ORACLE
JD EDWARDS ENTERPRISEONE

JD Edwards EnterpriseOne Tools 8.97 Autopilot Guide

October 2007

ORACLE

JD Edwards EnterpriseOne Tools 8.97 Autopilot Guide
SKU E1_TOOLSTPI-B1007

Contents

About This Documentation Preface

JD Edwards EnterpriseOne implementation guides provide you with the information that you need to implement and use JD Edwards EnterpriseOne applications from Oracle.

This preface discusses:

- JD Edwards EnterpriseOne application prerequisites.
- Application fundamentals.
- Documentation updates and printed documentation.
- Additional resources.
- Typographical conventions and visual cues.
- Comments and suggestions.
- Common fields in implementation guides.

Note. Implementation guides document only elements, such as fields and check boxes, that require additional explanation. If an element is not documented with the process or task in which it is used, then either it requires no additional explanation or it is documented with common fields for the section, chapter, implementation guide, or product line. Fields that are common to all JD Edwards EnterpriseOne applications are defined in this preface.

JD Edwards EnterpriseOne Application Prerequisites

To benefit fully from the information that is covered in these books, you should have a basic understanding of how to use JD Edwards EnterpriseOne applications.

You might also want to complete at least one introductory training course, if applicable.

You should be familiar with navigating the system and adding, updating, and deleting information by using JD Edwards EnterpriseOne menus, forms, or windows. You should also be comfortable using the World Wide Web and the Microsoft Windows or Windows NT graphical user interface.

These books do not review navigation and other basics. They present the information that you need to use the system and implement your JD Edwards EnterpriseOne applications most effectively.

Application Fundamentals

Each application implementation guide provides implementation and processing information for your JD Edwards EnterpriseOne applications.

For some applications, additional, essential information describing the setup and design of your system appears in a companion volume of documentation called the application fundamentals implementation guide. Most product lines have a version of the application fundamentals implementation guide. The preface of each implementation guide identifies the application fundamentals implementation guides that are associated with that implementation guide.

The application fundamentals implementation guide consists of important topics that apply to many or all JD Edwards EnterpriseOne applications. Whether you are implementing a single application, some combination of applications within the product line, or the entire product line, you should be familiar with the contents of the appropriate application fundamentals implementation guides. They provide the starting points for fundamental implementation tasks.

Documentation Updates and Printed Documentation

This section discusses how to:

- Obtain documentation updates.
- Download documentation.

Obtaining Documentation Updates

You can find updates and additional documentation for this release, as well as previous releases, on Oracle's PeopleSoft Customer Connection website. Through the Documentation section of Oracle's PeopleSoft Customer Connection, you can download files to add to your Implementation Guides Library. You'll find a variety of useful and timely materials, including updates to the full line of JD Edwards EnterpriseOne documentation that is delivered on your implementation guides CD-ROM.

Important! Before you upgrade, you must check Oracle's PeopleSoft Customer Connection for updates to the upgrade instructions. Oracle continually posts updates as the upgrade process is refined.

See Also

Oracle's PeopleSoft Customer Connection, http://www.oracle.com/support/support_peoplesoft.html

Downloading Documentation

In addition to the complete line of documentation that is delivered on your implementation guide CD-ROM, Oracle makes JD Edwards EnterpriseOne documentation available to you via Oracle's website. You can download PDF versions of JD Edwards EnterpriseOne documentation online via the Oracle Technology Network. Oracle makes these PDF files available online for each major release shortly after the software is shipped.

See Oracle Technology Network, http://www.oracle.com/technology/documentation/psftent.html.

Additional Resources

The following resources are located on Oracle's PeopleSoft Customer Connection website:

Resource	Navigation
Application maintenance information	Updates + Fixes
Business process diagrams	Support, Documentation, Business Process Maps

Resource	Navigation
Interactive Services Repository	Support, Documentation, Interactive Services Repository
Hardware and software requirements	Implement, Optimize + Upgrade; Implementation Guide; Implementation Documentation and Software; Hardware and Software Requirements
Installation guides	Implement, Optimize + Upgrade; Implementation Guide; Implementation Documentation and Software; Installation Guides and Notes
Integration information	Implement, Optimize + Upgrade; Implementation Guide; Implementation Documentation and Software; Pre-Built Integrations for PeopleSoft Enterprise and JD Edwards EnterpriseOne Applications
Minimum technical requirements (MTRs)	Implement, Optimize + Upgrade; Implementation Guide; Supported Platforms
Documentation updates	Support, Documentation, Documentation Updates
Implementation guides support policy	Support, Support Policy
Prerelease notes	Support, Documentation, Documentation Updates, Category, Release Notes
Product release roadmap	Support, Roadmaps + Schedules
Release notes	Support, Documentation, Documentation Updates, Category, Release Notes
Release value proposition	Support, Documentation, Documentation Updates, Category, Release Value Proposition
Statement of direction	Support, Documentation, Documentation Updates, Category, Statement of Direction
Troubleshooting information	Support, Troubleshooting
Upgrade documentation	Support, Documentation, Upgrade Documentation and Scripts

Typographical Conventions and Visual Cues

This section discusses:

- Typographical conventions.
- Visual cues.
- Country, region, and industry identifiers.
- Currency codes.

Typographical Conventions

This table contains the typographical conventions that are used in implementation guides:

Typographical Convention or Visual Cue	Description
Bold	Indicates PeopleCode function names, business function names, event names, system function names, method names, language constructs, and PeopleCode reserved words that must be included literally in the function call.
Italics	Indicates field values, emphasis, and JD Edwards EnterpriseOne or other book-length publication titles. In PeopleCode syntax, italic items are placeholders for arguments that your program must supply. We also use italics when we refer to words as words or letters as letters, as in the following: Enter the letter *O*.
KEY+KEY	Indicates a key combination action. For example, a plus sign (+) between keys means that you must hold down the first key while you press the second key. For ALT+W, hold down the ALT key while you press the W key.
`Monospace font`	Indicates a PeopleCode program or other code example.
" " (quotation marks)	Indicate chapter titles in cross-references and words that are used differently from their intended meanings.
. . . (ellipses)	Indicate that the preceding item or series can be repeated any number of times in PeopleCode syntax.
{ } (curly braces)	Indicate a choice between two options in PeopleCode syntax. Options are separated by a pipe (|).
[ ] (square brackets)	Indicate optional items in PeopleCode syntax.
& (ampersand)	When placed before a parameter in PeopleCode syntax, an ampersand indicates that the parameter is an already instantiated object. Ampersands also precede all PeopleCode variables.

Visual Cues

Implementation guides contain the following visual cues.

Notes

Notes indicate information that you should pay particular attention to as you work with the JD Edwards EnterpriseOne system.

Note. Example of a note.

If the note is preceded by *Important!,* the note is crucial and includes information that concerns what you must do for the system to function properly.

Important! Example of an important note.

Warnings

Warnings indicate crucial configuration considerations. Pay close attention to warning messages.

Warning! Example of a warning.

Cross-References

Implementation guides provide cross-references either under the heading "See Also" or on a separate line preceded by the word *See.* Cross-references lead to other documentation that is pertinent to the immediately preceding documentation.

Country, Region, and Industry Identifiers

Information that applies only to a specific country, region, or industry is preceded by a standard identifier in parentheses. This identifier typically appears at the beginning of a section heading, but it may also appear at the beginning of a note or other text.

Example of a country-specific heading: "(FRA) Hiring an Employee"

Example of a region-specific heading: "(Latin America) Setting Up Depreciation"

Country Identifiers

Countries are identified with the International Organization for Standardization (ISO) country code.

Region Identifiers

Regions are identified by the region name. The following region identifiers may appear in implementation guides:

- Asia Pacific
- Europe
- Latin America
- North America

Industry Identifiers

Industries are identified by the industry name or by an abbreviation for that industry. The following industry identifiers may appear in implementation guides:

- USF (U.S. Federal)

- E&G (Education and Government)

Currency Codes

Monetary amounts are identified by the ISO currency code.

Comments and Suggestions

Your comments and suggestions are important to us. We encourage you to send us your feedback about our PeopleBooks and other reference and training materials. Please include the release numbers for the PeopleTools and applications that you are currently using. Email your comments to PSOFT-INFODEV_US@ORACLE.COM.

Common Fields Used in Implementation Guides

Address Book Number Enter a unique number that identifies the master record for the entity. An address book number can be the identifier for a customer, supplier, company, employee, applicant, participant, tenant, location, and so on. Depending on the application, the field on the form might refer to the address book number as the customer number, supplier number, or company number, employee or applicant ID, participant number, and so on.

As If Currency Code Enter the three-character code to specify the currency that you want to use to view transaction amounts. This code enables you to view the transaction amounts as if they were entered in the specified currency rather than the foreign or domestic currency that was used when the transaction was originally entered.

Batch Number Displays a number that identifies a group of transactions to be processed by the system. On entry forms, you can assign the batch number or the system can assign it through the Next Numbers program (P0002).

Batch Date Enter the date in which a batch is created. If you leave this field blank, the system supplies the system date as the batch date.

Batch Status Displays a code from user-defined code (UDC) table 98/IC that indicates the posting status of a batch. Values are:

Blank: Batch is unposted and pending approval.

A: The batch is approved for posting, has no errors and is in balance, but has not yet been posted.

D: The batch posted successfully.

E: The batch is in error. You must correct the batch before it can post.

P: The system is in the process of posting the batch. The batch is unavailable until the posting process is complete. If errors occur during the post, the batch status changes to *E*.

U: The batch is temporarily unavailable because someone is working with it, or the batch appears to be in use because a power failure occurred while the batch was open.

Branch/Plant Enter a code that identifies a separate entity as a warehouse location, job, project, work center, branch, or plant in which distribution and manufacturing activities occur. In some systems, this is called a business unit.

Business Unit Enter the alphanumeric code that identifies a separate entity within a business for which you want to track costs. In some systems, this is called a branch/plant.

Category Code Enter the code that represents a specific category code. Category codes are user-defined codes that you customize to handle the tracking and reporting requirements of your organization.

Company Enter a code that identifies a specific organization, fund, or other reporting entity. The company code must already exist in the F0010 table and must identify a reporting entity that has a complete balance sheet.

Currency Code Enter the three-character code that represents the currency of the transaction. JD Edwards EnterpriseOne provides currency codes that are recognized by the International Organization for Standardization (ISO). The system stores currency codes in the F0013 table.

Document Company Enter the company number associated with the document. This number, used in conjunction with the document number, document type, and general ledger date, uniquely identifies an original document.

If you assign next numbers by company and fiscal year, the system uses the document company to retrieve the correct next number for that company.

If two or more original documents have the same document number and document type, you can use the document company to display the document that you want.

Document Number Displays a number that identifies the original document, which can be a voucher, invoice, journal entry, or time sheet, and so on. On entry forms, you can assign the original document number or the system can assign it through the Next Numbers program.

Document Type Enter the two-character UDC, from UDC table 00/DT, that identifies the origin and purpose of the transaction, such as a voucher, invoice, journal entry, or time sheet. JD Edwards EnterpriseOne reserves these prefixes for the document types indicated:

P: Accounts payable documents.

