency options (optional):

   - Domestic
   - Foreign

   Note Concerning Currency Validations
   If you have not declared a currency validation, the system hides and disables these options. You must declare a currency validation to use these options.

6. Click a selection from the Source of Expected Data list.

   The source can be a literal value, variable, header control, or grid column.

7. In the value selection list, enter a literal value or choose the name of a variable, header control or grid column.

   This step specifies the value that you associate with the database column.

   Note
   With playback turned on, if you choose a header control or grid column, AutoPilot highlights (with a BlueCue) the designated control or column in the form. Be sure to choose a header control or grid column to which you have previously entered a value. Likewise, if you have chosen the name of a variable, be sure that you have followed the steps outlined previously for setting the variable's value.

8. Click the Insert button.
9. Before you finish the association, make sure you write a command that follows steps 1-7, but choose a Key Selection Value from the Value Type options.

   Data from all the individual columns from which you expect to have data returned is now associated with a single key column.

Manipulating the AutoPilot Tool Bar

You might work frequently with the tool bar during an AutoPilot session because you use many of its buttons to write context and action commands. To make your work easier, you can also move the tool bar and change its size and shape.

For instance, if you want more vertical space for the command pane, you can move the tool bar from near the top horizontal edge of the AutoPilot form to either the right or left vertical edge. You can also float the tool bar, moving it entirely out of the AutoPilot form and onto the desktop. Finally, after you have moved the tool bar from one position to another, you can return it to its original position by double-clicking the bar.

Executing a Validation

After you have declared the validation and associated the data with database tables and columns, you are ready to execute the validation.

► To execute a validation

1. From the Command menu, choose Execute Validation.
2. From the drop-down menu in the Validation Name list, choose a validation.

   Note
   AutoPilot populates the SQL Statement list. The statement contains the data dictionary aliases of the tables and columns that you associated with the data that you entered, the name of the validation, and the key selection value.

3. Click the Insert button.

   Later in the script, you might delete the records. You might want to validate that these deleted records are no longer in the database.

4. Repeat steps 1-3.
5. Choose the Expect No Match Records checkbox option.
6. Click the Insert button.

   Note
   You can declare and set the value of a variable to test validation success. See *Validation Success* in the *AutoPilot Guide* for more information.

The Command Line

You might want to run another program or programs within an AutoPilot script. For example, you might prepare a PowerPoint or Excel presentation that you want to include within the script. After you run that presentation, you might decide to close the program and then return to AutoPilot to continue scripting inputs.

You can complete these tasks by choosing Command Line from the Command menu. Type the path to the program that you want to run, much as you use the Run function in Windows. AutoPilot opens the program and the document or presentation that you created.

You can also send a command line message to make screen captures of J.D. Edwards ERP software forms at designated points in the script playback process. You can make these captures in a particular language version, and you can store them in a directory and file that you create.

Scripting a Command Line Command

Scripting the command line command requires that you know the path to the program that you want to run and, for example, a document in the program that you want to open. You can also send a command line command that captures the current form. You specify a folder where AutoPilot stores the screen captures.

Sending a Command Line Command

To open a program type the path for the program in the Command Line field. On playback, AutoPilot reads the path and opens the program.

► To send a command line message

1. From the Command menu, choose Command Line.

   The command pane displays the unpopulated Command Line list and radio button options.

2. Choose the Command Line option.
3. Complete the following field:

   - Command Line

   Type the path to the program that you want to open.

   Note
   Turn playback off if you do not want the program to open while you are writing the script.

4. Click Insert.

Capturing a Current Form

You can set up in advance the path and file extension for any screen captures that you make using the Capture current OneWorld window command. You do so by clicking Tools, choosing Options from the drop-down menu, and then clicking the Directories tab on the Options form.

In the Screen Capture field on the Directories tab, you type the path where AutoPilot will store your form captures. You then choose the format in which you want to save the images. You can click the scroll button to locate the format that you desire.

After you make these designations and you want to use the Command Line command to capture a form, type a name for the image in the File Name field, and AutoPilot captures the current form and saves it in the specified format in the specified directory.

► To capture a current form

1. From the Tools menu, choose Options.
2. On the Options form, click the Directories tab.
3. Complete the following field:
   - Screen Capture

     Type the path where you want AutoPilot to store the screen captures.
   - Format

     Using the drop-down menu, choose the file extension, such as .tif, that you want to use for the screen captures.
4. Click OK.
5. With a script open, from the Command menu, choose Command Line.
6. Click the Capture current OneWorld window option.
7. With a J.D. Edwards window active, complete the following field:
   - File Name
8. Click Insert.

   AutoPilot stores the image of the window in the drive and directory that you specified in the Options form.

Working with the Script Pane

You can work in the script pane to delete, modify, and move the commands that you have created. Working with the script pane requires that you understand the structure of the script tree that you build as you insert commands. You must also learn how to work with the tree to change its structure.

Parent-child relationships make up the script tree. Every script begins with the Begin Script command, from which any number of commands descend. Subsequent context commands are the parents of action commands and sometimes of other context commands. These parent context commands and their children make up nodes in the script pane. AutoPilot indents any command that is the child of another command. You can change the sequence of commands and the relationship between commands by dragging and dropping. For example, you can make one context command the child of another. This means that any changes you make to the parent command affect the child command.

Whatever the modifications are that you make to the script, the important thing to remember is that these modifications change the way that the script is run. The changes that you make should be based on what you want to accomplish by running the script. For example, you might drag a declared variable command line to the top of the script to make it global because you need all the commands in the script to have access to the value that you set for the variable.

Understanding the Script Pane Structure

As you write AutoPilot scripts, you build a tree structure in the script pane that is based on parent-child relationships. If you understand the structure of scripts, you can more easily modify your scripts and customize them to your specifications.

Because scripting requires a context, context commands are the basis of each script that you write. Each context command that you write and insert in the script creates a node, which appears in the script pane with a plus/minus sign that you can use to expand or collapse the node.

Each command that you write, whether it is a context command or an action command, becomes a command line in the script. Action command lines are attached to context command nodes, are indented beneath the node, and are affected by any changes that you make to the node, such as changing the repeat count.

In some cases, a context command line that forms a node is also a child of another context command line that forms a node. In this case, AutoPilot indents the child node, which is affected by any changes that you make to the parent command.

Finally, the script pane contains an insertion cursor, which indicates the position of the next command that you write. You can change the position of the insertion cursor either by clicking a command line or by dragging the insertion cursor. You can drop the insertion cursor into the script tree as a child of a node, which means that the next command you write is a child, or you can make the next command a parent that is independent of changes to other nodes in the script.

Command Lines

Command lines illustrate the choices that you make in the command pane to create context or action commands. The command lines in the script pane express either the context in which you create your script or the actions that you take within the context. For example, a header in a form is a context; the action you take within that context might be typing data in a specified control.

Context commands direct AutoPilot to applications, UBEs, processing options, interconnected applications, forms, header controls, grid columns, and QBE lines. They therefore express the environment in which actions, such as typing data and pressing buttons, are carried out. Context commands form the trunk of the script tree.

The context command that initializes a series of action commands and other context commands forms a node. The expand/collapse button in the script pane identifies the node. The expand/collapse button that represents the node appears in the script pane as a plus or minus sign inside a box next to the context command line.

Context commands perform the following functions:

- Form nodes that can be identified in the script with the node symbol or button, which appears as a plus or minus sign.
- Form nodes that can be expanded or collapsed by clicking the expand/collapse button.
- Form nodes that function as the parents of action commands and, sometimes, other context commands. These children are indented beneath the parent context command in the script pane.
- Can form discrete command line units. If two nodes are parallel to one another, commands that you add to one node do not affect the node that is parallel to it.
- Can initiate a sequence of other commands. The sequence can consist of the action commands and other context commands that are children of the parent command.
- May be played back multiple times if you change the repeat count of a node. Any commands that are attached to a context command in the script pane are played back as many times as you specify in the repeat count.

You script action commands to specify actions to be taken after you have scripted context commands.

In contrast to context commands, action commands have the following characteristics:

- Must be attached to, or be the children of, a context command
- Cannot have children attached to them
- Cannot be assigned repeat counts
- Are always indented beneath context commands in the script pane, which indicates that they are subordinate to context commands in the command hierarchy

Insertion Cursor

The insertion cursor, identified in the script pane as a red arrow, points to the position in the script at which you can insert a new command. If you insert commands sequentially without adjusting the script, the insertion cursor appears at the end of the script each time that you insert a command.

However, you can move the insertion cursor from one point in the script to another by clicking a command line. This moves the insertion cursor to the position directly below an action command line. If you then create and insert a new command, it appears at the point of the insertion cursor.

If you click a context command line, the insertion cursor appears at the end of the selected branch. If you leave the insertion cursor in this position, a new context command that you write creates a node, indicated by a minus or plus sign, that is parallel to the node on which you clicked. New commands that you write are attached to this node.

Expand/Collapse Button

The button that identifies the node also allows you to expand or collapse it. The expanded node reveals all command lines that are attached to the node. The collapsed node reveals only the context command that you scripted to initiate the node. When you expand a node, the button displays a minus sign. When you collapse a node, the button displays a plus sign.

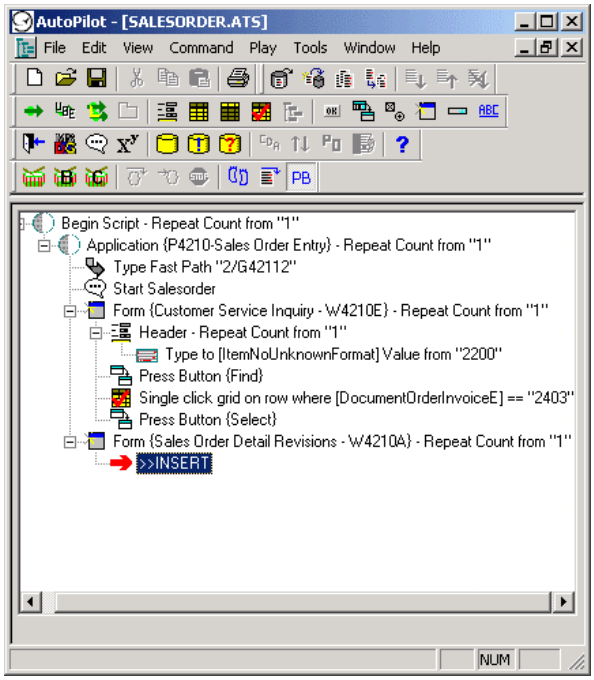
Parallel Nodes

When you change a node that is parallel to another node, the parallel node is unaffected. For example, if you write an Application command and write two Form commands in the same application, the Form commands are represented by nodes that are parallel to one another in the script pane. Any change that you make to one does not affect the other.

Indented Nodes

A node that is indented beneath another node in the script pane is affected by any change that you make to the parent. The indention of nodes reveals the hierarchy of context commands. For example, if you want to work with a particular form, you must first choose an application. Therefore, AutoPilot inserts and indents the Form command line below the Application command line.

Similarly, if you decide to add a form by pressing a button, AutoPilot inserts and indents the action command Press Toolbar Button below the Form command line.



The hierarchy of nodes expresses the logic that you follow to build AutoPilot scripts. For example, you might write the following sequence of commands to enter data to the header of a form and update the database by clicking OK:

- Application
- Form
- Header
- Type to
- Press Toolbar Button {OK}

Because you must launch an application before you can write any of the subsequent commands, the Application command forms a parent node in the script pane. Likewise, you must launch a form before you can enter data in a header control, so the Form command is a parent to the Header command, which is indented. Finally, the header is the context for performing the action of entering data to a control, so the action command Type to is indented beneath the Header command line.

Any context or action command that you insert to the script as a child of another command is affected by changes you make to the parent. For example, if you change the repeat count in an Application command line to 3, during playback the application launches three times, and any action commands that you write, such as clicking a toolbar button in the form, are performed three times.

Drag and Drop

You change the sequence of commands and the structure of the script by using the mouse to drag and drop commands. The following rules govern how you can use the drag-and-drop capability in the script pane:

- An action command within one context command node cannot be dragged into another context command node. For example, when a Type to command is attached to a Form command node, you cannot drag it to another Form command node.
- A context command node that is attached to one Application node cannot be dragged into another Application node.
- A context command node that you drag onto another context command node and insert as a child includes all commands that are attached to it.
- A context command node that you insert as a child is included in the playback of the parent context command node.
- The repeat count of the parent context command node applies to that node and to any other nodes that are attached to it as children.

You can make one parallel node a child of the other if you drag and drop it into the node that was formerly its sibling. AutoPilot indicates the parent-child relationship by indenting one node beneath the other.

Before AutoPilot creates the parent-child relationship, it displays the dialog box that asks you to confirm that you want to insert one node as a child of another.

Repeat Count

Every context command that creates a node in the script pane contains a repeat count. The repeat count specifies the number of times that AutoPilot plays the node and all of the commands attached to it. You can change the repeat count by selecting the node, entering a new repeat count in the command pane, and clicking the Update button.

See Also

- *Updating the Repeat Count in a Node* in the *AutoPilot Guide*

Modifying Scripts

You can customize scripts to your precise specifications in order to effectively test applications. You can also alter the structure of the script tree either stylistically, by expanding or collapsing nodes, or substantively, by adding, deleting, editing, or dragging commands.

You can change the order of the commands and, therefore, the structure of the scripts that you create, either as you are scripting or after you have completed scripting a series of commands. You can use your mouse, your keyboard, and the AutoPilot command pane to add, delete, or edit commands. You can also modify the structure of the script tree by using the mouse to move the insertion cursor and command lines.

Expanding and Collapsing a Node

When you expand all the nodes in a script, you can view the script in its entirety, with all command lines exposed. However, as your scripts get longer, you might want to see only a portion of the scripted commands. In that case, you can collapse nodes so that only the context commands that originated them are visible in the script pane. You can collapse or expand the entire script or certain portions of the script by clicking the node buttons, which are identified in the script pane by plus and minus signs.

Clicking a node button with a plus sign expands the node while clicking a node button with a minus sign collapses the node.

You can choose any point at which to collapse the tree. Choosing parent and child nodes to collapse provides a further illustration of the script tree structure. For example, when you click the Expand/Collapse button on a parent node, any nodes that you have inserted in it as children also collapse. When you click a child node, only that node collapses.

► To collapse the script tree

1. In the script pane of the AutoPilot form, expand any or all the nodes by clicking any node button that displays a plus sign.

   Expanded nodes show a button with a minus sign.

2. Click any node that shows a minus sign.

   The node that you clicked collapses and displays only the context command line.

Note

You can also collapse all nodes in a branch by right-clicking on the first node of the branch and choosing Collapse All. You can collapse all nodes in the script by right-clicking on the Begin Script line and choosing Collapse All.

► To expand the script tree

1. In the script pane of the AutoPilot form, collapse any or all the nodes by clicking any node button that displays a minus sign.

   Collapsed nodes display a button with a plus sign.

2. Click any node that displays a plus sign.

   The node that you clicked expands and shows the context command line and any commands attached to it.

Note

You can also expand all nodes in a branch by right-clicking on the first node of the branch and choosing Expand All. You can expand all nodes in the script by right-clicking on the Begin Script line and choosing Expand All.

Adding Command Lines

You might decide that you want to insert a command in your script after you have passed the point at which you want to insert it. For example, you might decide that you want to script an input to a header control after you have scripted the move to another form. You can accomplish this by placing the insertion cursor at the point in the script in which you want to insert a new command.

You can use the insertion cursor to insert a new command in the script only if the buttons in the tool bar are active. If they are not active, click and hold down the mouse button, drag the insertion cursor on top of a command line and release the mouse button. If the system prompts you to insert the cursor as a child, click Yes.

Nodes

A node consists of a parent context command and any related context commands and action commands that you attach to it. After you create a node by scripting a context command, you can write action commands and sometimes other context commands to develop the node and the script.

► To add a command line to an existing script

1. In the script pane of the AutoPilot form, click the insertion cursor.
2. Click and hold down the mouse button.
3. Drag the insertion cursor to the point in the script in which you want to insert a new command.

   As you drag the cursor, an indicator arrow appears. The arrow, which points up or down, indicates whether you place the cursor above or below a highlighted command line.