R: Accounts receivable documents.

T: Time and pay documents.

I: Inventory documents.

O: Purchase order documents.

S: Sales order documents.

Effective Date Enter the date on which an address, item, transaction, or record becomes active. The meaning of this field differs, depending on the program. For example, the effective date can represent any of these dates:

- The date on which a change of address becomes effective.
- The date on which a lease becomes effective.
- The date on which a price becomes effective.
- The date on which the currency exchange rate becomes effective.
- The date on which a tax rate becomes effective.

Fiscal Period and **Fiscal Year**

Enter a number that identifies the general ledger period and year. For many programs, you can leave these fields blank to use the current fiscal period and year defined in the Company Names & Number program (P0010).

G/L Date (general ledger date)

Enter the date that identifies the financial period to which a transaction will be posted. The system compares the date that you enter on the transaction to the fiscal date pattern assigned to the company to retrieve the appropriate fiscal period number and year, as well as to perform date validations.

JD Edwards EnterpriseOne Autopilot Preface

This preface discusses Oracle's JD Edwards EnterpriseOne Autopilot companion documentation.

JD Edwards EnterpriseOne Autopilot Companion Documentation

Additional, essential information describing the setup and design of JD Edwards EnterpriseOne Tools resides in companion documentation. The companion documentation consists of important topics that apply to many or all JD Edwards EnterpriseOne Tools. You should be familiar with the contents of these companion guides:

- JD Edwards EnterpriseOne Tools Foundation Guide
- JD Edwards EnterpriseOne Tools Development Tools Guides
- JD Edwards EnterpriseOne Tools Virtual Autopilot Guide

This guide contains references to server configuration settings that JD Edwards EnterpriseOne stores in configuration files (such as jde.ini, jas.ini, jdbj.ini, jdelog.properties, and so on). Beginning with the JD Edwards EnterpriseOne Tools Release 8.97, it is highly recommended that you only access and manage these settings for the supported server types using the Server Manager program. See the *Server Manager Guide* on Customer Connection.

See Also

JD Edwards EnterpriseOne Tools 8.97 Virtual AutoPilot Guide, "Getting Started with JD Edwards Virtual Autopilot"

CHAPTER 1

Getting Started with JD Edwards EnterpriseOne Autopilot

This chapter discusses:

- JD Edwards Autopilot Overview
- JD Edwards Autopilot Implementation

JD Edwards Autopilot Overview

Oracle's JD Edwards EnterpriseOne Autopilot is an automated testing tool that you can use to create scripts to test the execution of JD Edwards EnterpriseOne applications and to perform repetitive tasks, such as loading data, entering sales orders, or creating screen shots.

JD Edwards Autopilot Implementation

This section provides an overview of the steps that are required to implement JD Edwards Autopilot.

In the planning phase of the implementation, take advantage of all JD Edwards sources of information, including the installation guides and troubleshooting information. A complete list of these resources appears in the preface in *JD Edwards EnterpriseOne Autopilot Companion Documentation* with information about where to find the most current version of each.

JD Edwards Autopilot Implementation Steps

This table lists the steps for the JD Edwards Autopilot implementation.

Step	Reference
1. Install JD Edwards EnterpriseOne	*JD Edwards EnterpriseOne Tools Release 8.97 Server Manager Guide*
2. Install JD Edwards Autopilot	*JD Edwards EnterpriseOne Release 8.12 Scripting Tool Installation Guide*

CHAPTER 2

Using JD Edwards Autopilot

This chapter provides an overview of Oracle's JD Edwards EnterpriseOne Autopilot and discusses how to set up JD Edwards Autopilot to run scripts on the web client.

Understanding JD Edwards Autopilot

You create JD Edwards Autopilot scripts by using the tool to write commands that run essential JD Edwards EnterpriseOne functions and processes, such as:

- Launching applications.
- Launching forms.
- Executing form interconnections.
- Running universal batch engines (UBEs).
- Setting processing options for interactive applications and for UBEs.
- Entering data in header controls.
- Entering data in grid columns.
- Entering data in Query By Example (QBE) lines.
- Clicking toolbar buttons.
- Clicking buttons.
- Selecting grid lines.
- Performing database validations.
- Selecting combo boxes.
- Traversing tree paths.

JD Edwards Autopilot Flexibility

JD Edwards Autopilot runs scripts against JD Edwards EnterpriseOne windows and HTML clients. If both clients have been implemented, the same JD Edwards Autopilot script can serve both platforms. The tool can do this because it reads and loads the specifications for each operation that you perform and passes the data through the operating system to JD Edwards EnterpriseOne 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.

JD Edwards Autopilot's flexibility also enables you to:

- Save scripts on your local drive or in a script repository that is shared by others.

- Run scripts in a standalone 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 error and warning messages and about script playback events, including application programming interface (API) calls.
- Send scripts to others.

JD Edwards Autopilot Advantages

JD Edwards Autopilot offers these advantages for scripts that test key business processes:

- Decreases the time and effort required to create automated test scripts.
- Remains usable despite changes in the JD Edwards EnterpriseOne software because it reads and loads specifications directly.
- Enables users to write scripts that are compatible with future releases of the software
- Presents a user interface that hides the complexity of the tool
- Provides the user flexibility to customize scripts by changing, for example, Object Configuration Manager (OCM) mappings
- Interacts with changing technologies

JD Edwards Autopilot Components

JD Edwards Autopilot contains these components, which enable you to create scripts:

- Command pane, where you make choices to define the processes that you want to run in JD Edwards EnterpriseOne software, such as launching an application
- Insert button, which enables you to insert a command to a script and create a script object that defines what the command does.
- Script pane, which contains a running log of the commands that you insert into a script.
- Toolbar, which enables you to navigate the JD Edwards Autopilot window and to resize it to your specifications.
- Menu bar, which contains the options that you need to run JD Edwards Autopilot.
- Title bar, which identifies the script on which you are working.
- Status bar, which displays information about a JD Edwards Autopilot session, including processes that the application is running and brief definitions of JD Edwards Autopilot commands.

Note. All tasks in this guide are based on the assumption that you have opened the JD Edwards Autopilot window and that you have either started a new script or opened an existing script.

Unsupported Controls

These controls are not supported in JD Edwards Autopilot:

- Wizard Forms

- Text Block Control
- Clickable Grid Column
- Media Object Control
- Saved Query Control
- Editable Parent/Child Control
- Lean Manufacturing Control
- Calendar Control
- Text Search Control
- Combo Box populated through ER - JD Edwards Autopilot could not validate

Unsupported Features

These features are not supported in JD Edwards Autopilot:

- Portal Grid Column
- Sort Switch Grid Format
- Switch Grid Format, Create New Grid Formats
- MAF – During capture, only one application stack on one browser window can be captured. During playback, only one browser window will be launched
- Application Recovery
- Type-ahead
- Mozilla and Safari browsers
- Data Browser

Prerequisite

Before you complete the tasks in this guide:

- You should have a working knowledge of common JD Edwards EnterpriseOne concepts, which you can find in the Foundation guide.
- You should also have a good understanding of at least one JD Edwards EnterpriseOne system, such as Accounts Payable or Sales Order Entry.

Setting Up JD Edwards Autopilot to Run Scripts on the Web Client

You can use JD Edwards Autopilot to run scripts on the web client. To do so, you must configure it with the necessary requirements that enable it to access the web client. If your web client and your Windows client both have the same login information, you can use the Sign On screen to set up JD Edwards Autopilot for both environments. If the logins are different, you have to set up the Windows client and the ApWebIni.ini file separately. The ApWebIni.ini file contains the login information that JD Edwards Autopilot needs to run scripts on the web client; it is located in the c:\WINNT folder.

Important! The ApWebIni.ini file should exist only in c:\WINNT. Remove any additional copies of the file from system/bin32 or any other directories that could be in the Windows search path.

Client Configuration Requirements

Before setting up JD Edwards Autopilot, verify that these requirements have been met:

- A full, working JD Edwards EnterpriseOne Windows client is installed
- All ODBC data sources necessary to run JD Edwards Autopilot exist in the web environment.
- The OCM mappings used by the web environment are identical to the Windows environment that users will log into when launching JD Edwards Autopilot. The tool locates database objects via OCM using the specified environment name. In particular, database validations will fail if the web environment is different from the currently installed Windows environment.
- The JD Edwards EnterpriseOne version located on the Windows client must match the version running on the web server.
- If running JD Edwards Autopilot against a local web server, such as Websphere Express or Oracle Application Server, ActivConsole must be running concurrently when the scripts are running. ActivConsole is used by the web server to execute local business functions.

To set up JD Edwards Autopilot to run scripts on the web client:

1. Open JD Edwards Autopilot.
2. From the Tools menu, select Options.
3. On the Options window, click the Sign On tab and enter this information:
   - User ID

     Type the user ID you use to log into JD Edwards EnterpriseOne.
   - Password

     Type the password you use to log into JD Edwards EnterpriseOne.
   - Environment

     Type the JD Edwards EnterpriseOne environment on which JD Edwards Autopilot is running.

     Note. The user must type a valid Windows environment name. An invalid entry will cause an error similar to the following to display upon startup: *Failed to locate datasource 'Object Librarian — BuildPY 'with SQL call: SELECT OMOOWN, OMDATB, OMDLLNAME, OMOMTO, ...FROM SYSBLD.F98611 WHERE OMDATP = Object Librarian — BuildPY'.*

- Role

  Type the role you are using to log into JD Edwards EnterpriseOne.

  Note. After modifying any Tools Options settings in JD Edwards Autopilot, you must restart JD Edwards Autopilot for them to take effect

4. From the Tools menu, select the Select JD Edwards EnterpriseOne Client option.
5. On the JD Edwards EnterpriseOne Client window, select HTML

   If Internet Explorer displays a Security dialog, you can prevent this from occurring by adjusting the internet security level. In Internet Explorer, from the Tools menu select Internet Options. On the Internet Options window, select the Security tab and decrease the internet security level.
6. If the Web Client environment name is different from the Windows environment name to which JD Edwards Autopilot connects, open the ApWebIni.ini file and ensure the settings are as follows:

```
[File Version]
Version=1

[Form Conversions]
W98305A=W98305WA
W98305D=W98305WD

[Control Conversions]
W98305A:57=W98305WA:118
W98305D:9=W98305WD:12

[Configuration]
UseExisting=1  (Instructs JD Edwards Autopilot to use existing browser session
                if possible)
CloseOnLogoff=1 (Instructs JD Edwards Autopilot to close browser after exiting
                 JD Edwards EnterpriseOne)

[Fast Path]
4/g4221=4/g4221W

[SignOn]
UseOverride=1  (Overrides user and password set in JD Edwards Autopilot's Tools
                 Options Signon page)
User=DR7220161
Password=DR7220161
Environment=DV810  (Use JDV810 when connecting to Web Application)
Role=*ALL
```

7. Close JD Edwards Autopilot and restart so settings will be activated.

Web Server Configuration Requirements (Administrators Only)

Though JD Edwards Autopilot is a Windows executable program, users can use it to run scripts on the Web Client. To do so, as a system administrator, you must configure the Web Client environment.

To configure the Web Client Environment:

1. Ensure that a full, working JD Edwards EnterpriseOne Web Server (JAS Server) is installed
2. Verify that the jas.ini file has these entries:

```
[OWWEB]
AutopilotIDs=TRUE
InYourFaceError=TRUE

[LOGIN]
decryptors=X|com.jdedwards.base.util.encryption.XORDecoder

[ERPINTERACTIVITY]
InteractivityLevel=HIGH
MultipleBrowserEnabled=FALSE
```

3. Restart the Web Client/server so the settings will be activated

---

Note. JD Edwards Autopilot does not work with Interactive HTML if Multiple Application Framework (MAF) is activated. Ensure that MAF is disabled.

CHAPTER 3

Using the JD Edwards Autopilot User Interface

This chapter provides an overview of Oracle's JD Edwards EnterpriseOne Autopilot user interface and discusses how to:

- Open the JD Edwards Autopilot window.
- Use panes in the JD Edwards Autopilot window.
- Use bars in the JD Edwards Autopilot window.
- Manipulate the JD Edwards Autopilot window.
- Manipulate the JD Edwards Autopilot toolbar.

Understanding the JD Edwards Autopilot User Interface

You use panes and bars in the JD Edwards Autopilot window to write commands that create a script. The window consists of two panes: the command pane and the Script pane. The command pane is the area where you make selections that create commands. As you make the selections to create a script, JD Edwards Autopilot displays the script as command lines in the Script pane, where you can move, delete, and edit commands. The window also contains four bars–the title bar, menu bar, toolbar, and status bar–which you use to create and identify the script.

The JD Edwards Autopilot window is resizeable, which means that you can 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 windows within the parent window by clicking Options on the menu bar. Additionally, the toolbar is dockable, so you can move to any position within the window or move it to the desktop.

Opening the JD Edwards Autopilot Window

This section provides an overview of opening the JD Edwards Autopilot window and discusses how to open the JD Edwards Autopilot window for scripting.

Understanding Opening the JD Edwards Autopilot Window

When you start JD Edwards Autopilot, a splash screen appears, followed by the JD Edwards Autopilot window. Using this window, you create scripts to test JD Edwards EnterpriseOne applications and carry out repetitive tasks. The JD Edwards Autopilot window initially is blank

Opening a JD Edwards Autopilot Window for Scripting

To open a JD Edwards Autopilot window for scripting:

1. From the desktop or the appropriate directory, launch JD Edwards Autopilot.
2. Select File from the JD Edwards Autopilot menu bar.
3. Click New.

   The JD Edwards Autopilot window appears with some of the toolbar buttons activated.

   The toolbar, which is located directly beneath the menu bar, contains buttons that represent the various commands, such as Application, that you can run in JD Edwards Autopilot. When you pass the mouse pointer over one of these buttons, or over one of the names in the drop-down menu under Command in the menu bar, a description of the command appears in the status bar, which is located at the bottom of the JD Edwards Autopilot window.

   When you place the mouse pointer over the splitter bar, you can change the size of the command or Script pane by dragging the bar up or down. When you initiate a scripting session, the command pane is blank. You make the command pane active by initiating a command, such as Application. You can view the names of the commands by selecting Command in the menu bar.

Using Panes in the JD Edwards Autopilot Window

This section provides an overview of panes and discusses how to:

- Use the command pane.
- Use the Script pane.

Understanding Panes

The two major components of the JD Edwards Autopilot window are the command pane and the Script pane.

The command pane is the top pane of the window and is divided into lists, from which you make choices that create commands that JD Edwards Autopilot runs in JD Edwards EnterpriseOne. The command pane also contains the Insert button, which you click to insert a command to the script.

The command pane enables you to make the selections that define a particular script of commands that JD Edwards Autopilot runs in JD Edwards EnterpriseOne. 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 appears in the bottom pane of the window. It displays a running log and detailed description of the commands that you insert in the script. From the Script pane, you can move 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.

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

Pane	Component	Purpose
Command	Lists	Make selections from or type entries in lists to define a command that JD Edwards Autopilot runs in JD Edwards EnterpriseOne.
Command	Insert button	Click this button to insert a command into the script.
Script	Insertion cursor	Move this red arrow to any spot in the script where you want to insert a new command.

Using the Command Pane

The command pane is the area where you begin writing commands to create a JD Edwards Autopilot script. You begin the command-writing process by selecting a command in the Command menu on the menu bar or by clicking buttons in the toolbar. When you select commands, distinct list areas appear in the command pane.

Note. Neither the Insert button nor lists appear until you select a command. You make selections from (or add entries to) the command pane lists. When you click the Insert button, a command line appears in the script.

If you are scripting a power form, the subforms that are associated with it display in the Form/Subform Hierarchy window on the Command pane. Click a subform to create script commands that you will use to test the functionality of the subform.

The command pane might also contain options. For example, when you select the Application option in the Command menu, the command pane contains options for Use Default Form and Processing Options Only. When you select the Select Grid Line option in the Command menu, the command pane contains options that enable 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 selections in (or add entries to) these lists in order to write the commands that you insert in a script. A command pane list can be either populated or unpopulated. You make selections from populated lists and input data into unpopulated lists. In either case, however, you use the lists to write script commands.

Populated lists in the command pane contain the items that you select to create a script command. For example, when you select the Application option in the Command menu (or click the Launch Application button in the toolbar), JD Edwards Autopilot displays applications and descriptions of each application in an Applications list in the command pane.

JD Edwards Autopilot populates a second list, the Menu list, when you click the Launch Application button. 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 reflect selections that you make in the menu bar or on the toolbar. Other populated lists include:

- 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, and options (such as radio buttons and check boxes) in forms
- Names of processing options for applications
- Sources of input to forms, such as literal values, user defined code (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 text, numbers, special characters, spaces, or a combination of these.

You can enter this in unpopulated lists:

- Literal values to be input in header controls, grid columns, or a Query By Example (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 shot
- A tree path that identifies a unique path to a node in a JD Edwards EnterpriseOne form

Insert Button

If you make selections and add entries to lists in the command pane and then click the Insert button, JD Edwards Autopilot inserts a command line in the Script pane. Each inserted command becomes a part of the script that JD Edwards Autopilot runs in JD Edwards EnterpriseOne. The insertion cursor, which appears as an arrow in the Script pane, follows the last command that you insert.

When you select an application and version from the command pane and click the Insert button, you automatically launch JD Edwards EnterpriseOne if you have activated the playback while scripting feature (identified by the initials *PB*) in the toolbar.

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

If the playback while scripting command is activated, as you make selections from the command pane in JD Edwards Autopilot and click the Insert button, the tool runs the scripted commands in JD Edwards EnterpriseOne.

Using the Script Pane

As you write a script, you can observe its progress because JD Edwards Autopilot records each command that you insert in the Script pane, which displays each command.

The Script pane consists of two components: command lines and the insertion cursor. The command lines reflect 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 indicate the context in which a command runs and the action that is taken in the context. Context commands specify *where* script actions occur, and action commands specify *which* actions occur in the script.

Command line can contain these 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 insert in the script, such as an Application command or a Press Toolbar Button command.
- A description of the general context or action in JD Edwards EnterpriseOne–for example, Application.
- A description of the particular context or action in JD Edwards EnterpriseOne–for example, {P0411 - A/P Standard Voucher Entry}.
- The source of the input into a form and its value.

  A value of *1,* for example, means that you have inserted the literal value 1 in 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. For example, to click a button in a form, you first establish the context, which is the form.

This table summarizes the context commands that you write using JD Edwards Autopilot and the results of those commands:

Context Command	Result
Application	Launch JD Edwards EnterpriseOne application.
UBE (universal batch engine)	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 the normal transaction sequence.
Processing Options	Display processing options for a selected application in the JD Edwards Autopilot command pane.
Set Header Control Value	Specify the header control in which 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 to input data.
Form	Specify the form in which to take additional actions.

Context Command	Result
UBE Selection	Launch the Data Selection form.
UBE Processing Options	Display processing options for a selection UBE in the JD Edwards Autopilot command pane.
UBE Print	Launch the 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, the Press Toolbar Button command 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. You take these actions to accomplish something in a context. For example, you store the value of a variable to use it in a header control, grid column, combo box, or tree path.

This table summarizes the action commands that you write using JD Edwards Autopilot and the results of those commands:

Action Command	Result
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	Click standard buttons in a form, perform form and row exits, submit UBEs, select a grid tab, or click the grid scroll bar button.
Press Push Button	Click special buttons that do not reside on the toolbar of forms.
Check box/Radio Button	Select check box or radio button options in the header portion of a form.
Select ComboBox Item	Select items in forms that use combo boxes instead of header controls.
Press Clickable Text	Click a link on a form.
Exit OneWorld	Exit JD Edwards EnterpriseOne.
Command Line	Encapsulate a path to another program in the JD Edwards 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 JD Edwards Autopilot to wait before proceeding with script playback.

Action Command	Result
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 a SQL statement to validate whether an expected value is returned from the database.
If <var> == <var>	Write a conditional (If /Then) statement.

See Also

Chapter 4, "Understanding Context Scripting," page 27

Chapter 6, "Scripting Actions," page 53

Chapter 7, "Working with the Script Pane," page 105

Chapter 7, "Working with the Script Pane," Modifying Scripts, page 109

Using Bars in the JD Edwards Autopilot Window

This section provides an overview of the bars in the JD Edwards Autopilot window and discusses how to:

- Use the title bar.
- Use the menu bar.
- Use the toolbar.
- Use the status bar.

Understanding the Bars in the JD Edwards Autopilot Window

In addition to the command and Script panes, the JD Edwards Autopilot window includes four bars that enable you to create a script. Bars in the JD Edwards Autopilot window identify the script, contain options and buttons for scripting commands, and identify the functions of buttons contained in the window.

Using the Title Bar

The title bar is the horizontal bar that appears at the top of the JD Edwards Autopilot window and identifies the name or title of the script that you are writing in the tool. The text enclosed in the brackets in the title bar is the name of the script.

Using the Menu Bar

The menu bar appears beneath the title bar. It is composed of options that contain drop-down menus from which you can make selections that enable you to write the script and to set up how JD Edwards Autopilot runs.

File Menu

Many of the selections in the drop-down File menu represent essential Windows functions. You can create a new script, open an existing one, close a script, save it, or print it. These options are JD Edwards Autopilot features:

Feature	Action
Send To	Enables you to send a script that you have written to another user who has access to JD Edwards Autopilot.
Properties	Enables you to assign identifying features to the script, such as the function that the script tests.
Repository	Provides 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 JD Edwards Autopilot users to obtain examples of scripts that test particular functions.
Import	Enables you to import a previously saved XML script.

Edit Menu

Two functions are available when you need to modify existing scripts. These options enable you to copy and paste command lines or branches of scripts rather than deleting a command line or branch and then having to rewrite it:

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 the insertion point.