4. When you reach the point in which you want to insert the new command, release the mouse button.
5. Follow the steps required to insert the desired command in the script.

Note
If you cannot insert a command at a given point in the script, AutoPilot displays a disallow signal as you drag the insertion cursor.

Deleting Command Lines

You can delete lines from a script. If you delete a parent command line, you also delete all of its children.

► To delete a command line

1. In the script pane of the AutoPilot form, select a command by clicking it in the script.
2. Right-click the mouse.
3. Click Delete.

Moving Command Lines

You can further modify an existing script by moving command lines. You can move action commands that are attached to a context command to change, for example, the order in which the action commands play back.

You can change the order of action commands within a context command node by using the drag-and-drop capability in AutoPilot. During playback, AutoPilot replays these commands within the node in the new order. However, you can also change the structure of a script by moving a context command and inserting it as a child of another context command.

► To change the sequence of action commands

1. In the script pane in the AutoPilot form, select an action command line by clicking it.
2. Click the mouse.
3. Holding down the mouse button, drag the highlighted command line until it is on top of another command that you choose.

   The indicator arrow that appears as you are dragging the command line indicates whether the command appears above or below the targeted command line in the script pane.

4. Release the mouse.

► To change the sequence of context commands

1. In the script pane of the AutoPilot form, select a node by clicking it.
2. Click the mouse, then hold the button down and drag the mouse.

   A disallow symbol prevents you from dropping the node in an improper spot in the script. If the node can be dropped, the disallow symbol disappears.

3. When the target node is highlighted and the indicator arrow is pointing downward, release the mouse.
4. In the dialog box, click Yes or No when the system prompts you to insert as a child.

   AutoPilot inserts the dragged node in an indented position, or as a child to the target node if you answer Yes. If you answer No, the nodes are parallel.

Editing Command Lines

You can also make substantive changes to the content of the script by editing the command lines in the script pane. You make these changes by highlighting a command line in the script pane, and then working in the command pane to make new choices from lists to update the content of the command.

Note

Press Toolbar Button command lines cannot be edited. These must be deleted or added to the script as desired.

Editing an Action Command Line

While you can change only the repeat counts of context command lines, you can change the content of many action commands by clicking the command line and then choosing options in the command pane.

► To edit an action command line

1. In the script pane of the AutoPilot form, click an action command line.

   The command pane displays lists from which you can make choices to update the content of the command line.

   Note
   AutoPilot highlights the original choices that you made in the command pane.

2. In the command pane lists, click any new choices that you want to make.
3. Click Update.

   AutoPilot changes the command line in the script pane to reflect the changes that you made.

Editing a Context Command Line

The substance of the context command itself cannot be changed unless you delete it and insert a new one. However, you can change the number of times that AutoPilot loops through the node during script playback.

► To edit a context command line

1. In the script pane of the AutoPilot form, click the context command line.
2. In the command pane, choose a value from the Define Repeat Count list, such as literal or variable.
3. In the Repeat Count list, type the number of times that you want AutoPilot to loop through the node during playback.
4. Click Update.

Script Retention and Reuse

AutoPilot allows you to do more than write scripts on a one-time basis. AutoPilot permits you to save, modify, reuse, combine, and send scripts. These capabilities broaden the scope of and audience for your tests. AutoPilot helps you accomplish the following goals, which are integral to building a system of scripts:

- Saving scripts, which you can reuse or modify
- Including scripts with other scripts to broaden the scope of testing
- Passing variables between scripts in a master script that is composed of a parent script and one or more children
- Sharing scripts

You can save scripts either on your local drive or in the AutoPilot script repository. When you build scripts by including one or more scripts with another, you can retrieve the scripts either from the local drive or from the repository. Scripts that you create by including scripts can pass variable values; you accomplish this by declaring variables as external and forging links

between variables in separate scripts. Finally, you can e-mail any script that you create to colleagues.

Script Saving

You can save scripts as you work by clicking the File option in the menu bar. You assign a name to the script, which is saved in the directory you specify on the Directories tab of the Options form. Give the file a name that relates to the application that you are testing. If you continue to work on the script, you can save as you work.

If your computer freezes, or if J.D. Edwards ERP or AutoPilot fails while you are in an AutoPilot session, AutoPilot saves any scripts that are open. You can use the Configure tab on the Options form to set the conditions under which AutoPilot auto-saves as you work.

See Also

- *Options for Configuring AutoPilot* in the *AutoPilot Guide*

Script Includes

You can expand your testing scope by including one or more scripts on your local drive or in the script repository with a parent script. Do so by choosing Include Local Script or Include Reposited Script from the Tools menu. AutoPilot creates a copy of the script that you want to include and inserts it at the point in the open script that you have placed the insertion cursor. The included script becomes a child in a master script.

Whether you include a script you created on your local drive or a script from the repository, AutoPilot displays a form that includes all of the scripts that you have stored locally or to a form from which you can select one or more scripts from the repository. You choose the scripts to include, and AutoPilot inserts them as children of the master script. An Include command line contains the path to the included script--for example, `[C:\atg\ats\UBE blind app.ats]`.

You sometimes must edit a script before you include it with another. For example, if you scripted data input in one script, and that data is also included in another script, you must delete the data from the included script before you write an Include command. If you do not, when AutoPilot plays back the included script, it twice loads data into a form, which results in an error. You can open a script that has one or more scripts included in it and edit any of the included scripts. AutoPilot reloads the original script with the changes that you made to the included script.

Each time that you write a command to include a script, AutoPilot displays a dialog box that asks whether you want to continue script playback on include branch error. If you answer yes at this prompt, you ensure that, if an error occurs during the playback of the included script, AutoPilot reports the error but continues with playback of any other included scripts that exist. This feature is particularly useful if you are running very long scripts or batches of scripts.

See Also

- *Understanding the Script Repository* in the *AutoPilot Guide*

Variable Linking between Scripts

When you include scripts with a parent script, you might also want to share variable values between the scripts. AutoPilot provides a mechanism to accomplish this sharing called variable linking. In linking variables, you declare a variable in a parent script. In writing a script that you intend to include in the parent script, you also declare a variable, but if you want to link the variable, you declare it as external. AutoPilot allows you to link the externally declared variable to any variable that you declared in the parent script. The link means that you can pass between scripts the value that you set for the variable.

To increase the versatility of your scripting, AutoPilot allows you to designate a default value for any variable that you designate as external. Doing so allows you to run an included script in stand-alone mode. AutoPilot uses the default value wherever the value is needed in the script.

You can link variables between locally-generated scripts and reposited scripts, or you can link variables between a local script and a reposited script. If the included script contains a variable that you have declared as external, AutoPilot prompts you to identify the variable to which you want to establish a link in the parent script.

Default Values for External Variables

You can declare a variable as external and assign a default value to it. You give a variable a default value so that you can run a script in stand-alone mode. For example, you might write a script that tests one set of functions by itself. You might then include this script with one or more others. You can pass values between variables in the scripts by declaring a variable as external and linking it to a declared variable in the parent script. However, you might also want to have the ability to play back the original script in stand-alone mode. If you assign a default value to the variable and run the script in stand-alone mode, AutoPilot uses the default value to run the script. If you leave the default value of the external variable blank, AutoPilot reads the value as a null string.

When you create a script with an included script and linked variables, AutoPilot can pass a default value in the parent script to linked variables in the child script. However, if you assign a default value to an external variable in the child script, AutoPilot does not pass this value to the linked variable in the parent script. In this case, AutoPilot either overrides the default value with a value that you set for the variable in the parent script, or it passes the value as a null string if you do not set a variable value in the parent script.

The following table summarizes three scenarios and the results that occur when you write scripts with default variable values:

External Variable Default Value Set In:	Variable Link To:	Result
Parent script	External variable in child (included) script	AutoPilot passes the default value to linked variables in any child (included) scripts.
Child script	Variable in parent script	AutoPilot overrides the default value during playback, either with the value that is set for the variable in the parent script or with a null string if no value is set for the variable in the parent script.
Stand-alone script	Not applicable	The variable with the default value behaves as a local variable. AutoPilot uses the default value wherever the script indicates.

External Variables for Script Linking

You use external variables to help pass values between scripts. An external variable can receive a value from a variable in another script to which it is linked, or it can pass a value to a variable in another script. An option in the command pane allows you to designate a variable as external.

For example, you create Script A and declare a variable X. You then create Script B, declare a variable X, and designate it as external. Next, you include B with A. Script A is the parent script, and Script B is the child. AutoPilot asks that you link the externally declared variable to a variable in Script A. You link variable X to variable X.

So far, you have provided the mechanism for passing a variable value from one script to another. However, suppose that, in Script A, you set the value of variable X to an address number, such as 4245. With the link established, AutoPilot can now pass this value to Script B when you run the two scripts together.

Variable Links

When you include with your parent script other scripts that contain external variables, you must define the links to each external variable so that AutoPilot can pass values between scripts. Defining the links allows AutoPilot to store data in a declared variable in the parent script or retrieve data from the parent script and reuse it in included scripts that contain external variables.

Continue Script Playback on Include Branch Error

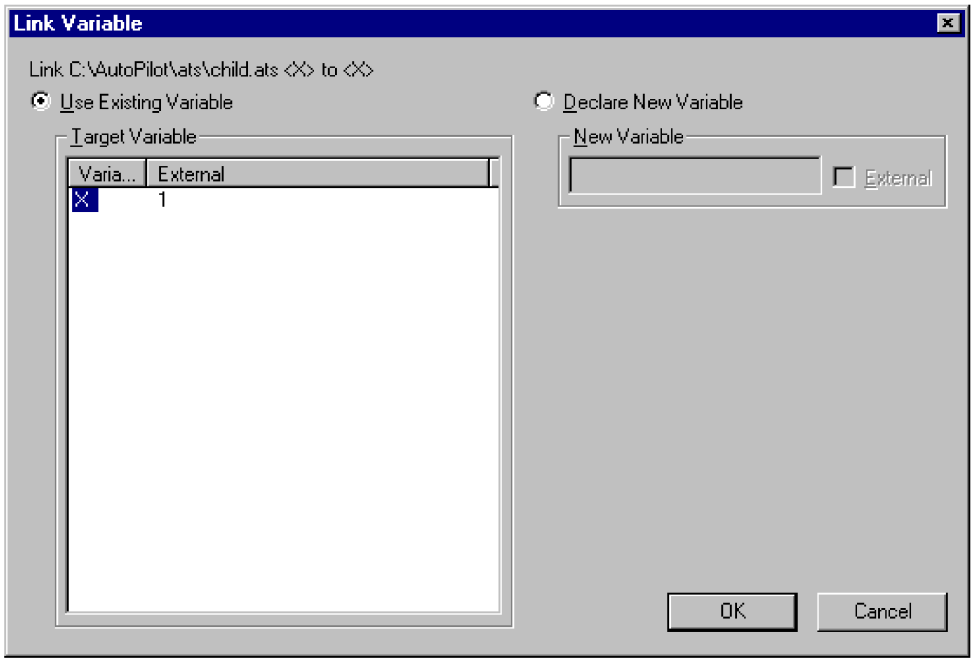
Before you run an Include command, AutoPilot displays a message box that prompts you to continue the script playback on include branch error.

Continuing the script on include branch error means that, if AutoPilot encounters an error in an included script during playback, playback moves on to the next included script rather than failing the entire playback session. Choosing Yes is recommended if you intend to test a long series of included scripts.

After you respond to the prompt, AutoPilot displays another message that prompts you to link any unlinked external variables.

Link Variable Form

You use the Link Variable form to establish the link between the variable that you declared as external in your included script and a variable that you declared in the parent script.



Note

The Link Variable form contains a link path that identifies the name of the included script and its externally declared variable, along with the name of the variable in the parent script. The path `Link C:\AutoPilot\ats\child.ats <x> to <x>` indicates that the included script is child.ats and that its externally declared variable `<x>` will be linked to the variable `<x>` in the parent script. If you want to change the link path, you click the name of a different variable from the parent script.

When you select a variable, AutoPilot establishes the link and inserts a Link script object and command line in the script pane. AutoPilot inserts and maintains the link relationship in the parent script.

If you click Cancel on the Link Variable form and then attempt to save the script, AutoPilot continues to prompt you to supply the variable link. You can save the script without establishing the link; however, when you open it, the system displays the Variable Link form again.

Recursive Value Searching

AutoPilot searches recursively for links, meaning that the search continues until it meets one of the following conditions:

- The linked variable in the parent script is not external.
- The linked variable is in the master parent script, meaning that AutoPilot has searched until it reached the top of the script tree.

AutoPilot can repeatedly search for a value; therefore, you can establish links between variables in multiple script parent-child relationships. When you play back a script that has external variables, AutoPilot searches the parent script for the link to the external variable. When AutoPilot finds the link, the Link script object determines whether the Declare command for the variable is external. If the declared variable is not external, AutoPilot stops its search. If the declared variable is external, AutoPilot continues to search for links.

For example, suppose that you create four separate scripts: A, B, C, and D. Each script contains variables that you declare as external. You set the value of the variable in Script D, and then include Scripts C and D with B. When you insert the Include command, AutoPilot prompts you to create the links between the parent script (B) and the included scripts (C and D). When you run the script, AutoPilot searches for the link in the parent script. If you did not declare the linked variable in the parent script as external, the recursive process ceases.

Suppose, however, that you decide to include Script B with Script A. Script A now becomes the master parent script. You now declare the variable in Script B as external and include it with Script A. When you run the script, AutoPilot searches for the link in the parent script. When AutoPilot reaches Script B, it reads the declared variable. Since that variable is external, it continues to search the tree until it reaches the master parent script, Script A.

You can create as many parent-child relationships between scripts as you need. Because it continues to search for values until it finds none, AutoPilot can maintain as many links as you need.

The master parent script, meaning the script at the top of the tree, should not contain variables that you declare as external. These are unresolved variables, meaning that they have been declared as external, yet they have not been linked. Declaring these variables as external causes your script playback to fail. However, good scripting technique dictates that any external variable should always link to another variable in a parent script.

See Also

- *Indented Nodes* in the *AutoPilot Guide*

Script Sharing

After you create a script and modify it to your specifications, you can e-mail it to another person who might be interested in using it or including it with another script. However, only people who have installed AutoPilot on their computers can run the script.

Broken Links

If you remove an external attribute from a variable, you break its link to the link object, which can no longer find its reference. When you see a link object in a parent script, you should verify that a variable in the child script is designated as external in the script pane. If that designation is not there, you have a broken link.

AutoPilot notifies you that you have broken the link when you attempt to load the script containing the parent and the included scripts. The Research Broken Links form displays the broken links.

Note

The Research Broken Links form contains the name of the broken link path and a description of the link error. If you click the broken link, AutoPilot highlights the corresponding Link command line in the script pane.

You might break the links by inadvertently deleting the linked variable in the parent script or by deleting one of the included scripts before you remove the links in the parent script. In either of these cases, you must repair the broken links.

Reusing Scripts

After you create and modify scripts in AutoPilot, you can save them and then reuse them as you need them. You can include scripts that you have created in other scripts that you are working on, and you can e-mail saved scripts to other scripters and testers, if the person with whom you are corresponding has AutoPilot installed.

Saving and reusing scripts allows you to reduce the workload of colleagues who are testing the same applications that you are testing. You accomplish these tasks using the AutoPilot menu bar and the file directory on your computer.

AutoPilot also allows you to declare variables as external. This means that you can link the variable that you declare as external to a variable in another script and pass the value of the variable between scripts. Moreover, you can change the value of the variable to which you link the external variable. The ability to pass values between scripts and to change those values increases the versatility of your scripts, making them reusable and dynamic entities.

Including Scripts

Including scripts allows you to broaden your testing scope. You can include one or more scripts with another script, either from your local drive or from the script repository. Each time that you include scripts with another script, you create a master script. The script that contains the included scripts is the parent, and the included scripts are its children. You can edit included scripts within the master, but any changes that you make to an included script affect the master.

► To include one local script with another

1. In the script pane, place the insertion cursor at the point at which you want AutoPilot to insert the included script.
2. From the Tools menu, choose Include Local Script.
3. In the Select files to include form, click the name of the script that you want to include.