View Menu

You use the View menu to set up the appearance of the toolbar and status bar in the JD Edwards Autopilot window.

Command Menu

The drop-down menu that appears when you select Command in the menu bar contains the commands that you can write to a script. These commands match the commands represented by the toolbar buttons that you can click to write commands to the script.

The Command menu also includes two options that are not represented by toolbar buttons. The If <var> = = <var> command represents the command to write a conditional statement.

Play Menu

The drop-down menu that appears when you select Play in the menu bar contains the names of the JD Edwards Autopilot playback functions. These functions are also represented by toolbar buttons.

Clicking Playback toggles the Playback button on the toolbar. When the Playback button is activated, JD Edwards Autopilot plays the commands that you write in the script as soon as you insert them.

Tools Menu

Using the Tools menu enables you to fine-tune the way in which a script runs, to view the results of test scripts that you have run, and to generate data that you can use in scripts. This table summarizes the options available from the Tools menu:

Tools Menu Option	Description
Generate Valid Values List	Create or select data to store in a text file that you use in the script.
Create a Script from Capture	Create a JD Edwards Autopilot script from the event stream that you capture.
Include Local Script	Select a script stored locally and then include that script within another script.
Include Reposited Script	Select a script stored in the repository and then include that script within another script.
Results	Review the results of JD Edwards Autopilot tests that you have run.
Unwanted Windows	Close unwanted windows while executing a script.

Tools Menu Option	Description
Select JD Edwards EnterpriseOne Client	Select from options to run a script to test Windows or HTML client.
Options	Launch the Options window, which contains tabs that you use to set up these options: • JD Edwards EnterpriseOne and JD Edwards Autopilot directories. • Speed. • Playback configuration. • Sign-on parameters. • Playback against JD Edwards EnterpriseOne HTML. • Script generation. • Specifications for script creation.

Generate Valid Values List

You can create a text file that contains one or more values by selecting the Generate Valid Values List option in the Tools menu. A window appears that enables you to select data and to save it in a file. You can then use this file as a source of input for a script. When you select the Generate Valid Values List option, 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.

Create a Script from Capture Option

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 scripting productivity and shorten the time it takes to learn how to use JD Edwards Autopilot. The option creates a framework of a JD Edwards Autopilot script. It does not create a script that you can run without modification.

Include Local Script Option

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

Include Reposited Script Option

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

Results Option

After you have played back a script, you can save the results of the test. JD Edwards Autopilot collects test results that you save and displays a summary of the test results in a Playback Results form. To access the Playback Results form, select Results from the drop-down menu of the Tools option on the menu bar.

Unwanted Windows Option

You use the Unwanted Windows option to close windows that appear as part of the system flow but are not needed for the script's 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 the action that occurs when the window appears. You can specify, for example, that JD Edwards Autopilot clicks OK on the Confirm Delete message box and continue playing the script.

Select JD Edwards EnterpriseOne Client

You can run your scripts as a Windows or HTML client by selecting an option from the Select JD Edwards EnterpriseOne Client form. If you select the HTML option, you must select Options from the Tools menu and specify a server on the JD Edwards EnterpriseOne HTML tab.

Options for Configuring JD Edwards Autopilot

Select Options to display a window with these seven tabs, which contain controls and options to set up JD Edwards Autopilot for testing:

- Directories
- Speed
- Playback
- Sign On
- JD Edwards EnterpriseOne HTML
- Script Generation
- Configure

The Directories tab enables you to specify where you start JD Edwards EnterpriseOne and where you store local scripts, screen shots, 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 window appears. You use this window to specify the path to each directory and the network drive on which that directory resides.

The Format option enables you to select a particular screen shot extension, such as .tif. If you do not want the option of adding scripts to the repository, you can disable it by selecting the Disable Repository option.

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

On the Sign On tab, JD Edwards Autopilot displays the user ID, password, role, and the JD Edwards EnterpriseOne environment to which you sign in. If you are signed in to a different environment than the one that appears in the Environment control, JD Edwards Autopilot displays a window indicating that you must change the sign-in environment to match the JD Edwards EnterpriseOne environment.

On the EnterpriseOne HTML tab, you can enter the universal resource locator for a JD Edwards EnterpriseOne web server, against which you can run a JD Edwards Autopilot script.

On the Configure tab, you can do this:

- Set how often the script is auto-saved.
- Select JD Edwards EnterpriseOne specifications, such as whether hidden edit and grid controls appear, and whether the system rebuilds file specifications each time that you run an application or only when JD Edwards Autopilot does not find the specifications.
- Set the threshold at which JD Edwards EnterpriseOne idles.
- Click the Rebuild F9860.ATX button to refresh JD Edwards Autopilot's list of application and report names.

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

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

Option	Description	Suggested Initial Setting
Cancel Playback on Comm Error	JD Edwards Autopilot cancels playback if a communication error occurs between client and server. Select this option when you are testing processes on a server.	Off.
Log Variables on Script Failure	JD Edwards Autopilot records the current value of variables when a script fails. This information can be useful when analyzing script failures. For example, suppose the journal date variable value is *06/03/05*. This causes the script to fail if the current year is not 2005. It is recommended that you activate this option.	On.

The bottom section of options enables 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 JD Edwards Autopilot captures.

You can import the event stream to the JD Edwards Virtual Autopilot Script Editor in Oracle's JD Edwards EnterpriseOne Virtual Autopilot tool, where you can create a virtual script. You can run the virtual script on a single workstation to simulate many users. This enables you to test the scalability of your system.

Important! JD Edwards Virtual Autopilot requires a JD Edwards EnterpriseOne Windows client. You can use JD Edwards Virtual Autopilot with JD Edwards EnterpriseOne Tools 8.97 and JD Edwards EnterpriseOne Applications 8.10 and prior. You cannot use JD Edwards Virtual Autopilot with JD Edwards EnterpriseOne Applications 8.11 and later releases, as these releases are on a web client only.

This table summarizes the options on the Playback tab that enable you set up the capture of script playback data:

Option	Description
None.	JD Edwards Autopilot captures no data about script playback.
JD Edwards warning and error messages.	JD Edwards Autopilot captures only data about warning and error messages.
Level 1 API calls.	JD Edwards Autopilot captures warning and error messages and captures information about top-level JDB and business function calls.
All API call levels.	JD Edwards Autopilot captures data about warning and error messages and information about all JDB and business function calls.

Window Menu

The Window menu provides several ways to change the size and arrangement of JD Edwards Autopilot windows. For example, you can select to tile or cascade the windows if you have several open at once. In addition, the Window menu displays the script or scripts that are currently open.

Help Menu

Selecting the About Autopilot option on the Help menu displays a copyright notice.

See Also

Chapter 6, “Scripting Actions,” Using a Valid Values List , page 55

Using the Toolbar

The toolbar is composed of buttons that you click to perform these actions:

- Script context and action commands
- Run script playback

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

See Also

Chapter 4, “Understanding Context Scripting,” page 27

Chapter 6, “Scripting Actions,” page 53

Chapter 8, “Playing Back the Script,” page 121

Using the Status Bar

To activate the JD Edwards Autopilot status bar, select it under the View menu. The status bar displays at the bottom of the JD Edwards Autopilot window and provides information about the JD Edwards Autopilot session. For example, after you have begun a session and are preparing to enter a new command, the status bar displays *Ready* to indicate that JD Edwards Autopilot is ready to accept a new command. When you pass the mouse pointer over a toolbar button, the status bar displays a description of the function.

Likewise, when you pass the mouse pointer over any item that appears in a drop-down menu of the menu bar, the status bar displays a description of the function of the item.

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

See Also

Chapter 6, “Scripting Actions,” page 53

Chapter 10, “Storing Scripts and Test Results,” Understanding the Script Repository, page 154

Chapter 10, “Storing Scripts and Test Results,” Understanding Script Reporting, page 159

Manipulating the JD Edwards Autopilot Window

This section provides an overview of the JD Edwards Autopilot window and discusses how to:

- Change the size of the JD Edwards Autopilot window.
- Arrange multiple JD Edwards Autopilot scripts.
- Size panes in the JD Edwards Autopilot window.

Understanding the JD Edwards Autopilot Window

You can easily change the arrangement and size of the JD Edwards Autopilot window and the panes within it. You can focus completely on one pane by manipulating the size of the window. 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 can also resize the JD Edwards Autopilot window and its panes when you are creating a script and playing it back. Adjusting the size of the window enables you to see both the JD Edwards Autopilot window and the forms that are active in JD Edwards EnterpriseOne.

If you close JD Edwards Autopilot and then open it again, the size of the JD Edwards Autopilot window, the arrangement of the panes, and the position of the toolbar appear as they did when you closed the session.

Changing the Size of the JD Edwards Autopilot Window

You can easily change the size of the JD Edwards Autopilot window by using the mouse. Moving the mouse pointer within the window produces double-headed arrows. You then can resize the window by dragging the borders.

Arranging Multiple JD Edwards Autopilot Windows

JD Edwards Autopilot enables 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 necessary.

You use the Window menu to change the size and arrangement of JD Edwards Autopilot windows when you work with multiple scripts. You can arrange the scripts so that you can view them simultaneously and easily move from one script to another.

If you work with multiple JD Edwards Autopilot scripts during a session, you can arrange the scripts in either cascade or tile fashion, using the Window menu. The Cascade command arranges the scripts in overlay fashion.

The top JD Edwards Autopilot window is active. Click another window to make it active. To resize a window, place the mouse pointer on a vertical or horizontal edge, press the mouse button, and drag the window in the direction that you desire.

The Tile command divides the area of the JD Edwards Autopilot window so that the existing JD Edwards Autopilot windows appear simultaneously, adjacent to one another.

To arrange multiple JD Edwards Autopilot windows:

1. Select either Cascade or Tile from the Window menu.
2. Use the mouse pointer to change the size of the parent JD Edwards Autopilot window or of any of the child windows.

Sizing Panes in the JD Edwards Autopilot Window

You can change the size of panes in the JD Edwards Autopilot window easily using the Split option, which appears in the Window menu. The Split option moves the mouse pointer 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 mouse pointer over the splitter bar.

To size panes in the JD Edwards Autopilot window:

1. Select the Split option 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 reducing the size of the panes, and click.

Manipulating the JD Edwards Autopilot Toolbar

This section provides an overview of manipulating the JD Edwards Autopilot toolbar and discusses how to:

- Relocate the toolbar.
- Resize the toolbar.
- Float the toolbar.

Understanding Manipulating the JD Edwards Autopilot Toolbar

You use the toolbar frequently during a JD Edwards 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 toolbar and change its size and shape.

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

Relocating the Toolbar

You can move the toolbar using the grabber, which is represented by a vertical bar. Two toolbars 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 scripts. Each bar contains a grabber, so you can move one, the other, or both.

To relocate the toolbar:

1. In the JD Edwards Autopilot window, place the mouse pointer over the grabber, which is represented by a vertical bar in the JD Edwards Autopilot window.
2. Holding down the mouse button, drag the toolbar, which you can now place either vertically along the right or left edge or horizontally along the bottom of the JD Edwards Autopilot window.