   If you want to include more than one script, hold down either the Control key or the Shift Key and click another script. When you click the name of a script, its name appears in the File Name list.

4. Click Open.
5. In the Continue script playback on include branch error form, click Yes or No.

AutoPilot inserts the script that you chose in the script pane of the open script at the point of the insertion cursor. AutoPilot inserts a Continue Playback or Fail Playback message on the Include command line.

You can change the message from Continue Playback to Fail Playback by clicking the command line and unchecking the Continue playback on error option in the command pane.

► To include a reposited script with another script

1. In the script pane, position the insertion cursor at the point at which you want to include a script.
2. From the Tools menu, choose Include Reposited Script.
3. On Select Script, complete any of the fields on any of the tabs to help narrow the search for your script, and then click OK.
4. On Include Repository Script, click the name of one or more script titles, and then click Include.
5. On the Continue script playback on include branch error form, click Yes or No.

   AutoPilot places the included script in the script pane of the open script at the point of the insertion cursor.

► To edit an included script

1. In the script pane of the AutoPilot form, select the Include command line of the script that you want to edit.
2. Right-click and choose Edit for a local script, or Check Out & Edit for a reposited script.

   Caution
   AutoPilot opens the included script. The included script might be a parent of other scripts. To edit a child of the script that you are editing, you must select it and choose Edit again.

3. Perform the necessary edits to the included script and from the File menu, choose Close.
4. Save the changes to the included script.

   If you edited a reposited script, AutoPilot prompts you to reload the script.

5. To load the changes to the included script, click Yes.

   AutoPilot reloads the parent script with the changes to the included script.

6. Save the changes to the master script.

Creating Variable Links

When you create variable links, you write the commands that allow AutoPilot to pass variable values between two or more scripts. You declare a variable and set its value just as you do in a stand-alone script. However, by declaring the variable as external, you indicate that it can be linked to a variable in another script. AutoPilot uses links to pass a value that you set in one variable in one script to another script.

The process of forging variable links requires that you write two or more scripts, declare certain variables as external, set a variable value, and then create the links between the variables. You can link variables between scripts that you maintain locally or between reposited scripts. Finally, if you break links between variables, you must recreate the links before your script will run.

► To declare a variable as external

1. From the Command menu, choose Variables.
2. Type the name of a new variable in the unpopulated New Variable list.
3. Click the External option.
4. If you are declaring the variable but not setting a value, choose Unknown/None from the Source of Value list and click the Insert button.
5. If you are setting a value for the variable, choose from the Source of Value list and the value selection list.
6. Click the Insert button.

Note
AutoPilot includes the word External in the Declare command line in the script pane.

See Also

- *Using a Variable as a Source of Input* in the *AutoPilot Guide*
- *Using the Source of Input List* in the *AutoPilot Guide*

Declaring a Variable as External

You declare a variable as external only if you want to link it to a variable in another script to pass a value between the scripts. You declare an external variable just as you declare a variable in a stand-alone script. In AutoPilot, you can choose an option that declares the variable as external.

Note

You can also write a script with a local variable and then, later, update the variable and make it external. You might do this when you want to include the script with another script.

Assigning a Default Value to an External Variable

You can set a default value for a variable, regardless of whether you choose to declare it as external. By assigning a default value, you ensure that you can run your script in stand-alone mode, even if you make the variable external and link it to a variable in another script. If you link the variable with the default value to a variable in a parent script, AutoPilot overrides the default value during playback, either with the value of the variable in the parent script, or with a null string if the variable in the parent script has no value.

► To assign a default value to an external variable

1. From the Command menu, choose Variables.
2. In the AutoPilot command pane, type the name of a variable in the New Variable list.
3. Click the External option.
4. In the Default list, type a value for the variable.
5. Click the Insert button.

Linking Variables between Local Scripts

You can establish links between scripts that you generate locally. To do so, you write a parent script with one or more declared variables, and then write one or more scripts that you want to include with the parent. The scripts to be included must contain externally declared variables if you want to establish links. You then open the parent script and choose the scripts to be included. The link path in the parent script points to the scripts that you have included as children in your parent script.

► To link variables between local scripts

1. From the Tools menu, choose Include Local Script.
2. In the Select files to include form, click the name of the script that you want to include.

   If you want to include more than one script, hold down the Control key and click another script. When you click the name of a script, its name appears in the File Name list.

3. Click Open.
4. In the Continue script playback on include branch error dialog box, click Yes or No.
5. In the dialog box that prompts you to link unlinked external variables, click OK.

   Note
   The system prompts you to link unlinked variables only if you have declared variables as external in one or more of the scripts that you are including with a parent.

6. In the Link Variables form, choose the name of the declared variable in the parent script that you want to link to the external variable in the included script.
7. Click OK.
8. Repeat steps 6 and 7 as many times as necessary to establish links for all external variables.

Linking Variables between Reposited Scripts

AutoPilot also allows you to link variables between reposited scripts or between local and reposited scripts. The reposited script that you want to include with a parent script must again contain a variable that you declared as external. You open the parent script and choose the scripts to include from the Include Repository Script form. The link path in the parent script points to scripts from the repository that you have included as children, and it contains the name of the script, rather than the path that appears when you include a local script.

► To link variables between reposited scripts

1. From the Tools menu, choose Include Reposited Script.
2. On Select Script, complete as many fields as necessary to narrow your search ,and then click OK.
3. On the Include Repository Script form, click the name of one or more scripts.
4. Click Include.
5. In the Continue script playback on include branch error dialog box, click Yes or No.
6. In the dialog box that prompts you to link unlinked external variables, click OK.
7. In the Link Variables form, choose the name of the declared variable in the parent script that you want to link to the external variable in the included script.
8. Click OK.
9. Repeat steps 7 and 8 as many times as necessary to establish links for all external variables.

See Also

- *Including Scripts* in the *AutoPilot Guide*
- *Understanding the Script Repository* in the *AutoPilot Guide*

Researching and Repairing Broken Links

You create a variable link between an external variable and a declared variable in a parent script. After you create that link, you must maintain it if you want AutoPilot to pass variable values between scripts. If you break the link between an external variable and a declared variable in the parent script, AutoPilot prompts you to use the Research Broken Links form to research the broken links and restore them.

► To research and repair broken links

1. In the Research Broken Links form, note in the Broken Links and Link Error headings the path to the script that contains the broken links.
2. Under the same headings, note the variables that have broken links.
3. Click OK.
4. Open the script that contains the broken links.
5. In the script pane, find each variable that has a broken link.
6. Click the Declare command line.
7. In the command pane, choose the External option.

8. Click the Insert button.
9. After you declare as external each variable that has a broken link, save and close the script.

Understanding Script Playback Functions

AutoPilot allows you to handle certain playback tasks automatically and to run playback manually. You can set up automatic script playback features. For example, you can specify that AutoPilot automatically capture and save the results of each script playback session.

You manually control the playback using eight buttons on the tool bar and the Pause key on the keyboard. Each tool bar button represents a specific playback function. You can also find each option in the Play menu. The table below describes the manual playback functions.

Play or Command Menu Choice	Playback Function
Play from Top (Play menu)	Play back the entire script from the top.
Play from Cursor (Play menu)	Play back the script from a chosen cursor position to the end of the script.
Play Branch (Play menu)	Play back a selected branch of the script.
Stepping on (Play menu)	Play one line at a time, or until the next breakpoint (see Step Next and Continue to Breakpoint).
Step Next (Play menu)	With stepping on, play one line at a time.
Continue To Breakpoint (Play menu)	With stepping on, continue playback from a chosen spot in the script to a breakpoint.
Stop (Play menu)	Stop playback.
Ignore Breakpoints (Play menu)	Ignore breakpoints during playback.
Comment/Wait (Command menu)	Insert a wait period to occur during playback.
Comment/Wait (Command menu)	Insert a comment in the script.
Pause button on keyboard	Pause playback.

Playing Back the Script

You can play back the script at any point, regardless of whether you are finished writing commands.

AutoPilot offers the following options for playing back the script:

- From the beginning to the end without interruption
- From a chosen cursor position to the end without interruption
- From the beginning of a chosen script branch (node) to the end of the branch
- From a chosen cursor position line-by-line
- From a chosen position to a designated breakpoint

You can stop playback at any time by clicking Stop in the Play menu or the Stop button on the tool bar.

You can script wait periods during playback. When you script a wait period, the playback stops for a length of time that you specify, and then it resumes. You can also pause playback indefinitely by pressing the Pause key on your keyboard. Resume play by pressing Pause again.

You can configure playback by choosing Options from the Tools menu and clicking the Playback tab. For example, you can set AutoPilot to capture and display the results of a playback session. The results are presented as an event stream: a time-stamped, chronological record of each AutoPilot and J.D. Edwards ERP software event that occurred during the session.

When you play the script, AutoPilot stops the playback if an error occurs in J.D. Edwards ERP software. If you have configured playback to save and display test results, the message `Script playback was not successful` appears in the Test Results form. If AutoPilot encounters no errors during playback, it displays the message `Script playback completed successfully`.

Automatic Script Playback Configuration

At any time, you can set or change your script playback configuration to let AutoPilot play back certain features without your intervention. To configure script playback, you use the Options form, which you access from the Tools menu on the AutoPilot form.

Playback during Script Creation

When you turn on the Play Back while Creating Script option, AutoPilot plays each command after you insert it in the script. If you want to write your script without any delay caused by script playback, you should turn off this option. Alternatively, if you want to observe each command that you script, you turn on this option.

Storage and Display of Playback Data

AutoPilot allows you to store and display the results of each script playback. If you choose to save results data, AutoPilot saves script playback results as a binary large object in the AutoPilot Playback Results Detail Table (F97214). If you display test results after playback, AutoPilot automatically displays the Test Results form after playback. This form contains tabs that you use to review additional information about script playback.

If you choose to save script playback results, you can view a history of playback results by choosing Results from the Tools menu.

See Also

- *Understanding Script Reporting* in the *AutoPilot Guide*
- *Options for Configuring AutoPilot* in the *AutoPilot Guide*

Breakpoint Handling

A breakpoint is a point in the script that halts playback until you manually continue it or cancel it. If you want to play the script uninterrupted, but keep any breakpoints that you have inserted, you turn on the Ignore Breakpoints during Playback option.

Playback Speed

If you turn on the Accelerated Playback option, J.D. Edwards ERP software notifies AutoPilot as soon as processing is complete so that playback can immediately continue. In general, you should turn on this option only when you are running relatively simple scripts that do not require J.D. Edwards ERP software to perform a large amount of processing.

Cancel Playback on Comm Error

If you turn on the Cancel Playback on Comm Error option, and you experience software communication issues, AutoPilot cancels the current script. If you do not turn on the Cancel Playback on Comm Error option, AutoPilot tries to complete the script.

Log Variables on Script Failure

If you turn on the Log Variables on Script Failure option, AutoPilot logs the final values of all variables in the event of a script failure.

When a script fails, the system writes the current value of variable assignments to the AutoPilot results output. You use this output to analyze script failures. In the results database, you can view variable contents and draw immediate conclusions as to the cause of a script failure. For example, the journal date variable value of "06/03/02" would cause the script to fail because we are currently working in 2003 and not 2002. This is typical of the variable data that you could view. J.D. Edwards highly recommends that you turn this option on.

Event Stream

The term *event stream* refers to the flow of information from J.D. Edwards ERP software to AutoPilot that occurs during playback. You click one of the following options to configure script playback to capture this information:

- None
- J.D. Edwards ERP software warning and error messages
- Level 1 API calls
- All API call levels

See Also

- *Virtual AutoPilot Guide*
- *Analyzer Tool Guide*

Manual Script Playback Options

After you configure script playback, you run playback using the Command menu or the playback buttons on the tool bar. You can play the script from the top without interruption, play the script from any chosen spot to the end, play only a chosen branch of the script, play back the script one command at a time, play the script to a breakpoint, or stop playback. You can insert wait periods in the script to delay playback for a set period of time before it resumes, or you can manually pause and resume script playback.

Play from Top

When you click Play from Top in the Play menu, script playback begins with the first command line and continues until the end of the script unless AutoPilot encounters an error.

When you use this or any of the other playback functions, you can stop the playback by clicking the Stop button on the tool bar or by pressing the Pause button on your keyboard. Before you play back the entire script, remove any breakpoints that you inserted in the script.

Play from Cursor to End of Script

The Play from Cursor command allows you to choose any spot in the script and then play the script to completion, if AutoPilot does not encounter an error or a breakpoint.

Play a Chosen Branch of the Script

The Play Branch command lets you play only a single script node that consists of one or more context commands and a series of action commands.

Play from a Chosen Script Line Command

To manually control playback from a chosen point in the script, you choose the Stepping On command. You can play back either from the top, from a chosen cursor position, or from a chosen branch of the script. You can then choose how you want to play back the script, either one command line at a time or to a chosen breakpoint.

Play to the Next Script Line Command

After you click Stepping On and choose to play back either from the top, from a chosen cursor position, or from a chosen branch, you choose the Step Next command to play the script one command line at a time. Script playback does not proceed until you click the Step Next button on the tool bar, or choose Step Next from the Play menu.

Toggle a Breakpoint

A breakpoint is a command line that you choose to stop playback until you resume or cancel playback. You can insert as many breakpoints in the script as you need. Doing so gives you another way to isolate areas of the script and observe the playback.

You toggle the breakpoint by placing the insertion cursor after the line in the script where you want playback to break, right-clicking the mouse, and choosing Toggle Breakpoint. You can script multiple playback breakpoints. The breakpoint itself does not stop playback. You can do so only by clicking the Stop button.

Continue to Breakpoint

After you click Stepping On and choose a position from which to play the script, you can play back the script to a breakpoint. AutoPilot plays all commands until it reaches the breakpoint.

Wait Before Proceeding

You use the Comment/Wait command to script waiting periods, or pauses, in the playback. You can insert one or more wait commands anywhere in the script pane. After the prescribed wait period has elapsed, playback resumes without your intervention. You can specify the duration of the wait. You can insert waits in the script that are of sufficient duration to simulate the amount of time required to actually enter data in a header or detail area.

Enter comments in the Comments field. Enter the period to wait in the Time (mSec) field. Enter the time in milliseconds. For example, if you want a 30 second pause, enter 30000.

A Wait command produces a pause of determinate length. By contrast, when you press the Pause button on the keyboard, AutoPilot pauses playback until you click the Pause button again.

Script Comment

Using the Comment/Wait command, you can write brief comments that might, for example, explain the reason that you inserted a wait. If you exchange scripts with a colleague, you can use the comments to explain the actions that occurred at a particular point in the script or to explain what the script is designed to test. You type your comment in the Comment field in the command pane and insert it in the script.

Note Concerning Comment Length

AutoPilot truncates the comment in the script pane at 54 characters, including spaces.

AutoPilot also allows you to cut or copy comments from other scripts or from other documents and paste them into the Comment field of the command pane. For example, if a comment that you insert in one script is applicable to several other scripts, you copy that comment and paste it into other scripts.

When you choose Comment/Wait from the Command menu, the command pane also displays the following options:

- Wait until message window is closed
- Log To Test Manager
- Fail Script

Turn on the Wait until message window is closed option if your script presses the Delete button in a form. If you turn on the option, AutoPilot does not proceed with script playback until it has clicked OK on the Confirm Delete form.

Turn on the Log To Test Manager option if you plan to include a script as part of batch testing. You use the Test Manager tool to assemble script playlists for batch playback. If you turn on the Log To Test Manager option, the comments that you insert in a script are sent to Test Manager and included in a report after playback.

Turn on the Fail Script option if a critical event in your script caused the script to fail. If you turn on this option, AutoPilot automatically fails the script at the point in which you insert the command.

If you choose to fail the script, AutoPilot inserts a comment symbol in red.

You can use the Fail Script option in conjunction with logging comments to Test Manager by turning on both options. When you run a batch test, AutoPilot fails the script and generates a summary report in Test Manager that lists any comments that you include about the failure.

See Also

For more information about Test Manager, see the following topics:

- *Understanding AutoPilot Test Manager* in the *AutoPilot Guide*
- *Managing Script Testing* in the *AutoPilot Guide*

Ignore Breakpoints during Playback

You might want to preserve breakpoints that you have inserted in the script, but run the script one or more times without breakpoints. Rather than turn the breakpoints on and off, you can turn on the Ignore Breakpoints during Playback option. When you want to run the script to the breakpoint that you designated, turn off the Ignore Breakpoints during Playback option and play back the script.

Stop Playback

At any point during playback, you can stop the process by clicking Stop in the Play menu or the Stop button on the tool bar. If the script is playing, the Stop button turns red. When you click the Stop button, AutoPilot displays a Script Playback Cancelled message, and the Stop button returns to gray.

Running Script Playback

You run the various script playback functions in AutoPilot using the six choices in the Play menu or the six playback buttons on the tool bar. You can also use the Comment/Wait command to script one or more pauses in the playback and to insert comments in selected command lines in the script or as stand-alone lines. Finally, you can use the mouse to right-click a command line in the script pane and to turn a breakpoint on or off.

Pause Playback

You can use the Pause button on your keyboard to control script playback.

Pressing the Pause button during script playback pauses the playback. Press Pause again to resume playback.

AutoPilot must have control of the screen when you press the Pause button. To verify that AutoPilot has control of the screen, position the cursor anywhere in the AutoPilot form and click the mouse.

Playing the Script from the Top

Before you play the script from the top, close all open applications. If you do not close all applications, playback might not complete successfully. AutoPilot expects to control the mouse during the playback process. Do not move the mouse or attempt to open any applications or programs while playback is running.

► To play the entire script from the top

1. In the script pane of the AutoPilot form, verify that you have completed the following steps:
   a. Removed any breakpoints
   b. Turned off the Stepping On button

c. Closed all J.D. Edwards windows

2. From the Play menu, choose Play from Top.

Note
You can also start the playback process by pressing the F5 key on the keyboard.

Playing the Script from a Chosen Cursor Position

You use the Play From Cursor function when you want to begin playback from the position of the cursor, rather than from the top. Before you play back from cursor, ensure that the J.D. Edwards software is open to the appropriate place. For example, if you are starting playback on a certain form, ensure that the form is open.

► To play back the script from a chosen cursor position

1. In the script pane in the AutoPilot form, choose a command line from which you want to play back.
2. Click the command line to select it.
3. From the Play menu, choose Play From Cursor.

Playing a Branch of the Script

You can use the Play Branch feature to play a selected branch of a script. The scripts plays from the selected line to the Insert cursor.

► To play a branch of the script

1. In the script pane in the AutoPilot form, choose the first command line in a branch.
2. From the Play menu, choose Play Branch.

Playing the Script to the Next Command

After you turn on Stepping On, you choose a command line from which to play the script back, either from the top, from a branch of the script, or from another chosen cursor position. You can then play the script one line at a time or until the next breakpoint.

► To play the script back one line at a time

1. In the script pane in the AutoPilot form, choose a command line from which you want to play back.
2. From the Play menu, choose Stepping On.
3. From the Play menu, choose one of the following buttons:
   - Play from Top
   - Play Branch
   - Play from Cursor

AutoPilot enables the Step Next, Continue to Breakpoint, and Stop buttons.

4. To proceed to the next line, click Step Next.
5. To play to the next breakpoint in the script, click Continue to Breakpoint.
6. To discontinue the playback, click Stop.

Pausing Playback

Use the Pause button on the keyboard to pause script playback. Press Pause again to resume playback.

► To pause script playback after beginning playback from the top

1. From the Play menu choose Play from Top.
2. At the point from which you want to halt playback, press the Pause button on the keyboard.
3. To continue playback to the next command line, press the Pause button again.

Note
AutoPilot does not proceed to the next command line until you press the Pause button again.

4. To stop playback, click the Stop button on the tool bar.

► To pause script playback after beginning playback from a chosen command line

1. In the script pane in the AutoPilot form, choose a command line from which you want to play back.
2. Click the Play from Cursor To End Of Script button.
3. Click the Pause button on the keyboard.

   AutoPilot runs the selected command line and advances to the next command line.

4. Continue through the script one command line at a time, pressing the Pause button each time that you want to run a command.

Ignoring Breakpoints in the Script

You can ignore breakpoints that you turned on in the script if you want to play back the script without interruptions but do not want to turn off the breakpoints. To do so, choose Ignore Breakpoints from the Play menu. If you later want to stop playback at the breakpoints, choose the menu option again to turn it off.

Toggling a Breakpoint

If you want the script to play only to a predetermined command line in the script, you can toggle a breakpoint by highlighting it and then right-clicking the command line at which you want playback to break. You can toggle as many breakpoints as you like. For example, you might toggle a breakpoint when you have created a lengthy script and want to play back only a portion of it rather than the entire script.

When you toggle a breakpoint and then play back the script, the playback proceeds to the line on which you set the breakpoint, and then it halts until you either stop playback or continue it to another breakpoint. You can also highlight and right-click a command line to remove a breakpoint.

► To toggle a breakpoint

1. In the script pane of the AutoPilot form, choose a playback breakpoint by selecting a line in the script.
2. Right-click the mouse.
3. Click Toggle Breakpoint.

   AutoPilot inserts the breakpoint to the script.

4. To remove the breakpoint, click a command line on which you entered a breakpoint.
5. Right-click the mouse.
6. Click Toggle Breakpoint.

   AutoPilot removes the breakpoint that you inserted.

Playing the Script to a Breakpoint

After you have toggled a breakpoint, you can play back the script, either from the top or from a cursor position of your choice. When AutoPilot reaches the command line that contains the breakpoint, playback halts. However, AutoPilot does not cancel playback. If you want to continue scripting, or if you want to play back the script differently, you must click the Stop button to cancel playback.

► To play the script to a breakpoint

1. From the script pane in the AutoPilot form, toggle a breakpoint in the script.
2. From the Play menu, click Play from Top.
3. If the script plays to the breakpoint and you want to continue scripting, toggle off the breakpoint.
4. From the Play menu, click Stop.

Continuing Playback to a Breakpoint

After you toggle one or more breakpoints and turn on stepping, you can play the script back from breakpoint to breakpoint by clicking the Continue to Breakpoint button on the tool bar.

► To continue playback to a breakpoint

1. From the script pane in AutoPilot, choose a playback breakpoint by selecting a line in the script.
2. Right-click the mouse.
3. Click Toggle Breakpoint to insert the breakpoint in the script.
4. Choose a point in the script from which you want to play back.
5. From the Play menu, choose Stepping On.
6. Click the Play from Cursor To End Of Script button.
7. Click Continue to Breakpoint.

Note
You can set as many breakpoints in the script as you desire and click Continue to Breakpoint each time that playback reaches one.

► To ignore breakpoints during script playback

1. In the AutoPilot script pane, toggle on one or more breakpoints in the script.
2. Click the Ignore Breakpoints button in the AutoPilot tool bar.

Inserting a Wait Command in the Script

If you insert a breakpoint in the script, playback halts when AutoPilot reaches the breakpoint. Playback does not resume or stop without your intervention. Alternatively, if you click Comment/Wait in the Command menu, you can script a specified wait period, or pause, at a predetermined script command line. When the playback reaches this command line, the wait occurs, and then playback proceeds.

► To insert a wait command in the script

1. In the script pane in the AutoPilot form, select a command line in the script by clicking it.
2. From the Command menu, choose Comment/Wait.
3. Press the Tab key or place the cursor in the unpopulated Time (msec) list.
4. Type a time, in milliseconds, for the wait.

   Note
   Do not use commas when you type the time of the wait.

5. Click the Insert button.
6. Run a playback command.

Inserting a Comment in the Script

You can also use the Comment/Wait command to insert into the script pane comments about the command line that you chose or general comments about the script, including its purpose. If you want to include the script in the batch testing that Test Manager runs, and you want the comments to appear in a summary report after batch testing, choose the Log To Test Manager option.

Failing a Script

You can automatically fail a script by turning on the Fail Script option. This option appears in the command pane when you click the Comment/Wait button. Turn on the Fail Script option to include the script in a batch that Test Manager runs.

► To fail a script

1. In the script pane of the AutoPilot form, place the insertion cursor at the point that you want to fail the script.
2. From the Command menu, choose Comment/Wait.
3. In the command pane, choose the Fail script option and click the Insert button.

   AutoPilot inserts a red Comment symbol in the script pane to indicate that the script fails at that point.

Setting Transaction Times in the Script

A transaction is a series of events, bounded by a start and end point. You can also insert comments in the script to measure playback transaction time. You use the Comment/Wait command to assign a name to the transaction, to insert a starting point, such as launching an application, and to insert a finishing point, such as closing J.D. Edwards ERP software. Setting transaction times in the script provides important information about the time that J.D. Edwards ERP software requires to run a series of commands.

See Also

- *Virtual AutoPilot Guide*

► To set transaction times in the script

1. In the script pane of the AutoPilot form, determine the command line that represents the start of the transaction, and then place the insertion cursor directly above it.
2. From the Command menu, choose Comment/Wait.
3. In the unpopulated Comment list of the AutoPilot command pane, enter Start, a space, and then a name for the transaction.
4. Click the Insert button.

   AutoPilot inserts a command line that marks the start of the transaction.

5. Determine the command line that represents the end of the transaction, and then place the insertion cursor after it.
6. From the Command menu, choose Comment/Wait.
7. In the unpopulated Comment list of the AutoPilot command pane, type End, a space, and a name for the transaction.

   AutoPilot inserts a command line that marks the end of the transaction.

   Note
   The name that you assign to the end of the transaction must exactly match the name that you assign to the start of the transaction.

8. Click Insert.

► To insert a comment in the script

1. In the script pane of the AutoPilot form, place the insertion cursor at the point in the script in which you want the comment to appear.
2. From the Command menu, choose Comment/Wait.
3. In the unpopulated Comment list of the AutoPilot command pane, type a comment.
4. Choose the Log to Test Manager option if you want AutoPilot to include the comment in a summary report after testing.
5. Click Insert.

Creating a Sample AutoPilot Script

Although AutoPilot can be used to create a script to verify any application, this sample script uses the application A/P Standard Voucher Entry (P0411). This section includes step-by-step instructions for developing a sample script for this application. This sample script does not provide examples of every function or feature of AutoPilot. For example, this script tests an interactive application and does not launch a UBE. Consult other sections of this guide if you need information about a function that is not included in the sample script.

The steps for writing a script vary from one script to another. The precise steps that you include in a script are mainly determined by your knowledge of a specific application.

Creating the Sample AutoPilot Script

You use AutoPilot to create customized scripts that apply to the specific applications that you most often use. The sample script presented in this documentation illustrates how you use many of the commands that are included in AutoPilot. However, the sample script does not include all context and action commands, nor does it represent a definitive method for using AutoPilot. For example, this script tests functions that you perform in an interactive application. You might also want to test the launch and submission of a UBE, which require you to write a different set of commands.

Launching an Application and Form

Launching the application from AutoPilot establishes one basic scripting context that you need for many other scripting commands that you might want to insert and run. Launching an application does not have to be the first command that you script in an AutoPilot session. However, Application is often at least one of the first commands that you script.

Choosing an application in AutoPilot also requires that you choose a form that is part of that application. After you launch an application from AutoPilot, you can script a variety of additional context and action commands. For example, after you have established the context as a form, you can then script inputs for header controls, grid columns, and QBE lines. After you establish one of these contexts, you can script pressing buttons to move to different forms or to perform functions within the active form.

► To launch an application and form

1. From your desktop or the appropriate directory, launch AutoPilot.

   The AutoPilot splash screen appears, followed by the AutoPilot form.

2. From the File menu, choose New.

   The command pane and script pane are unpopulated.

3. From the Command menu, choose Application.

   The Application list is populated, while the Menu list remains unpopulated.

4. Click an application code, such as P0411 for A/P Standard Voucher Entry.

AutoPilot populates the Menu list with items that appear under the following headings:

- Fast Path, which corresponds to the fast path command that AutoPilot will use to launch the application
- Menu Text, for example, Standard Voucher Entry
- Version, which specifies which version of the application will be launched

5. Click a Fast Path command to launch the application that you chose, such as 3/G0411, Standard Voucher Entry, version ZJDE0001.
6. Click the Insert button in the lower right corner of the command pane.

Note
If you have the playback button in the tool bar turned on, AutoPilot launches the software, and the form that you specified in the Fast Path (in this case, Supplier Ledger Inquiry) appears.

Declaring a Variable

Before you can use a variable as a source of input, you must first declare it, or give it a name, to specify the place in which you store the value. You can declare a variable at any point in the script, but after you declare a variable, you might want to place it at the top of the script to make it global, meaning that you can set its value at any point in the script. If you make the variable global, you can launch multiple applications within the script and use the stored value in any of the applications. If you decide to declare a variable after you launch an application, you can use the mouse to drag the Declare command for the variable to the top of the script to make it global.

For this script, you use a previous document number to retrieve voucher entry data. You declare the variable early in the script so that you have a place already established to store the previous document number as soon as you know what it is.

► To declare a variable

1. From the Command menu, choose Variables.
2. Type a name for the variable in the New Variable field.

   For this exercise, call the variable Previous Document #.

3. In the Source of Value list, click Unknown/None.

Note
Choosing Unknown/None means that you want to give the variable a name at this point. You do not yet set its value.

4. Click the Insert button.

   AutoPilot inserts the Declare command for the variable after the Supplier Ledger Inquiry Form command line. At this point, you can use the variable only within that Form node because it is attached to the Form node.

5. Click the Declare command line in the script pane to highlight it.
6. Hold down the mouse button and drag the Declare command line until it is on top of the Application command line. When the indicator arrow is pointing up, release the mouse button.

   You have now attached the Declare command line to the Begin Script node, and you can use a value that you set for the variable at any point in the script.

Adding a New Form

For this sample script, you want to move from the Supplier Ledger Inquiry form to the Enter Voucher – Payment Information form. To do so, you script pressing the Add button to change forms. You also choose Enter Voucher – Payment Information from the Next Form list so that the Form command line in AutoPilot matches the active form.

► **To add a new form**

1. From the Command menu, with the playback button turned on, click Press Toolbar Button.
2. From the Button list, click Press Standard Button.

   Note
   The options listed under Press Standard Button in AutoPilot correspond to the buttons on the menu bar in the form.

3. Click Add.
4. From the Next Form list, choose the name of the form that appears in the software when you click Add, which is Enter Voucher - Payment Information in this example.
5. Click the Insert button.

   AutoPilot adds the commands that you inserted in the script pane. In the playback mode, the Enter Voucher - Payment Information form appears in the software.

   Note
   If you chose Unknown/None from the Next Form list in step 4, you must complete the following additional steps. You must script a Form command line that corresponds to the form that is active in the software. At all times, the most current Form command line in the AutoPilot script pane must correspond to the form that is active in the software. If the command line in the script pane does not correspond to the active form in the software, you cannot continue scripting.

6. With the AutoPilot playback button turned on so that the software is active, note the name of the software form that is active, which is Enter Voucher - Payment Information in this example.
7. From the Command menu, choose Form.

   The system displays a list of all forms within the current application.

8. Choose the name of the active form from the Form list.

   In this example, you choose Enter Voucher - Payment Information.

9. Click Insert.

   In the script pane, a Form command line appears that includes the name of the form that you chose.

Typing Data in a Header Control

Continue the sample script by adding commands to type inputs in header controls. Like Application and Form, Header is a context command because it establishes the context in which you take further actions.

► To type data in a header control

1. From the Command menu, choose Set Header Control Value.
2. In the Header Control list, choose the name of a header control into which you want to input data, for example, Company.

   Note
   The header control that you choose in AutoPilot is highlighted with a BlueCue.

3. Choose one of the following for the source of input for the control:
   - Literal Value
   - Valid Values List
   - Variable
   - UDC Visual Assist Value
   - Form Interconnect Visual Assist Value
   - Clear
   - Set Focus

   For this example, choose Literal Value. When you choose a source of input, AutoPilot captions the value selection list. The wording of the caption depends on the source of input.

4. Click inside the unpopulated Literal Value list and type 1.
5. Click the Insert button.

   AutoPilot types the scripted input in the Company header control in the Enter Voucher - Payment Information form. AutoPilot encloses literal values in quotes in the script pane.

6. Continue to script inputs in header controls by clicking Set Header Control Value in the command menu, choosing a control, choosing a source of input, and selecting a value.