You can divide the two sections of the toolbar and place one along a vertical edge and one along a horizontal edge of the JD Edwards Autopilot window, or you can place them together.

Resizing the Toolbar

You can change the size and shape of the toolbar. You can do so easily by using clicking and dragging.

To resize the toolbar:

1. In the JD Edwards Autopilot window, place the mouse pointer at the edge of the window.
2. When a double-headed horizontal arrow appears, press the mouse button.
3. Drag the mouse up, down, or across.

   As you resize the JD Edwards Autopilot window, the toolbar resizes along with it.

Floating the Toolbar

You can drag the toolbar completely outside the JD Edwards Autopilot window and use it on the 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 toolbar is floating, you can use the mouse to resize it.

To float the toolbar by dragging:

1. In the JD Edwards Autopilot window, place the mouse pointer over the grabber in the toolbar, which is represented by a vertical bar.
2. Press the mouse button.
3. Drag the toolbar to any position desired.

   An outline of the toolbar appears as you drag it.
4. When the outline of the toolbar appears in the shape that you want, release the mouse button.

To float the toolbar by double-clicking:

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

To resize and reshape the toolbar from the floating position:

1. In the JD Edwards Autopilot window or on the desktop, with the toolbar in a floating position, place the mouse pointer over 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 button.

   The toolbar in its new configuration appears.
5. Double-click the top of the toolbar to return it to its original configuration.

CHAPTER 4

Understanding Context Scripting

This chapter provides and overview of context scripting and discusses:

- Context scripting.
- Context commands.

Context Scripting

To create a script using Oracle's JD Edwards EnterpriseOne Autopilot, you select 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 JD Edwards EnterpriseOne applications.

You can insert two kinds of commands in a JD Edwards Autopilot script: context commands and action commands. You use context commands to establish the setting that you test. These settings include applications, universal batch engines (UBEs), interconnected applications, processing options, forms, headers, grid columns, and QBE lines. After you establish a context, you write action commands, which accomplish specified tasks that you perform in JD Edwards EnterpriseOne software, such as clicking a button or typing in a header control.

Context commands can depend on other context commands. For example, suppose that 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.

Context Commands

This section discusses:

- Context command overview.
- Application command.
- UBE command.
- Application Interconnect command.
- Processing Options command.
- Form command.

- Header command.
- Grid Column command.
- Query By Example (QBE) command.

Context Command Overview

You write context commands during a JD Edwards 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 JD Edwards Autopilot functions that you test. In general, you must write context commands before you can decide which actions to take.

The lists in the command pane reflect the context command that you select. 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 select another context, a form. Both of the command lines appear in the Script pane, and JD Edwards Autopilot indents the Form command line beneath the Application command line. This indentation indicates that the Application command is the parent of the Form command.

This hierarchy 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 it.

This is a simple hierarchy of JD Edwards 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 usually subordinate to a primary or secondary context command.

  They typically appear as children of other commands in the script.

Note. These are generalizations. For example, when a Form command is the parent of a Header, Grid Column, or QBE command, it is primary to that command, but it is secondary to the Application or Application Interconnect command.

Application Command

You use the Application command to launch interactive versions of applications. Selecting the Application command enables you to select 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 to interconnect to another application.

UBE Command

You use the UBE command to launch previously created UBE versions when you need to submit a UBE to JD Edwards EnterpriseOne software for processing. JD Edwards Autopilot enables 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 JD Edwards 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 JD Edwards 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 select 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

JD Edwards Autopilot automatically activates both of these options when you select UBE. The lists in the command pane change to reflect whether you have activated these options. For example, if you select a UBE but clear the Execute FASTPATH option, JD Edwards Autopilot removes the Menu Item list.

The Execute FASTPATH Option

To launch a UBE from a menu, you select the Execute FASTPATH option and then select an option from each of the three lists in the command pane: UBE, Menu Item, and Version. When you click the Insert button, JD Edwards Autopilot sends the Fast Path command to JD Edwards EnterpriseOne software.

In some cases, you 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 can also access a UBE from the Batch Versions program (P98305). In addition, you might launch a UBE that is coded to automatically submit another UBE. In any of these cases, you clear the Execute FASTPATH option after you select a UBE, and JD Edwards Autopilot removes the Menu Item list that contains the Fast Paths.

The Create Work With Batch Versions Commands Option

When you select options in the command pane and click the Insert button, JD Edwards Autopilot automatically declares and sets a variable that stores the UBE version that you select or that is automatically submitted. When you select the Create Work With Batch Versions commands option, JD Edwards Autopilot performs these tasks without your intervention:

- Interconnects to the Batch Versions program.
- Launches the Work With Batch Versions - Available Versions form.
- 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 clear this option, JD Edwards Autopilot writes no script lines to exit to the Work With Batch Versions - Available Versions form. Disable 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, JD Edwards Autopilot removes the Version list from the command pane and disables both of the options. When you click the Insert button, JD Edwards 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 select.

This table lists five scenarios for launching a UBE and the combination of options that you select:

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, you must write the command to submit the UBE. You do so by writing a Press Toolbar Button {Submit} command. This command clicks the Submit button on the Version Prompting form. If you want to select data for your report, select 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. The software automatically submits the UBE, and you can write the command to print it.

UBE Data Selection

If you launch the UBE with the Create Work With Batch Versions commands option, you can also use the Criteria Design Aid feature in JD Edwards Autopilot to select the data for the report. The UBE context command and the UBE data selection action command work together when you script in JD Edwards 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 Check box/Radio Button command in JD Edwards Autopilot to select the Data Selection option, and then submit the form by writing a Press Toolbar Button command. JD Edwards Autopilot then enables you to script the data selection criteria in the command pane.

You script data selection by selecting the UBE Selection option in the Command menu or by clicking the CDA button on the toolbar. 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 enables you to click the OK button on the Data Selection form.

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, select the SKIP option to cause the Operator and Comparison entries for the new line to duplicate the entries for the previous line.

Although the data selection feature in JD Edwards Autopilot is essentially the same as the Criteria Design Aid feature, the JD Edwards Autopilot feature has some limitations. For example, you can select a left operand in the Data Selection form by clicking a selection in a drop-down menu. The drop-down menu does not exist in JD Edwards Autopilot. The name of the object that populates the left operand in JD Edwards EnterpriseOne 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 JD Edwards EnterpriseOne 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, JD Edwards Autopilot presents in the value selection list the names of all variables that you have declared and enables you to select one.

After you enter the name of a declared variable, JD Edwards 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 can write a script that enters transactions, and then launches a UBE and extracts particular data for the report. Creating a variable enables 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 the play back while scripting feature activated, you can observe the way in which JD Edwards Autopilot enters the criteria in the Data Selection form. If you select the UBE Selection option again, you can enter selection criteria on another line. When you have completed your data selection, select the Press OK option in the JD Edwards Autopilot command pane and click the Insert button. If processing options exist for the UBE version that you launch, they appear next, and you script processing options commands, as necessary.

UBE Processing Options

After you submit a UBE, some versions prompt 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 toolbar buttons that you select to set UBE processing options are distinct from those that you select to set processing options for interactive applications.

When you set processing options for a UBE version, you select the UBE Processing Options option in the Command menu. You then select options from the Processing Options list in the command pane, and then write a Press Toolbar Button {OK} command. JD Edwards Autopilot inserts the selected processing options in the script and runs them during playback.

UBE Print Command

You can send your UBE to print after you have submitted it or after JD Edwards EnterpriseOne has automatically submitted it. You do so by selecting the UBE Print option in the Command menu or clicking the Stop button on the toolbar.

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

Wait for UBE to Complete Option

When you select this option, JD Edwards 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, JD Edwards Autopilot selects the printer queue. When the printer completes all of the submitted UBEs, JD Edwards Autopilot resumes playing the script.

If you clear this option, JD Edwards 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 select it. In either environment, JD Edwards Autopilot does not require your intervention to handle 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 JD Edwards Autopilot from waiting for a printer window to appear before JD Edwards Autopilot continues running the script. If you clear this option and a printer window does not appear, JD Edwards Autopilot continues to wait, and the script fails to advance. If you select this option, JD Edwards Autopilot does not wait for a print window and it continues 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, you can select the Create exit Work With Batch Versions command option. Then, JD Edwards 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 select this option.

Application Interconnect Command

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

Selecting Application Interconnect in the Command menu enables you to use the Script pane in JD Edwards Autopilot to insert new Application and Form command lines that mirror the application and form that are active in JD Edwards EnterpriseOne 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 JD Edwards Autopilot script Application and Form commands match the application and form that are active. If you do not script the Application Interconnect command, you cannot continue scripting because the Form command line in the Script pane does not match the form and application that are active.

Remember, JD Edwards 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 select the Create Work With Batch Versions commands option in the command pane when you select a UBE. JD Edwards Autopilot launches the UBE and then automatically writes a series of commands to the script, including an Application Interconnect to the Batch Versions program.

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

When you select the Press Custom Button option to script an Application Interconnect command, you *initiate* the exit to a new application or form. JD Edwards 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 need to exit to a new application. You select the Press Toolbar Button option in the Command menu. When you select the Press Custom Button option, a tree node expands.

You select Form or Row, and then, by clicking one or the other, select 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 select a form or row selection in JD Edwards Autopilot, new lists appear in the command pane. You select an application and form, and then click the Insert button. JD Edwards Autopilot runs the form or row selection and interconnects to the application that you chose.

You can close the interconnected application and return to the previous form. In that case, you must write another Application Interconnect command and Form command in JD Edwards Autopilot to ensure that the command lines in the Script pane match the application and form that are active.

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

See Also

Chapter 4, “Understanding Context Scripting,” Form Command , page 34

Chapter 5, “Writing Scripts,” Setting the Context as a Form , page 50

Processing Options Command

You can use JD Edwards Autopilot to set processing options for interactive versions of applications that you run. During playback, JD Edwards 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 application. To do so, you select Application in the Command menu, select an application and menu item, and then select the Processing options only option in the command pane.

Note. JD Edwards 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, JD Edwards Autopilot cannot distinguish between them during script execution. JD Edwards 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 selected application version. With the play back while scripting feature activated, you can view the Processing Options form and its tabs.

JD Edwards Autopilot serializes the processing option IDs when you create the script. When you load the script for playback, JD Edwards Autopilot finds the matching processing option IDs in JD Edwards EnterpriseOne and displays processing option text that is consistent with the release for which you play the script.

If a new release changes the processing option ID and the text, JD Edwards 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 JD Edwards Autopilot, you select from both the Application list and the Menu list in the command pane. The selection from the Menu list specifies the form and version that appears in JD Edwards EnterpriseOne when you click the Insert button. The Form command line appears automatically in the Script pane whenever you select an application and form from these lists and click the Insert button.

Scripting commands in JD Edwards Autopilot requires that the Form command line in the Script pane mirror the form that is active. You can verify that the two mirrors one another by selecting a form from the Next Form list in the command pane, by selecting Form in the Command menu, or by clicking the Form button on the toolbar.