Creating a Valid Values List

For this sample script, you want to script inputs in the header control Long Address Number. However, instead of entering a literal value in the control, use Valid Values List as the source of input.

A valid values list consists of values that you collect and store under a name of your choice. You use a valid values list if, for example, you want to input more than one value in a header or grid column. The following types of valid values lists exist:

- List of literal values
- Simple database query

For this script, you create a list of literal values, which contains values that you assign. With a simple database query, AutoPilot draws the values from a database that you choose. You can continue to enter literal values in each header control. However, you can include as many values in the valid values list as you want. Each time that you play back the script; AutoPilot automatically inserts one of the values in the appropriate header control.

If the value that you want to insert in a header or grid column is constant, choose a literal value. However, if the value is likely to change, you might prefer to create a valid values list so that AutoPilot inserts a new value each time. In this example, the long address number is different for each vendor, so create a valid values list and give it the name Vendors.

► To create a valid values list

1. From the Tools menu, choose Generate Valid Values List.
2. Choose the List of Literal Values option from the Select Data File Type window.
3. Click Next.
4. Type a file name in the File Properties list. For this example, name the list Vendors.
5. Enter one or more values in the Enter Values list. The values should be stacked vertically in the box.
6. Click Finish.
7. From the Command menu, choose Set Header Control Value.
8. In the Header Control list, choose the name of a header control into which you want to input data, for example, Long Address Number.
9. Click Valid Values List as the source of input.
10. Click the name Vendors, which is the name of the valid values list that you created.
11. Click the Insert button.

    AutoPilot types the first value in the list in the Long Address Number control in the Enter Voucher - Payment Information form. AutoPilot identifies the valid values list in the script pane with a backslash, the name that you assigned to the list, and the extension .atd.

12. Complete the following tasks:
    - Script an input to the header control Business Unit. Use a literal value of 1.
    - Script an input to the header control DateForGLandVoucherJULIA. Use a literal value of 063005.

- Script an input to header control DocVoucherInvoiceE, using a list of literal values that you created and gave the name Sequential 5-Digit Numbers.

When you have inserted inputs in each of the header controls, the script pane should contain five Type to action commands within the Header node.

Typing Data in a Grid Column

You now decide to script inputs in the grid columns. For this sample script, you need to make voucher payment inputs in the grid columns. The Grid command, like the Header command, is a context command because it establishes the grid column in the form as the environment in which you take additional actions.

The names of the grid columns in the Grid Column list of the AutoPilot command pane match the names of the grid columns on the form. When you choose the name of a grid column from the Grid Column list, a BlueClue appears as an arrow over the corresponding grid column in the form.

For the sample script, enter vouchers in multiple rows of the grid columns of the form. You can enter literal values in each grid row, but it is easier to create a valid values list and then update the repeat count of the command line. For this script, type inputs in the Gross Amount and Remark grid columns.

► To type data to a grid column

1. Create a list of literal values that contains five values and name it Random Dollar Amounts. This list contains gross amounts paid to vendors.
2. From the Command menu, choose Set Grid Cell Value.

   The command pane includes the following lists:

   - Grid Column
   - Source of Input
3. Choose the name of a grid column from the Grid Column list. For example, choose Gross Amount.
4. Click Valid Values List in the Source of Input list.
5. Click Random Dollar Amounts.
6. Click the Insert button.
7. Repeat steps 1-6 to script an input in the Remarks column.

   Name the list of literal values Random JE Explanations. This list contains explanations for each entry in the Gross Amount column, such as Rent.

   After you insert inputs in each of the grid columns, the script pane should contain two Type to action commands within the Detail Information (grid) node.

Updating the Repeat Count

Because you scripted as the source of input two lists of literal values that contain five values each, you might want to change the repeat count for this node so that each of the five values is input in the grid during playback.

► To update the repeat count

1. If the playback mode is turned on, turn it off by clicking the Playback button in the tool bar.
2. In the script pane, click Detail Information, the node that you want to update.

   In the command pane, review the following lists:

   - Define repeat count from
   - Repeat Count
3. Choose Literal Value from the Define repeat count list.
4. In the Repeat Count list, type a number.

   In this case, you type the number 5, because you want to script entering five separate values that you included in your valid values lists.
5. Click Update.

   Note
   The repeat count in the Detail Information node is now 5. If you play back the script, AutoPilot loops through this node five times. Each time it loops, AutoPilot inserts in the Gross Amount and Remark grid columns a different value from your valid values lists.
6. Click the Playback button in the tool bar to turn on the playback function.

Updating the Database

Having scripted the desired entries in the Enter Voucher – Payment Information form, you now can update the database with the new entries by clicking OK. For this script, you also write a new Form command because clicking OK opens a new form.

In the new form, you enter data, click OK to add it to the database, and then return to the previous form by writing a new Form command.

Before You Begin

- ❑ Create a list of literal values to use in Account Number column of the Enter Voucher - G/L Distribution form. Name this list Random JE Account Numbers. See *Creating a List of Literal Values* in the *AutoPilot Guide*

► To update the database and confirm a new form

1. From the Command menu, choose Press Toolbar Button.

   Review the following lists in the command pane:

   - Button
   - Next Form

   The Button list in the command pane has a default value of Press Standard Button. For this script, under Press Standard Button, the options match the following buttons on a form:

   - OK
   - Delete
   - Cancel

2. Choose OK from the Button list.
3. From the Next Form list, choose the form that follows when a user clicks OK for this application and version. For the sample script, you choose Enter Voucher - G/L Distribution.
4. Click Insert.

   The Enter Voucher - G/L Distribution form appears. Credit the full voucher payment amount to a particular account number. After you credit the account, update the database, and then return to the Enter Voucher - Payment Information form. To do so, complete the following tasks:

   1. Enter an account number in the appropriate grid column in the G/L Distribution form.
   2. Click the OK button to update the database.
   3. Confirm the Enter Voucher - Payment Information form.

Note
You can create the list of literal values before you write the script. You can also enter a valid literal value in the Account Number column.

► To enter data, update the database, and return to a previous form

1. Create a list of literal values.

   Name the list Random JE Account Numbers. This list contains valid G/L bank account numbers.

2. From the Command menu, choose Set Grid Cell Value.
3. In the Grid Column list, choose Account Number.
4. Click Valid Values List in the Source of Input list.
5. Click Random JE Account Numbers.
6. Click Insert.

After you enter data in the Account Number grid column in the Enter Voucher - G/L Distribution form, and distribute the amount from the Enter Voucher - Payment Information to the indicated account number, you update the database and return to the Enter Voucher - Payment Information form.

7. From the Command menu, choose Press Toolbar Button.
8. Choose Standard Button.
9. Choose OK.
10. In the Next Form list, choose Enter Voucher - Payment Information, and then click the Insert button.

Setting the Value of a Variable

After you return to the Enter Voucher – Payment Information form, retrieve the previous document number. You will use it later to search in the Supplier Ledger Inquiry form for the data you input in the Gross Amount and Remarks grid columns in the Enter Voucher – Payment Information form. You need to retrieve and store the document number, which you can accomplish by assigning its value to the variable that you declared earlier. The variable that you declared earlier has only a name, Previous Document #. Its value must be derived from a source that you choose.

► To set the value of a declared variable

1. From the Command menu, choose Variables.
2. In the Existing Variable list, choose the name of the variable that you declared earlier in the script.
3. In the Source of Value list, choose one of the following:
   - Literal Value
   - Valid Values List
   - Variable
   - Header Control Data
   - Grid Cell Data

   In this sample script, retrieve and store the previous Enter Voucher - Payment Information document number. Because the previous document number appears in a header control, choose Header Control Data as the source of value.
4. Choose or type a value.

   In this case, AutoPilot populates the value selection list with the names of the header controls in the Enter Voucher - Payment Information form. The value selection list, therefore, is called Header Control. Choose Previous Document.
5. Click Insert.

   You have now assigned a value to the variable that you named Previous Document #. You have told AutoPilot to derive that value from the header control Previous Document in the form Enter Voucher - Payment Information. AutoPilot stores that value in your declared variable. Because you previously made the variable global, you can use this value at any point that you choose in the script.

Returning to a Previous Form

When you return to the previous form, enter the value of the variable in the QBE line on the Supplier Ledger Inquiry form. In the sample script, click Cancel to return to the Supplier Ledger Inquiry form. Choose Press Toolbar Button from the Command menu and choose Cancel from the Press Standard Button options in the Button list. You then choose the next form that appears when you press the Cancel button. Therefore, you choose Supplier Ledger Inquiry from the Next Form list.

► To return to a previous form

1. From the Command menu, choose Press Toolbar Button.
2. Choose the Cancel option from Standard Button in the Button list.
3. Select the form Supplier Ledger Inquiry from the Next Form list.
4. Click the Insert button.

   In J.D. Edwards software, the Supplier Ledger Inquiry form becomes active. In the AutoPilot script pane, the Form command line shows Supplier Ledger Inquiry.

Entering Data to a QBE Line

Because you have assigned a value to the declared variable, Previous Document #, you can now use the value that you stored.

You decide to enter the stored value, the previous document number, to the QBE line of the Supplier Ledger Inquiry form, then script pressing the Find button to retrieve the values that you entered to the grid in the Enter Voucher – Payment Information form.

► To enter data in a QBE line

1. From the Command menu, choose Set QBE Cell Value.
2. Choose the grid column Document Voucher Invoice Entry from the Grid Column list.
3. Choose Variable from the Source of Input list.
4. In the value selection list, choose the name of the variable that you declared. Remember that it also now contains the value that you set.
5. Click the Insert button.

   AutoPilot inputs the previous Enter Voucher - Payment Information document number, which you stored in the variable that you named Previous Document #, in the QBE line of the grid, in the Document Voucher Invoice Entry column.

Finding Records

To find the values that you entered in the Enter Voucher – Payment Information form, click Find. Because you do not move to a new form, do not choose another form from the Next Form list before clicking the Insert button.

► To find records

1. From the Command menu, choose Press Toolbar Button.
2. Click Standard Button in the Button list, and then choose Find.

You do not need to choose an option from the Next Form list because you are remaining on the Supplier Ledger Inquiry form.

3. Click the Insert button.

   The grid fills with the voucher entries that relate to the document number that you input in the Document Voucher Invoice Entry grid column in the QBE line.

Selecting Records and Deleting Them from the Database

Now you can script the deletion of records from the database. These records appear in the detail area of the Supplier Ledger Inquiry form because you scripted pressing the Find button in the previous task. To delete these records, you must select them, and then script clicking Delete.

When the script plays, the records that you select to delete in this task are actually deleted, just as they would be in a live session. Before you perform this task, verify that you are in a test environment and do not click the Insert button until you are sure that you have selected the correct records to delete. AutoPilot automatically clicks OK on the Confirm Delete dialog box that appears when you have selected a grid line for deletion. You cannot click OK or Cancel.

► To select records and delete them from the database

1. From the Command menu, choose Select Grid Row.
2. In the command pane, choose from the Source of Row Number list.
3. Type a literal value of 1 in the value selection list.

   By typing this value, you choose the row that contains the record that you want to delete.

4. Click the single click option in the Action on grid row list.

   This command selects the row in the detail area of the grid.

5. Click the Insert button.
6. From the Command menu, choose Press Toolbar Button.
7. Click Standard Button in the Button list, and then choose Delete.
8. Click the Insert button.

   AutoPilot clicks OK on the Confirm Delete form that appears in the software and deletes the record that you selected. You can repeat this command as many times as necessary to delete any and all records that you want to delete.

Completing the Script

At the conclusion of your scripting, you exit all open applications. For the sample script, press the Cancel button. Because you started on the Supplier Ledger Inquiry form, this command allows you to exit the A/P Standard Voucher Entry program (P0411) and complete the script.

▶ To complete the script

1. From the Command menu, choose Press Toolbar Button.
2. Click Standard Button, and then choose Cancel.

   Because you are exiting the application, do not choose an option from the Next Form list.

3. Click Insert.

   Note
   By clicking Cancel, you return to the starting point of J.D. Edwards software. You should end each scripting session by clicking Cancel or Close, particularly if you intend to create additional scripts. Leaving windows open can impede playback of your scripts and make it difficult to write additional scripts.

4. From the File menu, choose Save.
5. Type a file name in the File Name field.
6. Click Save.

   You have completed the sample script. While creating the script, you launched an application, chose a form and version associated with the application and moved between forms by pressing buttons. You typed data in header controls, grid columns, and QBE lines. You derived that data from literal values that you typed in the value selection list and from valid values lists that you created and then chose from the value selection list. You added the data to the database, retrieved it, and deleted it. You declared a variable, set its value, and used it as a source of input in a QBE line. Finally, you canceled the application and saved the script.

Storing Scripts and Test Results

Scripts that you write are reusable, dynamic objects that are useful beyond the time that you complete them. AutoPilot not only allows you to write and run scripts; it allows you to make those scripts part of a larger base of knowledge about J.D. Edwards ERP software and to manage batch testing of scripts.

The script repository is a key component of the AutoPilot knowledge base. The repository is a database of scripts that you can add to or draw from. The database is stable because reposited scripts are controlled copies that can be changed only by the owner or an administrator who has permissions. The database is varied because many people with different areas expertise might contribute to it. Finally, the database is organized because you can assign defining properties to each script that you reposit. These properties help categorize your script by application, for example.

Capturing and storing test results is another important way in which AutoPilot allows you to build a knowledge base about J.D. Edwards ERP software. If you configure AutoPilot to capture playback results, it generates an event stream during playback. The event stream is a chronological, time-stamped record of AutoPilot and J.D. Edwards ERP events that occur during playback. AutoPilot stores these test results locally and in a repository, the AutoPilot Playback Results Detail Table (F97214). You can use these results to troubleshoot J.D. Edwards ERP processes. For example, you might identify a processing error or isolate a J.D. Edwards ERP error message.

The results repository is an important part of the J.D. Edwards automated testing process. To analyze playback events in detail, you can import an event stream from the repository to the J.D. Edwards Analyzer Tool. You might also import an event stream to the Virtual Script Editor, which is part of J.D. Edwards Virtual AutoPilot. Using the Virtual Script Editor, you can generate from the event stream a virtual script. You can use the virtual script to simulate, on a single workstation, multiple J.D. Edwards ERP users.

Finally, AutoPilot helps you manage the testing of your scripts. Using the AutoPilot Test Manager, you can create playlists of locally saved and reposited scripts and conduct batch testing, which frees you from the time-consuming chore of running one test at a time. You might use Test Manager to conduct testing of an entire suite of applications in one session.

See Also

- *Analyzer Tool Guide*
- *Virtual AutoPilot Guide*

Understanding the Script Repository

The script repository is similar to a library. It is a centralized location where you can find and retrieve scripts. You can search for scripts in the repository by browsing through all of the scripts that are available, or by focusing your search on, for example, scripts that test a particular suite of applications.

In addition, much like a library of books, the script repository acquires new materials. Each time that you or someone else creates a script, you can assign distinguishing properties to it and then add it to the repository, from which it can be retrieved and viewed by others. Scripts can circulate freely among those who have access to the system, or you can assign levels of security to scripts to restrict their circulation.

You can also use the repository differently than you would a library in that you can change the materials that you remove from it. For example, you check a script out of the repository with the intention of not only viewing it, but also changing it. Therefore, the AutoPilot script repository is more than a simple storage unit that holds old scripts and acquires new ones. It also permits a dynamic interchange between those that use it.

You can use the script repository to build a database of scripts. You use AutoPilot to do the following:

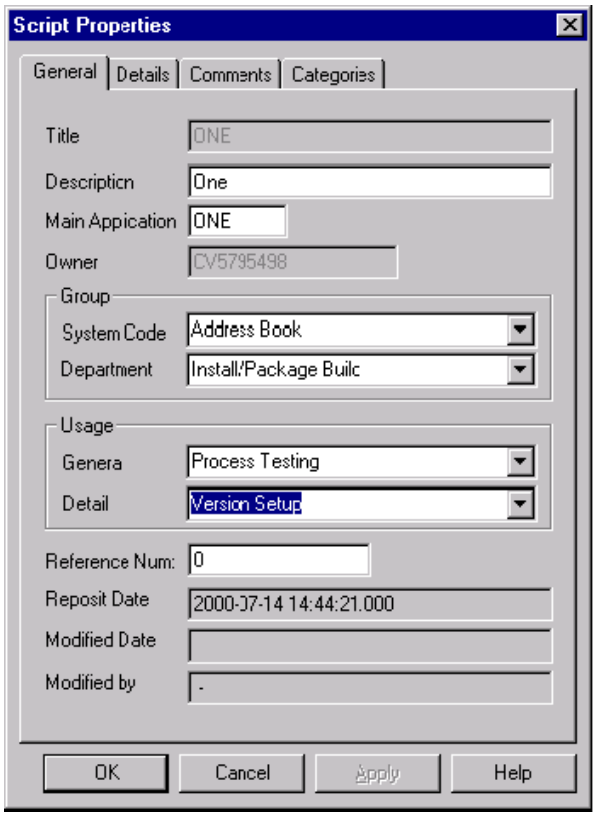
- Categorize scripts according to a set of user-definable criteria.
- Identify scripts with unique names.
- Add scripts to the repository.
- Browse for scripts.
- Check scripts in or out of the repository.
- Retrieve copies of scripts.
- Modify scripts.
- Assign security to scripts.
- Track changes that you make to scripts.
- Identify scripts that are included in other scripts.

Script Categorization

Much as a librarian would in adding a book to a library's collection, when you add a script to the repository, you assign it properties, such as title, description, the application that the script tests, the purpose of the test, and so on. The properties pages that are attached to each repository script provide important summary information for those who check out a script, and they provide a way for you to categorize scripts, which makes them easier to find.

Categorizing scripts also assists you when you want to run batch test scripts using AutoPilot Test Manager. Using Test Manager, you can browse the repository for scripts in a particular category, add them to a playlist, and automatically play them back.

Access the Script Properties form by choosing Properties from the File menu. The Script Properties form contains controls with scroll buttons.



The combo boxes contain user-defined values. You choose from these values to categorize your script. To ensure that the information in the database is consistent, reliable, and easy to access by browsing, use a consistent set of user-defined values, which you maintain in the User Defined Code Types table (F0004) and the User Defined Codes table (F0005), rather than using individual user text entries.

You can add to the values that appear in the combo boxes by using the User Defined Codes program (P0004A). The following table lists the relevant UDC codes that AutoPilot uses to populate the combo boxes in the Script Properties form:

Product Code (in P0004A)	**User Defined Code (in P0004A)**	**Combo Box in Script Properties Form in AutoPilot**
98 (Technical Tools)	SY (System Code)	System Code field on General tab
H97 (Benchmarking/Performance)	DN (Department Name)	Department field on General tab
H97	GU (General Usage)	General Usage field on General tab
H97	DU (Detail Usage)	Detail Usage control on General tab
H97	OT (Other)	Test Case control on Details tab

Note

The user defined combo box values also appear in the Select Script form and the Add Script to Repository form.

See Also

For more information about Test Manager, see the following topics:

- *Understanding AutoPilot Test Manager* in the *AutoPilot Guide*
- *Managing Script Testing* in the *AutoPilot Guide*

Property Pages for Scripts

As you add a script to the repository, you should complete property pages that provide fundamental information about it, such as title, description, owner, the application that the script tests, and so on. Completing property pages can also help classify the script as part of a large-scale testing effort. For example, you can designate the following script properties that make the script part of a larger suite of scripts:

- System code
- Department
- General use, such as benchmarking
- Detailed use, such as batch applications

You document the properties of your script by entering information in the Script Properties form. When you save the script, AutoPilot saves the data that you entered in the property pages along with the script.

When you add the script to the repository, AutoPilot saves the property page data in the database. This data loads when you check out a script from the repository, and it overwrites any subsequent property page changes that you might have made in the local script.

General Tab

The General tab of the Script Properties form contains a series of fields in which you enter data that defines your script.

The information that you enter on this tab provides baseline information about the script and its origins. You enter data in nine fields on the General tab of the Script Properties form:

Field	Purpose
Title	Script title that AutoPilot automatically enters when you save the script. You can change the title after you check in the script.
Description	Brief description of the script, such as the function that it tests.
Main Application	The primary application that your script tests.

Owner	Script owner, automatically identified as the person who adds the script to the repository. You can change the owner after you check in the script.
System Code	A user defined reporting system code.
Department	A user defined department or group name.
General	The general testing purpose of the script, such as benchmarking. The values for this parameter are user defined.
Detail	The specific testing purpose of the script, such as testing batch applications. The values for this parameter are user defined.
Reference Number	A code that identifies the script. For example, you might use this code to enter a SAR number that the script tests, or a regression test that the script runs to verify that an error has been corrected.

Note

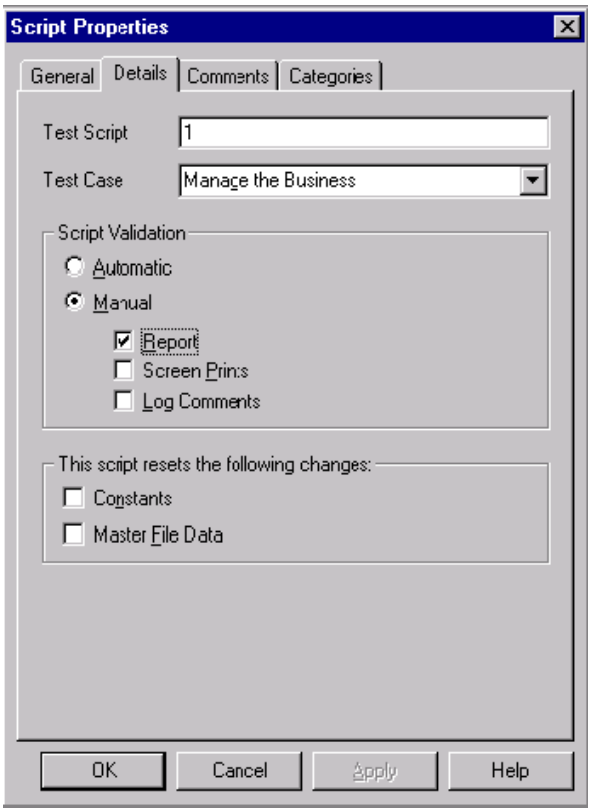
After you add the script to the repository, retrieve it from the repository, and view its properties, fields on the General tab display the reposit date and the last person to open the script, as well as the time and date that the script was opened.

Details Tab

On the Details tab, you can enter information that defines how the script fits into a test management scheme, as well as validation information and information about resetting data.

The following fields allow you to enter quality assurance-related data that is useful in large-scale testing:

- Test Script: You enter the collection of related scripts in which yours belongs, such as Tools Applications.
- Test Case: You enter the specific function that you are testing, such as turning on the Address by Effective Date feature. The values for this parameter are user defined.



Click one of the following options to indicate whether script validation occurs automatically or if you need to review the script output manually to determine if it ran successfully:

- Automatic
- Manual

You choose the automatic validation option if simply running the script successfully means that the functions you tested worked, and you do not need to further test the results. However, in some cases, such as when you test UBEs, you must manually review the output of the script to determine whether it was successful. For example, you might need to verify that a UBE report generated successfully.

If you click the Manual option, AutoPilot enables the following additional options:

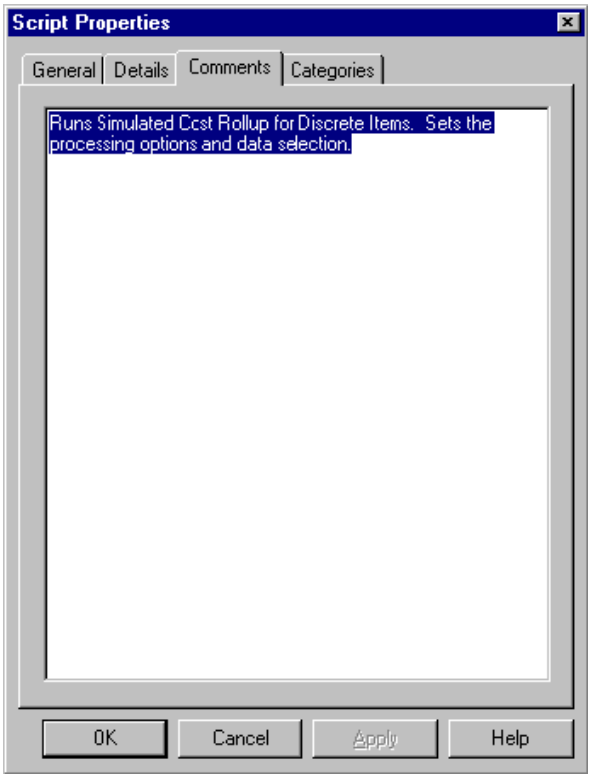
- Report
- Screen Prints
- Log Comments

Choosing one or more of these options reminds those who run the script to manually review the chosen output after the script runs.

Finally, you can indicate, using two more options, whether your script resets changes that you made to constants or to master file data, such as additions to or deletions from the Item Master table (F4101).

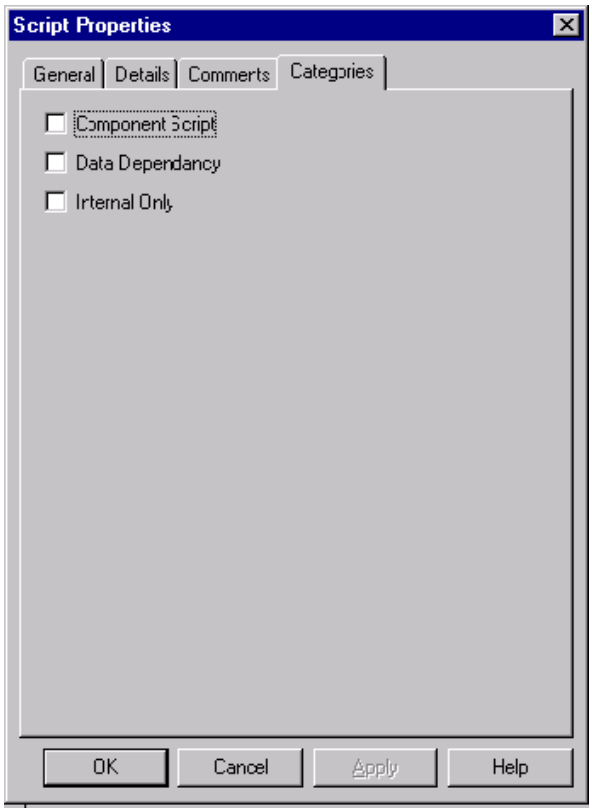
Comments Tab

On the Comments tab, you can enter additional descriptive information about the test, the purpose of the script, and any other information that you think is relevant.



Categories Tab

The Categories tab allows you to further define the object of your testing. Your system administrator creates user defined testing categories, which appear in the form as options. For example, you might define a category of testing as package verification. Options on the Categories tabs, such as Daily Build or Weekly Package, might indicate the type of verification testing that the script performs.



Naming Conventions for Saved Scripts

No predetermined rules exist for naming scripts that you intend to add to the repository. However, you should choose a specific naming convention for all scripts. For example, you might give your script a title that identifies the application, release, or function that it tests. In addition, you might want to specify whether the test is new or a retest. Following a naming convention also can help other users identify the purpose of a script. The most important purpose of assigning properties to the script is to accurately subcategorize it, and adhering to a naming convention helps accomplish that goal.

See Also

- *Script Retention and Reuse* in the *AutoPilot Guide*

Add to Repository Command

After you create a script, establish its properties, give it a unique title, and save it locally, you can add it to the repository using the Add Script to Repository form. You can use this form to change the properties of the script before you add it to the repository; but you cannot specify the owner because AutoPilot automatically assigns the owner ID, closes the script to prevent further changes, and adds it to the repository.

Browse Repository Scripts Command

The Browse Repository Scripts command allows you to search for scripts in the repository. You can make your search as narrow or as broad as you like, and after you have found the script that you are looking for, you can get a copy of it or check it out of the repository.

You use the following two forms to complete the Browse Repository Scripts command:

- Select Script
- Browse Scripts

You use the Select Script form to establish search criteria. The Browse Scripts form contains the titles of and information about the scripts in the repository that have properties that match the criteria that you specified in the Select Script form.

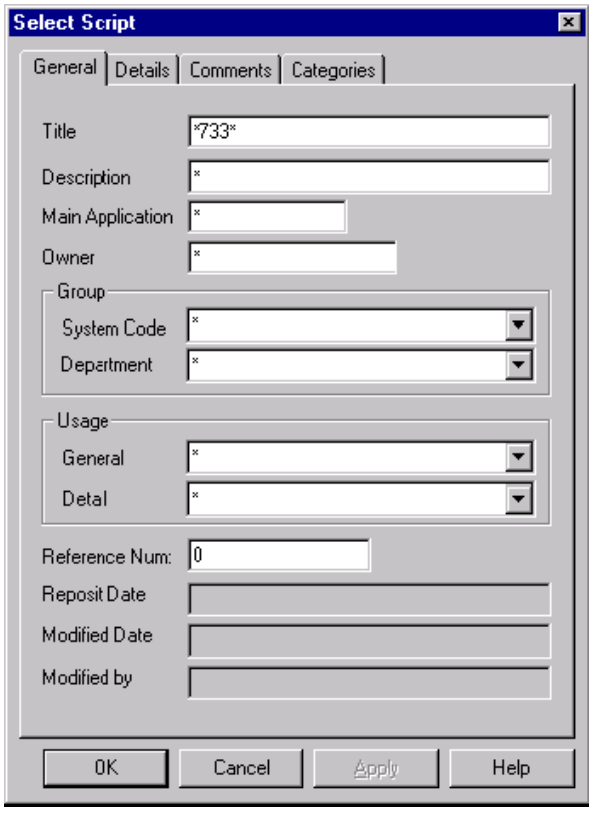
Select Script Form

The Select Script form contains the same tabs and fields as the Script Properties form contains. However, it is a query form rather than a form for entering script properties.

You can click any of the tabs, except Comments, and enter data in the fields or choose options to establish search criteria for a type of script for which you are looking, or you can enter the exact title of a script. AutoPilot matches the criteria that you set and the entries you and others made in the Script Properties pages before adding scripts to the repository. If you do not enter any field information, AutoPilot includes all the scripts in the repository.

All of the fields on the General and Details tabs on the Select Script form, with the exception of the Reference Number field, contain asterisks, or wildcards, that you can use to have AutoPilot include all scripts in its search.

You can use the asterisks alone or with an entry in a field. For example, if you want to find all scripts that tested the 73.3 release, you might enter *733* in the Title field. AutoPilot includes in its search all scripts that contain 733 in the title, regardless of any words that come before or after 733.



If you enter information in a field, AutoPilot automatically appends the wildcard to the text string. For example, if you enter P0911 in the Main Application field, AutoPilot returns all scripts with a main application property that includes P0911, such as P0911A, P0911B, and so on.

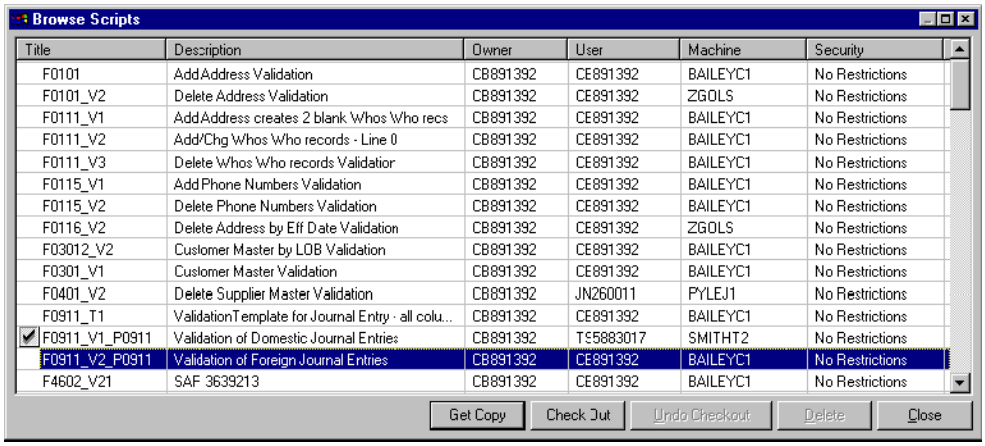
The Reference Number field contains a 0, which indicates that you have not entered a SAR number. If you leave the 0 in this field, AutoPilot does not use a reference number as a search criterion and includes all reference numbers, such as SARs, in its search.