Next Form List

The Next Form list ensures that the Form command line in the Script pane matches the active form. For example, suppose that you script clicking the Add button in a form, such as the Work With Addresses form, to access another form, such as the Address Book Revision form. To do so, you select the Press Toolbar Button option in the Command menu, you select Standard Button in the command pane, and you select Add from the tree. In the Next Form list, you can select the Address Book Revisions form. When you click Insert, JD Edwards Autopilot inserts the Form command {Address Book Revision}.

Form List

You might not know which form appears next in the software. Suppose that 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 select Unknown/None from the Next Form list.

However, when you select 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 select Form in the Command menu or click the Form button on the toolbar. In the command pane, a Form list displays the names of the forms that are included in the selected application. In this case, you select 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 JD Edwards Autopilot window.

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 selecting options–can take place. You begin scripting the Header command by selecting the Set Header Control Value option 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 select to display hidden controls by selecting Tools, Options, Configure, and then selecting the Display Hidden Edit Controls option under 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 select Control Properties. The system displays the Control Properties form. This form includes four sections: Control Description, Parent Application, Edit/Display Properties, and Options. To exit the form, click Cancel.

Note. After you select a header control, you can select 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, JD Edwards Autopilot inserts two command lines in the script. The context command line is Header. However, by selecting a control, a source of input, and the value for the input, you write an additional command. This command is the Type To action command, 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 clicking grid buttons and entering inputs to grid columns–can take place. You begin scripting the Grid Column command by selecting the Set Grid Cell Value option 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 selecting Tools, Options, Configure, and then selecting the Display Hidden Grid Columns option under 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 select the Control Properties form. The system displays the Control Properties form, which includes four sections: Grid, Column, Column Edit/Display Properties, and Column Properties.

Note. After you select a grid column, you make additional command pane selections, including a source of input for the control and the value of the input. When you click the Insert button, JD Edwards Autopilot inserts two command lines in the script. The context command line appears with the words *Detail Information.* By selecting a grid column, a source of input, and a value of the input, you write an additional command. This command is the Type To action command, 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 clicking the Find button–can take place. You begin scripting the QBE command by selecting the Set QBE Cell Value option in the Command menu. You then select a grid column where you want to type inputs.

Note. After you select a grid column, you make additional command pane selections, including a source of input for the control, and the value of the input. When you click the Insert button, JD Edwards Autopilot inserts two command lines in the script. The context command line appears containing the words *QBE Information.* By selecting a grid column, a source of input, and a value of the input, you write an additional command. This command is the Type To action command, 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.

CHAPTER 5

Writing Scripts

This chapter provides an overview of creating a script and discusses how to:

- Create a script from event capture.
- Write a script using context commands.

Understanding How to Create a Script

You can create scripts with Oracle's JD Edwards EnterpriseOne Autopilot by using either manual or automatic script generation. When you generate a script manually, you create the script by selecting commands from the Command menu. The commands you create will execute sequentially when the script runs. To create a script automatically, you use event capture to generate a script that is based on the actions you perform as you use the JD Edwards EnterpriseOne software.

Creating a Script from Event Capture

This section provides an overview of creating scripts from event capture and discusses how to use the Create a Script from Capture option.

Note. Event Capture works only with JD Edwards EnterpriseOne Windows-based applications. Starting with the 8.11 release, JD Edwards EnterpriseOne applications are exclusively web-based. Consequently, you can use Event Capture with 8.10 and prior releases, but not with 8.11 and subsequent releases.

Understanding How to Create a Script from Event Capture

You can use JD Edwards Autopilot to translate events in the software into a JD Edwards Autopilot script. Using the Create a Script from Capture option in JD Edwards Autopilot enables 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 a JD Edwards 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 capture, you create a script framework, not a finished script. JD Edwards Autopilot captures all the events in the stream. You might not need all those events. You will have to decide how to modify the 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 a JD Edwards Autopilot script and then email the script to your support organization.

Note. When you use the Create a Script from Capture option to create a script, it is recommended that you use the fast paths instead of the menus when calling an application.

Creating a Script from Event Capture

To create a script from event capture:

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

   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 JD Edwards 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.

   JD Edwards Autopilot captures and records every event. Ensure that you are performing actions deliberately in order to create the most accurate script. For example, if you click the OK button twice, JD Edwards 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 JD Edwards Autopilot script view pane.
9. From the File menu, select Save to save your script.
10. Modify the script as needed.

Writing a Script Using Context Commands

This section provides an overview of writing a script using context commands and discusses how to:

- Set the Context as a UBE
- Launch a UBE
- Set the Context as an Application
- Launch a UBE from a Menu
- Launch a UBE from a Report Menu
- Launch a UBE from a Row Menu
- Launch a UBE That Is Automatically Submitted
- Launch a UBE from Another UBE
- Submit a UBE

- Select Data for a UBE
- Set UBE Processing Options
- Print a UBE
- Set the Context as an Interconnected Application
- Set the Context as a Processing Option
- Define Unwanted Windows
- Set the Context as a Form
- Script the Form Command Using the Command Menu
- Set the Context as a Grid Column
- Set the Context as a Header
- Set the Context as a QBE Line

Understanding How to Write Scripts Using Context Commands

You can begin scripting context commands in one of three ways: by selecting a command from the Command menu, by clicking a hot key on the keyboard, or by clicking a toolbar 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, JD Edwards 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:

- Select a general context, such as an interactive application or UBE, by clicking the Command menu, a hot key, or a toolbar 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:

- Select 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.
- Select a source of input for the specific context.
- Select a value to be input in the specific context.

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, JD Edwards EnterpriseOne software launches any subsequent UBEs and then blindly submits them without any further intervention by JD Edwards 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 JD Edwards Autopilot submits the UBE, except when a menu is hard-coded to automatically submit it.

If you select 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 enables 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 select the way to print the UBEs that you submit. You can instruct JD Edwards Autopilot to wait for the UBE to print before resuming running the script, or you can send the UBE to print, but tell JD Edwards 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 JD Edwards Autopilot to launch a UBE from a variety of contexts. You might begin the 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 select to launch a UBE that is automatically submitted. Finally, you can launch a UBE that in turn launches one or more additional UBEs.

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 the 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 select the menu item that is associated with the version and processing options that you want to test.

Access JD Edwards Autopilot.

1. Select Application from the Command menu.
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. Click the Insert button.

JD Edwards Autopilot inserts the Application, Fast Path, and Form command lines into the Script pane. In the play back while scripting mode, JD Edwards Autopilot launches the specified version of an interactive application. The form appears on the screen with the JD Edwards Autopilot window, and you can navigate between the two.

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 select both of the options in the command pane: Execute FASTPATH and Create Work With Batch Versions commands. The first option establishes the Fast Path JD Edwards Autopilot uses to access the UBE; the second option commands JD Edwards 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 clear this option, you must manually create the command to submit the UBE from the Version Prompting form.

To launch a UBE from a menu:

1. From the Command menu, select UBE.
2. In the command pane, make a selection from each of the available lists:
   - UBE
   - Menu Item
   - Version
3. Select 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, JD Edwards 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 select a report. You select the UBE without the Execute FASTPATH option, and the Menu Item list containing Fast Paths to the UBEs disappears. You select 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, select Application.
2. In the command pane, select options from these lists:
   - Application
   - Menu
3. Click the Insert button.
4. From the Command menu, select Press Toolbar Button.

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

5. In the Button list, expand Custom Button, and then expand Report.

6. Select a report.
7. Click the Insert button.
8. From the Command menu, select UBE.
9. In the command pane, select a UBE.

   Do not select the Execute FASTPATH option. Clear it, if it is selected.

   Note. When you clear the Execute FASTPATH option, the Menu Item list disappears. Do not clear this option until you have chosen the UBE. If you clear the option before you select the UBE, JD Edwards Autopilot turns the option on again after you have chosen the UBE, and you must clear it again.

10. Select a version from the Version list.
11. Select 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, JD Edwards 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 select the UBE command. You click neither of the command pane options. The Menu Item list disappears, and you select a version. If you clear the Batch Versions option, JD Edwards Autopilot blindly submits the UBE.

To launch a UBE from a Row menu:

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

   Important! Do not select options from the Next Form list. If you select these options and insert the command, JD Edwards 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 these options:
    - Click by row number
    - Single click
12. In the Source of Row Number list, select a value source.
13. In the value selection list, enter a row number or select 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. Select a Row selection.

    Important! 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, select a UBE from the UBE list.

    Important! Deselect both the Execute FASTPATH and Create Work With Batch Versions commands options.

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

    JD Edwards 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 select the UBE, JD Edwards Autopilot disables both of the options and the Version list disappears. You select from the Menu Item list and click the Insert button, and then JD Edwards 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, select a UBE that will be blindly submitted from the UBE list.

   When you select the UBE in this scenario, JD Edwards Autopilot disables both options and the Versions list disappears.
3. Select 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, JD Edwards EnterpriseOne software automatically launches any UBEs that are called by the first one and blindly submits them. You do not select versions or printing options, or set processing options. Without further direction from the JD Edwards Autopilot script, JD Edwards EnterpriseOne 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, select a UBE from the UBE list.

   If the UBE is coded to launch another UBE, JD Edwards Autopilot disables the Execute FASTPATH option and removes the Menu Item list from the command pane.
3. Select a version from the Version list.
4. Deselect 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 select this option, JD Edwards 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 click 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, select Press Toolbar Button.
3. In the Button list in the command pane, click Standard Button.
4. Select 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 JD Edwards Autopilot, which you access either by clicking UBE Selection in the command menu or by clicking the CDA button on the toolbar. This feature enables 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, JD Edwards 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 the 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 JD Edwards 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 the 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, JD Edwards 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 enable you to set processing options. You use JD Edwards Autopilot to set the processing options for the UBE after you have selected the data that you want to appear in the report. You complete the UBE submission process by sending the report to the printer.

Important! 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, select UBE.
2. In the command pane, select options from these lists:
   - UBE
   - Menu Item
   - Version
3. Select both the Execute FASTPATH and Create Work With Batch Versions commands options.
4. Click the Insert button.
5. From the Command menu, select Check box/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, select Press Toolbar Button.
10. From the Button list, select Standard Button, and then Submit.
11. Click the Insert button.
12. From the Command menu, select UBE Selection.
13. In the command pane, complete these fields:
    - Line Number
    - Operator
    - Left Operand
    - Comparison
    - Right Operand

14. If you enter a variable in the Right Operand list, click one of these 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 select UBE Processing Options in the command menu, and then select options from the command pane.

To set UBE processing options:

1. Submit a UBE.
2. From the Command menu, select UBE Processing Options.
3. In the Processing Options list of the command pane, click the node of a processing options tab.
4. Select a processing option.
5. Select 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 select Variable as the value source, JD Edwards 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.