If you enter a reference number, you limit your search to those scripts that tested a particular SAR or to tests that are defined by another reference number.

Browse Scripts Form

When you click OK on the Select Script form, the Browse Scripts form appears. This form contains summaries of any scripts that match the data that you specified in the Select Script form. The Browse Scripts form contains the following column headings, which identify a script:

- Title
- Description
- Owner
- User
- Machine
- Security



The script owner is the person designated in the Script Properties form. The user is the last person who checked out or checked in the script. The machine identifies the workstation that the user used to check out the script. Security indicates the level of security that the owner attached to the script. The security levels and their meanings are as follows:

Security Level	Meaning
No restrictions	Anyone can check out and change the script; all properties can be changed except security level.
Owner Locked	Anyone can check out and change the script, but owner and security level cannot be changed.
No Checkout/in	Those who do not own the script can only get a copy of it and save any changes locally.
No Access	Those who do not own the script can only see that it resides in the repository.

The form also contains the following buttons:

- Get Copy, which allows you to get a copy of a script from the repository
- Checkout, which allows you to check out a script from the repository
- Undo Checkout, which allows you to undo a script checkout
- Delete, which allows you to delete a script from the repository if you are authorized to do so
- Close, which allows you to exit from the Browse Scripts form

AutoPilot disables the Get Copy and Checkout buttons until you click the title of a script. After you click a title, these two buttons are enabled.

You can use the Repository Script Properties form to review the properties of any script that you want to check out by right-clicking any script and choosing Properties. This form contains the four tabs that appear on the Select Properties and Script Properties forms. You cannot change script properties using the Repository Script Properties form.

You can get a copy of or check out a single script. You can also get multiple copies or check out more than one script. You can open the checked-out copy, check it in, or close the form, in which case the script remains checked out. If you select a combination of scripts that you checked out and scripts that someone else checked out, you cannot open a script or check it in.

Script Deletion

You can delete from the repository any script that you own. However, AutoPilot places the following restrictions on your ability to delete scripts:

- You cannot delete a script that you do not own unless you have authorization.
- You must enter a password that changes each day to delete a script that you did not add but that you are authorized to delete.
- You cannot delete a script that is included in another script.

Get Copy Command

While you are in the Browse Scripts form, you can get a copy of a script from the repository. AutoPilot allows you to view and run a copy of a script that you get from the repository, but you cannot make any permanent changes to it without first saving it as a local copy in your script directory.

AutoPilot enables the Get Copy button in the Browse Scripts form when you select a script that is not checked out. If you select a checked-out script, the Get Copy command is not available. Instead, you can use the Open command to access a copy of the checked-out script.

When you get a copy of a reposited script, the copy opens in AutoPilot. The form caption bar contains the word Repository, and the title and description of the reposited script.

If you make changes to the copy, then click Save, AutoPilot displays an error message advising you that the script is not checked out and that you must either check it out or save it as a local file before your changes take effect.

Checkout Command

You might retrieve a script from the repository with the intention of making changes to it. To do so, you must check out the script, make and save the changes, and then check it back into the repository. AutoPilot returns the script to the repository with the changes intact. The repository now contains a new version of the script.

When you check out a script from the repository, AutoPilot checks out to you the latest version of the script. A script cannot be checked out to more than one person at a time, which prevents two or more people from making changes to the script simultaneously. If the script is already checked out from the repository, AutoPilot prevents you from checking out the script.

You can also run the Checkout command when you are browsing scripts. In the Select Script form, you can limit your script search by choosing options and entering information on the various tabs.

Undo Checkout Command

You can undo your checkout command if, for example, you change an existing script but decide that you want to cancel the changes. If you check in the script, the changes take effect, and the repository has a new version of the script. If you undo the checkout command, none of the changes that you made to the script while it was checked out take effect in the original version. To undo the checkout, you click File in the menu bar, then Repository, then Check In/Check Out, and then Undo Check Out. A form confirms the undo checkout.

My Checkouts

You can find scripts you have checked out in the My Checkouts form. The My Checkouts form helps you keep track of the scripts that you have checked out so that you make sure to check them back in. The My Checkouts form contains the same headings that the Browse Scripts form contains. Each document icon next to the script title contains a green check mark, which indicates that you have checked out the script on the machine that you are currently using.

You cannot check out a script on one machine and check it in on another. The My Checkouts form shows all your current machine checkouts. If you check out a script on a machine other than the one on which you are currently working, the document icon next to the script title contains a red X, which indicates that you cannot check in the script from the current machine.

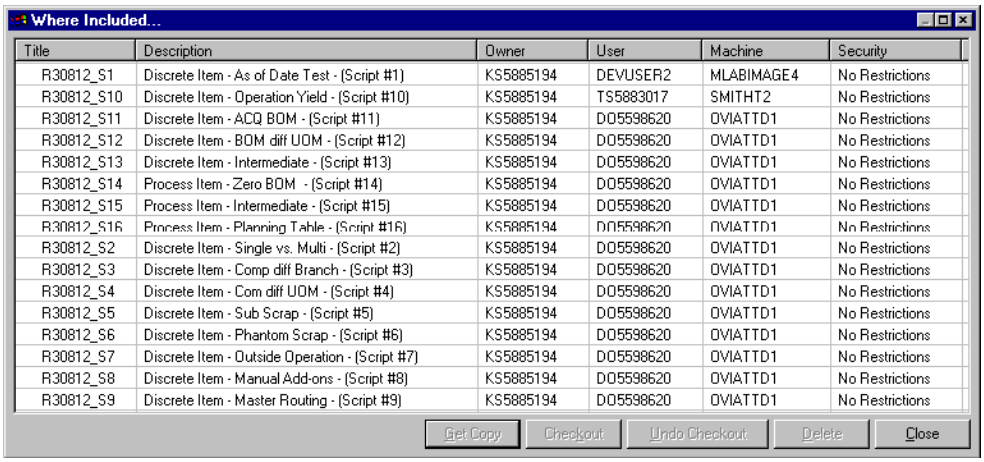
Check In Command

After you check out a script, you can work with it and make whatever changes you want to add to the reposited version. When you are satisfied with those changes, you must check in the script in order for them to take effect. With the script checked in to the repository, the new version that you created is available to others who might want to view it or work with it. When you check in a script, it automatically closes and goes into the repository, just as when you add a script to the repository.

Query by Include

You might want to know if a script is included with another script, because you cannot delete an included script. You can use the Where Included command to search the repository for all scripts that include another script.

When you enter the title of a script, AutoPilot displays on the Where Included form the titles and descriptions of any reposited scripts that include the script title.



From the Where Included form, you can choose a script and get a copy of it or check it out. The script that you select from this form is a master script, which means that it is the parent of the script for which you initially searched, as well as for any other scripts that might be included with it. If no scripts appear in the Where Included form, you can delete the script on which you conducted the search.

Working with the Script Repository

The AutoPilot script repository allows you to retrieve scripts for study, playback, and modification. You can add to the repository scripts that you create; others might retrieve copies to review the functions that you tested or to use a script as a template for another script. In turn, you can retrieve others' scripts for the same purpose. You can also check out scripts from the repository, change them, and return the new versions by checking them in.

The script repository works in conjunction with the other components of AutoPilot. After you have written a script, you can assign properties to it and save it locally. When you add it to the repository, you make it available to others in a centralized storage location, and it becomes a controlled version that can be changed only by following prescribed procedures.

Assigning Properties to a Script

Before you add a script to the repository, you might want to assign properties that remain with it when you save the script. Properties include the script title, description, main application tested, and other parameter values that you have assigned to it. These identifying features make it easier for you and your colleagues to conduct searches of the repository for scripts of a specified type. You save the property pages locally along with the script.

► To assign properties to a script

1. From the AutoPilot form, open a script.
2. From the File menu, choose Properties.

   The Script Properties form appears.

3. In the Script Properties form, enter information or choose options in the fields on each of the following tabs:
   - General
   - Details
   - Comments
   - Categories
4. Click OK.
5. From the File menu, click Save or Save As and assign to the script a title that follows the naming convention that your group or organization uses.

Adding a Script to the Repository

After you save a script and any properties that you have assigned to it, you can add it to the repository. The script must be open before you can add it. When you choose the Add to Repository command, AutoPilot allows you to assign script properties for the first time or to add to the properties that you have already assigned. After you add the script to the repository, AutoPilot identifies the script by the title and description that you entered.

► To add a script to the repository

1. In the AutoPilot form, open the script that you want to add to the repository.
2. From the File menu, choose Repository, and then Add to Repository.

   The Add Script to Repository form appears.

3. Enter any script properties that you want to assign to the script by entering information on the Add Script to Repository form.
4. Click OK.

   AutoPilot closes the script and checks it into the repository. A controlled copy of the script now exists in the repository. You can still change the local copy.

Browsing for Repository Scripts

You can browse the repository for scripts that you might want to run, use as a template, or modify. You can view all of the scripts in the repository or enter in search criteria in the Select Script form to search for scripts of a certain type. You enter information and choose options to establish search criteria. AutoPilot uses the criteria to display any scripts that have matching properties.

► **To browse for scripts**

1. From the File menu of the AutoPilot form, choose Repository, and then Browse Repository Scripts.

   The Select Script form appears. The form contains the same four tabs that appear on the Script Properties form and the Add Script to Repository form.

2. In the Select Script form, establish criteria for the kind of scripts that you want to retrieve.
3. Click OK.

   The Browse Scripts form appears and displays the titles of any scripts that matched the criteria that you specified in the Select Script form.

Deleting a Script from the Repository

You can delete a script from the repository if you are its owner or you have the proper authorization. As a precautionary measure, AutoPilot asks you to first confirm the delete.

Note

You cannot delete a script if it is included in another script.

► **To delete a script from the repository**

1. From the File menu of the AutoPilot form, choose Repository, and then Browse Repository Scripts.
2. In the Select Script form, enter your user ID in the Owner field on the General tab and click OK.

   Note
   If you are authorized to delete scripts other than those that you own, you can use other selection criteria by completing additional controls and options on the tabs in the Select Script form.

3. In the Browse Scripts form, choose the title of at least one script that you want to delete from the repository and click the Delete button.

   An AutoPilot form prompts you to confirm that you want to delete the chosen script. If you click Yes, and the script is open, AutoPilot closes the script.

4. Click OK.

   Note
   The deletion fails if the script that you want to delete is included in another script.

Assigning Security to a Reposited Script

After you add a script to the repository, you can assign security to it, or you can leave it unsecured. Assigning security to a script restricts the ability of others to access and change it.

You can assign security to a script if you are not its original owner, but only if the original script has no restrictions. However, you must make yourself the owner of the script before you can change its security.

Note

You can also assign security to a script from the My Checkouts form if you have checked the script out of the repository.

► To assign security to a reposited script

1. From the File menu of the AutoPilot form, choose Repository, and then Browse Repository Scripts.
2. In the Select Script form, enter your user ID in the Owner control on the General tab and click OK.
3. In the Browse Scripts form, choose one or more scripts and right-click.

   The system displays a pop-up menu that contains four security level choices.

4. From the pop-up menu, click one of the following to choose a security level for the checked-out script:
   - No Restrictions
   - Owner Locked
   - No Checkout/in
   - No Access

Getting a Copy of a Script

You can use the Browse Scripts form to get a copy of a script that you want to play back or to use as a template for creating another script. Important points to remember about script copies are:

- You can change the copy, but you can save the changes only to a separate local copy of the script, not to the reposited script.
- You cannot get copies of more than one script if one of the scripts that you choose is checked out.
- If you choose only the checked-out script, you get a copy of the script, including the changes made since the last checkout.

► To get a copy of a script

1. From the File menu of the AutoPilot form, choose Repository, and then choose Browse Repository Scripts.
2. In the Select Script form, establish criteria for the kind of scripts that you want.
3. Click OK.
4. In the Browse Scripts form, choose at least one script.
5. Click the Get Copy button.

Checking Out a Script

You might want to check out the script that was added to the repository. Only one person at a time can check out a script. You can make changes to a script that you check out, then check it back into the repository. AutoPilot saves all changes and creates a new version without asking you to add the script to the repository.

If someone has checked out a script, the document icon next to the script title in the Browse Scripts form contains a check mark or an X. An X appears if a script has been checked out to another computer. If you attempt to choose one or more scripts and one of them has already been checked out, AutoPilot disables the Get Copy and Checkout commands. If you choose the checked-out script only, you can open the script to check it into the repository or to undo the checkout.

► To check out a script

1. From the File menu of the AutoPilot form, choose Repository and Browse Repository Scripts.
2. In the Select Script form, establish criteria for the kind of scripts that you want.
3. Click OK.
4. On Browse Scripts, choose one or more scripts; and then click Check Out.

Undoing Script Checkout

You might want to undo a script checkout if, for example, you make changes to the script but then decide that you do not want the changes to take effect.

► To undo a script checkout

1. From the File menu of the AutoPilot form with a checked-out script open, choose Repository and Check In/Check Out, and then Undo Check Out.

   An AutoPilot form appears asking you whether you want to proceed.

2. Click Yes.

Checking in a Script

If you check out a script from the repository and make changes to it, you check it back in if you want your changes in the repository.

► To check in a script

1. From the File menu of the AutoPilot form, with a checked-out script open, choose Repository and Check In/Check Out, and then Check In.

   Note
   You can identify all of the scripts that you have checked out by clicking File, then Repository, and then My Checkouts.

   The script closes. AutoPilot checks the new script version into the repository.

► To query for included scripts

1. From the File menu of the AutoPilot form, choose Repository and then Where Included.
2. In the Where Included form, enter the title of a script and click OK.

   AutoPilot displays the titles of all scripts that include the script for which you entered the title. You can check out or get a copy of any of these scripts.

Querying for Included Scripts

You can use the Where Included form to search the repository for any scripts in which a given script is included. If a script is included within another reposited script, you cannot delete it from the repository.

Using a Command Line to Load a Repository Script

You can load any repository script into AutoPilot from a command line. The script command line parameter is passed in with an .ATR extension. This extension is used to designate repository scripts.

The command line calls the AutoPilot executable and identifies the specific repository script that you want to load into AutoPilot. The .ATR extension indicates to AutoPilot that it must access the repository to retrieve and load the script that you entered in the command line. Enter the command as follows: `C:\AutoPilot.exe` *`MyRepositoryScript`*`.ATR,` where *MyRepositoryScript* is the title of the repository script to be loaded.

Understanding Script Reporting

The script repository contains information about scripts that test particular applications and processes. The architecture of AutoPilot also contains a results repository, the AutoPilot Playback Results Detail Table (F97214). This repository contains information about the actual AutoPilot and J.D. Edwards ERP events that occur during script playback.

If you configure AutoPilot to capture and store playback events, it records each event using internally placed code and code in J.D. Edwards ERP software. When playback completes, AutoPilot sends the record of events, called the event stream, to the repository. You can view each event stream on the Test Results form.

Event Stream

The event stream provides a snapshot of the events that occurred during script playback. For example, you can see which tables were opened, which business functions were called, which event rules were invoked, and the time required to complete each event. You can also identify error and warning messages that appeared. This information can help you to troubleshoot problems that might have occurred during playback.

You also capture an event stream when you want to use J.D. Edwards Analyzer Tool to analyze script playback, or use J.D. Edwards Virtual AutoPilot to generate a virtual script that you can use to simulate multiple users on a single workstation.

Test Results Form

If you have configured script playback to capture, save, and display results, the Test Results form appears when playback completes. The Test Results form displays playback data, such as the event stream, which is a chronological listing of each event that occurred during playback. You can filter the list for test time, type, or text. You can also view previous test results, and you can review details about the results.

The Test Results form contains the following tabs:

- Browse Result Sets
- Summary
- JDE.INI
- JDE.LOG
- JDEBUG.LOG
- Screen Captures
- Messages
- Results

The Browse Result Sets Tab

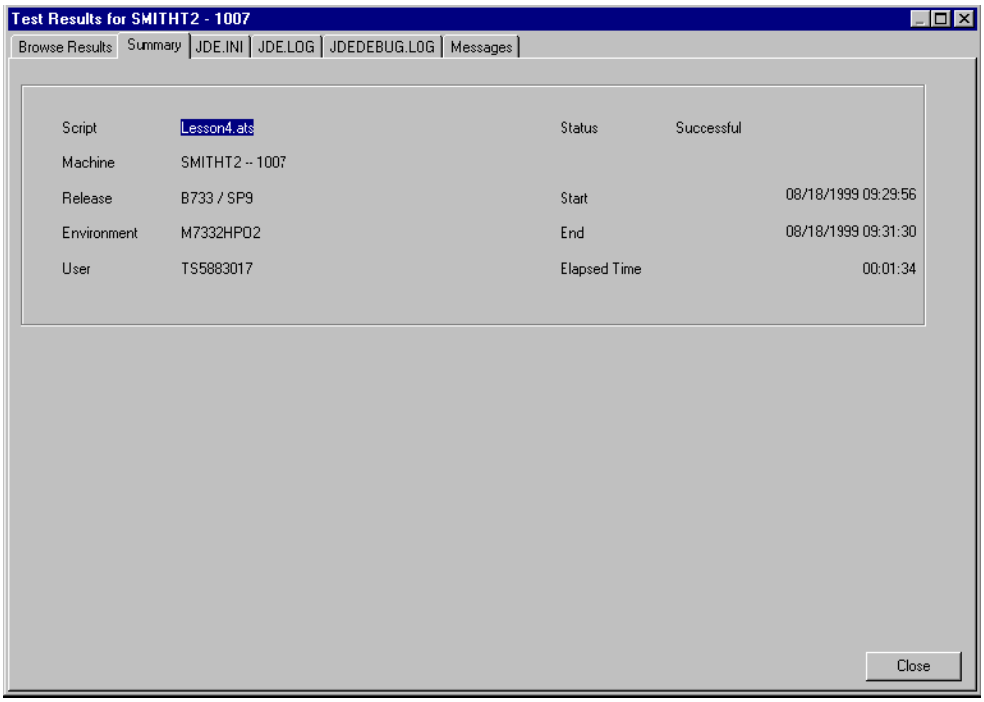
The Browse Result Sets tab contains summaries of all the tests for which you have saved results. You can also view the events in an individual test. A checkmark beside a test indicates that it was successful; an X indicates a test that failed or was cancelled.

The Test Results form for saved tests also permits you to print results and to export them to a spreadsheet using buttons at the bottom of the form. The Filter button allows you to filter the saved test results using the columns in the form as criteria. You use the Filter form to choose a filter criterion.

The Summary Tab

Clicking the Summary tab displays the following properties for each test that you run:

- Script
- Machine
- Release
- Environment
- User
- Start time
- End time
- Elapsed playback time
- Status of the playback



The JDE.INI Tab

The JDE.INI tab allows you to view the initialized settings for J.D. Edwards ERP software that existed before AutoPilot played the script. AutoPilot captures the file from C:\Winnt\JDE.INI, and then displays its contents on the tab. You can troubleshoot the file to see, for example, whether paths in the JDE.INI setting point to the correct database or drive. You can also use data on the JDE.INI tab to duplicate the results of one test in another.

The jde.log Tab

After script playback, AutoPilot captures the jde.log file from C:\jde.log and displays it on the jde.log tab of the Test Results form. You can view the contents of the file in order to track error messages that might have occurred during processing.

Screen Captures Tab

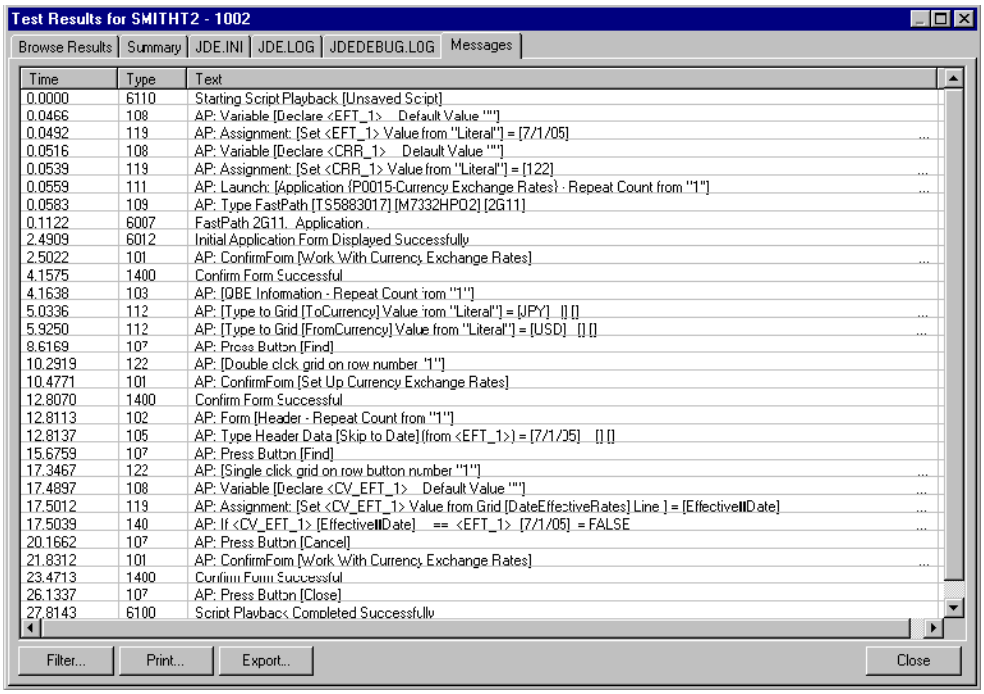
If a script fails, AutoPilot captures the screen that is active when the script fails and places the screen capture in the Screen Captures tab.

The jdedebug.log Tab

The jdedebug.log tab displays the jdedebug.log file that AutoPilot captures from the C:\jdedebug.log file after it completes script playback. You can troubleshoot the file to determine, for example, when normal execution of the script stopped. You can also review the timing of all processes that occurred during script playback.

The Messages Tab

Data that appears on the Messages tab of the Test Results form summarizes the script that AutoPilot played back.



You can review each context command and action command that you wrote in the script, as well as any error messages that AutoPilot might have generated. In addition, you can review any error messages that J.D. Edwards ERP software might have generated during playback.

On the Messages tab, you can filter, print, and export test results. You can filter test results for the following:

- A particular point during playback that an event occurred
- A particular kind of event, such as a message or an action in AutoPilot
- A text description of the event in the Test Results form

You can print your test results, provided that you have set up your default printer to do so. Using the Export button, you can export your test results to a spreadsheet.

See Also

- *Options for Configuring AutoPilot* in the *AutoPilot Guide*
- *Results* in the *AutoPilot Guide*

Results Tab

When a script finishes, you can see the results on the Messages tab in a grid format. The Results tab contains those same results in an enhanced tree control format that allows you to view individual events. You can use this tree control format to assist in troubleshooting. For example, if you are searching for errors in a script, you can identify them by the red exclamation marks next to the events.

Understanding AutoPilot Test Manager

AutoPilot Test Manager allows you to test multiple AutoPilot scripts in a batch, so that you can gather test results for archiving and review test results quickly. You use Test Manager to create a playlist that contains scripts that you saved on your local drive, scripts that you retrieved from the script repository, or a combination of both.

After you assemble a playlist, you run it. Test Manager launches AutoPilot, which, in turn, launches J.D. Edwards ERP software. Test Manager runs the playlist to completion, closing AutoPilot each time a script completes and opening AutoPilot immediately after the next script emerges in the queue. Test Manager displays a test status message of failure, success, or incomplete for each script that runs.

You can view the results of each script playback. These results include messages that help you analyze the cause of any script failure that might have occurred. Finally, you can save the playlists that you create, edit them, and replay them.

Script Display Pane

The script display pane in the AutoPilot Test Manager is the area in which you choose scripts to assemble your playlist. The pane contains two tabs, Local and Repository. When you choose the Local tab, Test Manager displays all the scripts that you have stored on your local drive.

When you click the Repository tab and then click the Repository Filter button, Test Manager launches the Select Script form, which you can use to enter search criteria for scripts that have been checked into the repository.

Test Manager populates the script display pane with the names of the scripts that match the property criteria that you enter in the Select Script form, and you can add these scripts to the playlist in the script storage pane.

Note

Test Manager creates copies of the repository scripts. It does not check out scripts from the repository. Therefore, adding a reposited script to the Test Manager playlist does not prevent other users from checking out the script from the repository and changing it.

See Also

- *Property Pages for Scripts* in the *AutoPilot Guide*
- *Assigning Properties to a Script* in the *AutoPilot Guide*

Script Storage Pane

You add scripts from the script display pane to the script storage pane to create a playlist. When you initially add scripts to the script storage pane, Test Manager displays the state of the script as Idle, meaning that you have added it to the playlist, but have not yet run it.

You can remove one or more scripts from the script storage pane. When you remove a script from the playlist, Test Manager asks you to confirm your action.

After you assemble the playlist, you run it by clicking the Run button in the toolbar. Test Manager launches AutoPilot and runs the scripts in the order that you listed them in the script storage pane. After the script runs, Test Manager displays one of the following states, depending on the results of the test: success, failure, cancellation, or incomplete.

Test Results Pane

After you have assembled and run a playlist, Test Manager summarizes the results in the test results pane. You can review the summary by clicking the Report button in the toolbar. The Test Result Summary displays the following information about the test:

- Total number of tests generated
- Status breakdown, including the number of scripts that failed, succeeded, were cancelled, or did not complete
- Name of the client machine
- J.D. Edwards environment in which the test was run
- J.D. Edwards release in which the test was run
- Name of the script
- Number of the test
- Status of each script run
- Time elapsed for each script run
- Comments you added to the script and designated for logging in Test Manager

If a script fails, click the Report button to display message types in the test results pane, along with the time of the message. These message types provide information about why the script failed, as well as information about warning messages that might have occurred during playback.

In addition, Test Manager provides information about J.D. Edwards warning messages. The following table lists some of the message types that can appear in the test results pane and summarizes their meanings:

Message Type	Explanation
110	Failure status with text ###FAILURE&&&.
138	Warning status. Each message type 138 includes the path for a screen capture that is included in the AutoPilot script.
2607	Failure status: no data returned.
2608	Failure status: unexpected records found during validation.
2609	Failure status: database validation failed.
3000	Failure status: J.D. Edwards status bar message that contains warnings or error text. Message text that includes `STB: Error...`, which indicates that the script failed. Warning messages do not indicate that the script failed. However, to help the tester, Test Manager summarizes all warning messages.
6016	Failure status: variable not found.
6301	Warning status: AutoPilot failed to set processing option text, which might cause a failure later in the script.

AutoPilot Test Manager Toolbar

The Test Manager toolbar allows you to control a test session and view its results. The Test Manager toolbar contains the following buttons:

Close	Close a test session. If you have not saved the playlist, you are prompted to do so.
Stop	Stop a script playback session.
Reset	Reset script status and test results.
Run	Initiate a test session, which launches AutoPilot.
Log	Displays the Test Results form, which contains detailed summaries of each test that you ran and saved in AutoPilot.
Report	Populates the test results pane with summary information about the playback.
Remove	Remove a script from the playlist in the script storage pane.
Add	Add a script from the script display pane to the script storage pane.
Up	Move a script up in the playlist.
Down	Move a script down in the playlist.

Managing Script Testing

You use AutoPilot Test Manager to create a playlist from scripts that reside on your local drive or in the script repository. Test Manager allows you to run multiple times, without intervention, a playlist that can contain a mixture of local and reposited scripts. You can save a playlist, or you can reset it and play it again from the top. You can view the summarized results of each playback in the test results pane, collected playback results, or the events of an individual test.

Creating a Playlist

You begin work in Test Manager by creating a playlist. You retrieve the scripts for your playlist from your local drive, from the script repository, or from both sources. The Add button on the toolbar enables you to move scripts from the script display pane to the script storage pane, where the playlist resides.

► To create a playlist

1. On your desktop or in the directory in which you store AutoPilot Test Manager, click the Test Manager executable.

   The AutoPilot Test Manager splash screen appears, followed by the AutoPilot Test Manager form.

Note
When the AutoPilot Test Manager form appears, the script display pane and the script storage pane might not display. Pull down the pane that holds them, using the grabber, which is represented by a pair of vertical bars.

2. In the script display pane of the AutoPilot Test Manager form, click either the Local or the Repository tab.
3. If you click the Local tab, choose a local script from the script display pane and click the Add button in the toolbar.

   AutoPilot Test Manager adds the test that you chose to the script storage pane.

Note
You can choose more than one test by choosing a script in the script display pane, holding down either the Control or the Shift key, and choosing another script or scripts.

4. Click the Repository tab.
5. Click the Repository Filter button.

   AutoPilot Test Manager displays the Select Script form.

6. In the Select Script form, choose any script criteria that you desire to narrow the number of scripts that you want to copy from the repository, and then click OK.

   AutoPilot Test Manager displays in the script display pane any repository scripts that match your search criteria.

7. Choose one or more repository scripts from the script display pane, and then click the Add button in the toolbar.
8. Continue adding local and repository scripts until you have created the playlist that you desire.
9. To change the sequence of the scripts, click a script in the script storage pane and click the Up or Down button in the toolbar.
10. To remove a script from the playlist, select it in the script storage pane and click the Remove button in the toolbar.

Saving a Playlist

After you create a playlist, you might want to save it, although you can run the test before you save it. Remember that if you do not save the playlist, Test Manager prompts you to do so when you exit from the form.

► To save a playlist

1. From the file menu of the AutoPilot Test Manager form, choose Save or Save As.
2. Assign the playlist a name and save it to the drive, directory, and file that you desire and click Save.

Note
AutoPilot Test Manager assigns to all playlists the default extension of `.apl`.

Running a Test

After you create a playlist, you can run the test. Test Manager launches AutoPilot, and then runs each script in the queue in the order that you set up in the script storage pane.

Test Manager launches AutoPilot, minimizes the AutoPilot form, and then begins running the first script in the queue. As each test completes, Test Manager displays its result in the script storage pane. When Test Manager finishes running a script, it closes AutoPilot, and then relaunches it with the beginning of the next script in the queue.

As each script completes running, Test Manager displays the result Success, Failure, Canceled, or Incomplete.

► To run a test

1. In the File menu of the AutoPilot Test Manager form, choose Open.
2. Open the drive, directory, and file in which you store your playlists, choose one or more playlists, and click Open.
3. In the toolbar of the AutoPilot Test Manager form, click Run.

   Note
   During playback, J.D. Edwards software remains open, unless a script contains an Exit OneWorld command, in which case J.D. Edwards software closes. In this case, with the beginning of the next script in the queue, AutoPilot Test Manager launches AutoPilot, which launches J.D. Edwards software.

4. To stop the test, click the Stop button in the toolbar.

Viewing Test Results

After Test Manager completes the playlist, you can review the playback results in one of two ways. Click the Report button to review in the test results pane a summary of the results from the current playlist. To view summaries of the tests from all the playlists that you ran and saved, click the Log button.

► To view test results

1. In the AutoPilot Test Manager form, open a saved playlist.
2. After AutoPilot Test Manager has completed the playlist, click the Report button in the toolbar.

   AutoPilot Test Manager populates the test results pane with a summary of the results for each script in the playlist.

3. To view summaries of all scripts that you have played back, click the Log button in the toolbar.

   AutoPilot Test Manager displays the Test Results form, which contains summary information about the results of all played-back scripts.

4. To view in detail all the events for the playback of an individual script, select the script in the Test Results form, and then click the Results tab or the Messages tab.

Resetting a Test

After you assemble a playlist and run it, you can reset the test, which overwrites the previous results. Resetting might be appropriate if scripts in the original test fail and you make changes to correct the failure.

► To reset a test

1. In the AutoPilot Test Manager form, open a playlist that you have already run.
2. In the toolbar, click Reset.

   If AutoPilot Test Manager displays a form advising you that resetting the test overwrites the existing results, click Yes.

3. In the Toolbar, click the Run button to rerun the test.