Printing a UBE

Clicking UBE Print in the Command menu produces three options in the command pane. If you select the option Wait for UBE to complete before continuing, JD Edwards Autopilot submits the UBE to the printer and waits for it to complete before it resumes the script playback. If you do not select this option, JD Edwards 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 JD Edwards 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, enables 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 select this option only if you launched the 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 JD Edwards Autopilot blindly submits the UBE, from the Command menu, select UBE Print.
2. In the command pane, click one or more of these 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 the 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 click a standard button, such as Add, you might exit from one JD Edwards EnterpriseOne 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, select the menu item Work With Customer Ledger Inquiry, then script clicking the Add button and Unknown/None from the Next Form list, JD Edwards EnterpriseOne 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 clicking the Add button, you exited to another application. You cannot continue scripting until the Application and Form command lines in JD Edwards Autopilot mirror the application and form that are active.

Using the Application Interconnect command, you can ensure that the 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. Select About JD Edwards.
3. Note the application ID and form name and click OK.
4. In JD Edwards 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, select from the lists that appear:
   - Application (select the application that is active)
   - Menu (select the form that is active)
7. Click the Insert button.

   JD Edwards 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.

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 select 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 enables you to select processing options from lists in the command pane.

To set the context as a processing option:

1. From the Command menu, select 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, select 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. Select a processing option.

   JD Edwards Autopilot populates the Source list in the command pane with two sources of input: literal and variable.
8. Select 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, JD Edwards Autopilot populates the Variables list with the names of any variables whose values you have set.
10. Click the Insert button.

    JD Edwards Autopilot enters to the Script pane a command line that summarizes the processing options you have chosen.

    Note. With the play back while scripting feature turned on, when you insert a processing option value in JD Edwards Autopilot, the tool 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 select the Processing options only option in the command pane.

See Also

Chapter 6, “Scripting Actions,” Using a Variable as a Source of Input , page 65

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 JD Edwards Autopilot scripts and their unexpected appearance might cause your scripts to fail. To prevent script failure, you can specify what you want JD Edwards 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 JD Edwards Autopilot to play when a particular window or message box appears.

To define unwanted windows:

1. From the Tools menu, select 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 If This Window Is Found field:

   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. Select one of these options:
   - Then Click Button

     Select 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

     Select 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

     Select this option to play the designated subscript if JD Edwards Autopilot finds the form that you specified in the If This Window Is Found field.
   - Fail Script

     Select 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, JD Edwards 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 JD Edwards 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, select a command that enables you to switch forms, for example, Press Toolbar Button.
3. In the Button list, click a button-clicking 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.

Scripting the Form Command Using the Command Menu

Access JD Edwards Autopilot.

To script the Form command using the command menu:

1. Insert an application into a script.
2. From the Command menu, select 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 toolbar.

   ---
3. Select 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 select 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 select a specific column to further refine the context.

To set the context as a grid column:

1. From the Command menu, select Set Grid Cell Value.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. In the Grid Column list, select a grid column in which you want to enter data.
4. Select a source of input from the Source of Input list.
5. In the value selection list, select or enter a value and click the Insert button.

When you click the Insert button, JD Edwards 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 select 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 select a specific control to further refine the context.

To set the context as a header:

1. In the Command menu of the JD Edwards Autopilot window, click Set Header Control Value.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. In the Header list, select a control to which you want to input data.
4. Select a source of input from the Source of Input list.
5. In the value selection list, select or enter a value and then click the Insert button.

   When you click the Insert button, JD Edwards 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 select 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 select 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, select Set QBE Cell Value.
3. In the Form/Subform Hierarchy list, select the appropriate form or subform.
4. In the Grid Column list, select a grid column in which you want to enter data.
5. Select a source of input from the Source of Input list.
6. In the value selection list, select or enter a value and click the Insert button.

   When you click the Insert button, JD Edwards Autopilot writes both a Grid (or Detail Information) context command line and a Type To action command line in the Script pane.

CHAPTER 6

Scripting Actions

This chapter provides an overview of scripting actions and discusses how to:

- Use the Type To command.
- Script the Type To command.
- Use the Select Grid Row command.
- Script the Select Grid Row command.
- Use the Press Toolbar Button command.
- Script the Press Toolbar Button command.
- Use the Press Push Button command.
- Script the Press Push Button command.
- Use the Select ComboBox Item command.
- Script the Select ComboBox Item command.
- Use the Build Tree Path command.
- Script the Build Tree Path command.
- Use the Database Validation command.
- Script the Database Validation command.
- Associate a validation.
- Execute a validation.
- Use the command line.
- Script a Command Line command.

Understanding Scripting Actions

Action commands within Oracle's JD Edwards EnterpriseOne Autopilot designate actions that a script carries out–for example, when users click buttons, select options, and enter data–within a context such as an application or form. Action commands require a context. They are essential because they specify the unique steps that the system takes within the context. For example, you must write action commands to create a transition between forms; to enter data in header controls, grid columns, or Query By Example (QBE) lines; to select lines in a grid; to perform database queries and updates; and so on.

Action commands also enable you to use a script to access a non-JD Edwards EnterpriseOne application, such as Microsoft Excel. You do this by sending a message to the system from a command line in JD Edwards Autopilot. You can also use the Command Line command to capture screens and store the images in a file for later use.

In addition, you can use action commands to enhance existing scripts. For example, you can write an action command to include a previously-created script within another script. For instance, in a script that requires entry of dates and then tests functions, you might include a standalone script that tests the date entry.

When you play back a created script, you can use action commands to configure the playback. For example, you can insert a Wait command in the script. This command instructs JD Edwards Autopilot to wait the specified length of time at a particular point in the script before it proceeds with playback. In addition, you can insert comments in the script to document the goal of the testing or to describe what occurs at a particular point during playback.

After you script the entry of form data, you can verify that JD Edwards Autopilot has entered the information in the specified database. The Database Validation command enables you to do that.

Action commands enable you to:

- Build scripts that test a particular set of processes.
- Test whether data is properly entered in the database.
- Modify and add comments to existing scripts.
- Configure the way that scripts run.
- Use applications external to JD Edwards EnterpriseOne to run a script or perform other tasks.

Using the Type To Command

This section provides an overview of the Type To command and discusses how to:

- Use the Header Control and Grid Column lists.
- Use the Source of Input list.
- Use literal values.
- Use a valid values list.
- Use variables.
- Describe variable scope.
- Use global variables.
- Use local variables.
- Use the value selection list.

Understanding 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 the Press Toolbar Button command–there is no item on the Command menu or toolbar button that represents the Type To command. To write it, you use lists to specify the context as a header, grid, or QBE line; you specify a header control or grid column; you designate a source of input for the control or column; and you select a value to input into the control or column. The value can be a literal or you can derive it from a variable, a list of valid values, a user defined command (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 JD Edwards Autopilot window.

Using the Header Control and Grid Column Lists

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 to script an input.

Using the Source of Input List

After you select a header control or grid column, use the Source of Input list to select one of these sources from which to retrieve a value to input into the header control or grid column:

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

Using Literal Values

When you select the Literal Value option 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 the exact text that JD Edwards Autopilot enters in the header control of a form when the script runs.

Using a Valid Values List

When you select a literal value as a source of input, JD Edwards Autopilot assigns only one value to a header control, grid column, or QBE line in a form. If you select a list of valid values as a source of input, you can create a text or numeric file that contains 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 selecting a database and querying it for values to include in the list.

You can use a list of valid values as an input source to run a script multiple times and, each time, enter a different value in a specified header control, grid column, or QBE line. As JD Edwards Autopilot loops through the script, the value that it enters in the control or column changes to reflect the values in the list. Alternatively, you might run the script once but enter five different values in a grid column. You can do this by creating a single list of valid values that contains five items.

If you exit a script and exit JD Edwards Autopilot, and then you open the script again, JD Edwards Autopilot resets the list of valid values. That way, when you play back the script, the first value that you entered in the list appears first. If you exit a script without exiting JD Edwards Autopilot and then open the script again, JD Edwards Autopilot uses the value that is next in order on the list when you exit the script.

Using Variables

You can select a value, store it, and then use it at a subsequent place in the script. You can also 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 retrieve from a list of valid values, header control, grid column, another variable, or from the literal input.

The Command pane of the JD Edwards Autopilot window displays these 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 selecting 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 JD Edwards Autopilot uses even if you do not set a value for the variable.

- Existing Variable list.

  JD Edwards Autopilot displays the names of any existing variables that you have declared in the script.

- Source of Value list.

  You select a source of value for the variable, such as a literal value, a list of valid values, a header control, a grid cell, or another variable. In addition, you can select 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 select the object that contains the value that you store in the variable. For example, if you select a header control as the source of value, you use the value selection list to select the specific control that contains the value. You can also obtain variable values from lists of valid values, from variables to which you have previously assigned a value, or from JD Edwards EnterpriseOne sources, such as error and warning messages or grid row counts.

This table explains other key terms that are related to variables in JD Edwards Autopilot scripts:

Term	Description
Variable scope	The range of commands within a script in which the value for a variable can be used.

Term	Description
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 JD Edwards 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 JD Edwards EnterpriseOne 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 list of valid values.
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.

Describing Variable Scope

The term *variable scope* refers to how broadly you can use the value of the variable within a script. You create a node each time that 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.

Using 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.

Using Local Variables

You can use the value of a local variable only within a portion of the script, specifically the node to which you attach 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 remains 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 enables you to pass a value to header controls, grid columns, and QBE lines throughout a single, standalone script. JD Edwards Autopilot also enables 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.

Default Values for Variables

You can create a script that you can use both in standalone mode and with other scripts. To do so, you assign a default value to the variable. For example, you might create script B that links to script A, which passes along a batch number value. Suppose, however, that you want to use script B by itself. If you set a default value in script B, JD Edwards Autopilot uses that value when you use script B in standalone mode.

You can also assign a default value to a variable that you do not declare as external. JD Edwards 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 enable 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 JD Edwards 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 enable the script to end if it does not meet the condition that appears in the statement.

You can write a conditional statement to ensure that a script tests an application even if the script does not meet the conditions that you expect to exist. For example, suppose that you test making revisions to an existing Address Book number. If the Address Book number exists, then JD Edwards 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, the script fails unless you write a conditional statement stipulating that if the Address Book number does not exist, JD Edwards Autopilot adds a form and runs commands to create a new Address Book entry.

A different conditional statement might test the converse. If the Address Book number does not equal the number that JD Edwards Autopilot returns to the QBE line of the Work With Addresses form, the Address Book number does not exist. If that condition is met, JD Edwards Autopilot clicks Add and creates a new entry. If the Address Book number does exist, JD Edwards 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 the conditional statement on these two variables.

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

Variable Addition

Variable addition enables you to change the value of a variable. One way that you can use this option is to scroll through a grid from top to bottom. For example, you can write a command that adds 1 to the row number of the grid each time JD Edwards 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, JD Edwards Autopilot scrolls through the entire grid, one line at a time, from top to bottom.

Variable Subtraction

Variable addition enables you to change the value of a variable. One way that you can use this option is to scroll through a grid. 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 1 from the row number of the grid each time JD Edwards Autopilot plays back the node of the script.

Variable Concatenation

Variable concatenation changes the value of an existing variable by concatenating some value to its current value. For example, if a variable has the value of 10, you could change it to 1025 by concatenating the value of 25 to the original value of 10.

System Variables

System variables obtain their values from JD Edwards EnterpriseOne, rather than from the information that you enter in JD Edwards 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.

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

System Variable	Description	Possible Uses
::ERRORS	JD Edwards Autopilot records the number of JD Edwards EnterpriseOne error messages that occur during script playback.	Use in conjunction with a conditional statement. For example, set a condition specifying that if the number of error messages returned is greater than 0, then JD Edwards Autopilot should run the Exit JD Edwards command.

System Variable	Description	Possible Uses
::WARNINGS	JD Edwards Autopilot records the number of JD Edwards EnterpriseOne warning messages that occur during script playback.	Use in conjunction with a conditional statement. For example, set a condition that specifies that if JD Edwards EnterpriseOne sends a warning message when the OK button is clicked, then JD Edwards Autopilot should click the OK button twice.
::FORMGRIDROWCOUNT	JD Edwards Autopilot records the number of completed rows in a grid.	Set a repeat count for a node to ensure that JD Edwards Autopilot accesses each line in the grid during script playback.

Note. The value of the Grid Row Count system variable is the number of rows currently loaded in the JD Edwards EnterpriseOne grid, 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.

Valid Values Count

If you select the Valid Values Count option as an input source, you also select a created valid value from the value selection list. The list can be stored either on your local drive or on a server. JD Edwards Autopilot counts the number of items in the list and stores that number as a variable. You can 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 that JD Edwards Autopilot repeats the script node matches the number of items in the valid values list, and you can write a command for JD Edwards Autopilot to enter the values from the valid values list to a cell in each line of the grid.

Variable Watch List

The Watch list, which you can select from the JD Edwards Autopilot View menu, is a separate window that displays the values of variables in the script during playback. Because the Watch list window is independent of other JD Edwards Autopilot windows, 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 that JD Edwards Autopilot sets the value of a variable, it adds the variable name and its value to the list. If the variable value that JD Edwards Autopilot enters in a header control or grid column is invalid and script playback stops, JD Edwards Autopilot stops adding values to the Watch list. Each time that you stop the script and replay it, JD Edwards Autopilot clears the Watch list.

Validation Success

The Validation Success variable enables you to quickly verify the existence of data that you expect as a result of running a script. After you declare a validation, you select Validation Success as an input source, and use the value selection list to select the name of the validation to verify.

After you associate and run the validation and run the script, JD Edwards Autopilot uses the Watch list to display the name and value of the validation success variable that you created. A value of 1 indicates that the validation was successful, and 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, JD Edwards Autopilot returns a value of 2 when playback ends.

UDC Visual Assist Value

You can also use a value from a UDC visual assist as an input source. Selecting the UDC Visual Assist Value option from the Source of Input list populates the UDC Visual Assist Value list with the same UDC values that appear when you click the Visual Assist button on the form. During playback, JD Edwards Autopilot expands and interprets this command by opening the Visual Assist form and selecting the specified value.

The UDC Visual Assist Value option 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, suppose that you run a company master search. In JD Edwards EnterpriseOne, when you click the Visual Assist button for company master search, a new application appears.

You create this form interconnect in the JD Edwards Autopilot script by selecting the Form Interconnect Visual Assist option in the Source of Input list. When you insert a Form Interconnect Visual Assist command into a script, JD Edwards Autopilot automatically inserts additional commands to perform the Application Interconnect and Confirm Form commands needed to access that visual assist. However, you must script the additional commands that are needed to select a value from the visual assist on the form.

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

Clear Source of Input

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

See Also

Chapter 6, “Scripting Actions,” Changing the Scope of a Variable , page 66

Chapter 7, “Working with the Script Pane,” Using Nodes, page 111

Using the Value Selection List

After selecting 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 these methods to supply the value:

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

Remember, the caption of the value selection list changes to reflect the option that you select in the Source of Input list.

Scripting the Type To Command

This section provides an overview of scripting the Type To command and discusses how to:

- Use the Header Control or Grid Column list.
- Use a literal value as a source of input.
- Create a list of literal values.
- Create a valid values list from a simple database query.
- Use valid values as a source of input.
- Update the repeat count in a node.
- Use a variable as a source of input.
- Declare a variable.
- Change the scope of a variable.
- Set the value of a variable.
- Use the value of a variable as a source of input.
- Update the value of an existing variable.
- Set conditional statements.
- Add a value to a variable.
- Subtract a value from a variable.
- Concatenate a variable.
- Create a variable to confirm validation success.
- Create a variable to store a valid values list count.
- Use a UDC visual assist value as a source of input.
- Use a form interconnect visual assist as a source of input.
- Clear an input from a header control or grid column.
- Use the value selection list.
- Assign a literal value.
- Assign a valid values list value.
- Assign a variable value.
- Assign a form interconnect visual assist value.
- Script the Type To command.
- Type data in a header control.
- Select options in a header.
- Type data in a grid cell.
- Assign a UDC visual assist value.
- Type data in a QBE line.

Understanding Scripting the Type To Command

You use 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, JD Edwards 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 select Set Header Control Value, Set Grid Cell Value, or Set QBE Cell Value in the Command menu. When you select 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 type data.

To use a Header Control or Grid Column list:

1. On the Command menu, select Set Header Control Value, Set Grid Cell Value, or Set QBE Cell Value.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. In the Header Control or Grid Column list of the command pane, select a header control or grid column in which to enter data.
4. Click the Insert button.

Using the Source of Input List

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

After you select a source of input, you select a value from the value selection list, which is 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 Literal Value field.

To use a literal value as a source of input:

1. In the command pane of the JD Edwards Autopilot window, select the Literal option from the Source of Input list.
2. In the Literal Value field, type an input as you would 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 of these.
3. Click the Insert button.

Using 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 to enter values multiple times or 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, JD Edwards Autopilot loops through the node five times and inserts a different item from the list each time. Conversely, you can enter *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, JD Edwards Autopilot inputs a different value from the list.

If you close the script, leave JD Edwards Autopilot open, and then open and rerun the script, JD Edwards 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 any script.

You must create a valid values list before you select it as a source of input for a form. To create a valid values list, select the Generate Valid Values List command from the Tools menu, enter values in the list, then name it and save it.

Creating a List of Literal Values

A list of literal values is a list that contains values that you select. 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. On the menu bar of JD Edwards Autopilot, click Tools.
2. Select Generate Valid Values List.
3. On the Select Data File Type form, select List of Literal Values and click Next.
4. On the Enter File Name & Date form, type a file name in the File Name field.
5. If you type the name of an existing file, the values in it automatically populate the list.
6. Type one valid, literal value per line, pressing ENTER after each value.
7. 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 JD Edwards Autopilot retrieves from the database and includes in the list, based on the table and column that you select. You can limit the number of records in the list, and you can specify the method that JD Edwards 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. On the Tools menu, select Generate Valid Values List.
2. On the Select Data File Type form, select Simple Database Query and click Next.
3. On the Select Table form, double-click a table.
4. On the Column Name form, select 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.

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

9. Click Finish.

Using Valid Values As a Source of Input

After you create a valid values list, you return to the Command menu and click the context for writing commands. You select a header control or grid column, and then select the Valid Values List option from the Source of Input list. When you select the Valid Values List option, 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. On the Header Control or Grid Column list in the command pane of the JD Edwards Autopilot window, select a header control or grid column.
2. Select the Valid Values List option in the Source of Input list.
3. Select the name of a valid values list that you have created.
4. Click the Insert button.

Updating the Repeat Count in a Node

To use all the values in a valid values list as inputs for a header control, grid column, or QBE line, change the repeat count in the Form command node of the script to match the number of items in the list. This ensures that JD Edwards Autopilot successively types each value in the control, grid column, or QBE line during playback until it exhausts the list.

To update the repeat count in a node:

1. In the Script pane of the JD Edwards Autopilot window, click the Form line for the node in which you scripted the list of valid values.
2. In the Repeat Count list in the command pane, type the number of times that you want the node to run.
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 (that is, 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 declare the variable, you can set it (assign a value to it) and store the value for later use in the script in the variable. 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 name it to indicate the place where a value that you set can be stored. You can insert the Declare variable command at any point in the script. Where you declare the variable 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. On the Command menu, select 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. Select an initial value for the variable, if desired.
4. Click the Insert button.

Changing the Scope of a Variable

The scope of a variable is the context within which you can use an assigned value. 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 select 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 select.

As you move the mouse pointer, an arrow appears over the Declare command line. An upward-pointing arrow indicates that the command line that you are dragging will be placed above the line that is highlighted when you release the mouse button. A downward-pointing arrow 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 JD Edwards Autopilot window, click the Declare command line.
2. Drag the Declare command line to another context by pressing 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 over 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 various points in the script that are determined by the scope of the variable.

To set the value of a variable:

1. On the Command menu, select Variables.
2. In the Existing Variable list of the command pane, select the name of the declared variable to which you want to assign a value.
3. In the Source of Value list, select one of these sources for the value:
   - Literal Value
   - Valid Values List
   - Variable
   - Header Control Data
   - Grid Cell Data
4. Depending on the source, perform one of these tasks:
   - If you assign a literal value, type that value 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 select Grid Cell Data as the source of value, JD Edwards 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, JD Edwards Autopilot stores the value from the row that you specified.

5. Click the Insert button.

   JD Edwards Autopilot sets the value that you select and stores it in the declared variable.

Using the Value of a Variable As a Source of Input

After you have declared a variable and set its value, you can use the value as a source of input. The scope that you establish 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 to input the value of the variable.
2. In the Source of Input list, select Variable.
3. In the Variable list, select the name of the declared variable for which you set a value when you began the scripting process.
4. Click the Insert button.

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

Updating the Value of an Existing Variable

Like any other JD Edwards Autopilot script command, you can change Set Value commands if it is necessary to do so.

To update the value of an existing variable:

1. In the Script pane of the JD Edwards Autopilot window, 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, select the name of the variable to update.
3. Select an option from the Source of Value list.
4. Select or enter a value.
5. Click the Update button.

Setting Conditional Statements

To use JD Edwards 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 enable you to write a conditional If/Then statement, which also includes an Else statement. JD Edwards Autopilot populates two of the lists with the names of the variables that you have declared. You write the left and right side of the conditional statement by selecting 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, for numeric comparison. When you select this option, JD Edwards Autopilot converts the text variables to numeric values before it compares them. To write a conditional statement that uses a string, rather than a numeric value, select the Is Not In option 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 to have JD Edwards Autopilot run a series of commands only if the first part of the conditional statement is not true.

To set a conditional statement:

1. In the JD Edwards Autopilot window, 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. On the Command menu, select `If <var > == <var >`.
3. In the command pane, select options from these lists:
   - Left Variable
   - Operator
   - Right Variable
   - Right Literal
4. (Optional) If the values of the variables are numeric, select the Numeric check box.
5. Click the Insert button.

   JD Edwards 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. 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 you can add a literal value. You can add a value to a variable regardless of whether you have merely declared the variable or both declared the variable and set its value. However, if you declare a variable and plan to set its value by adding a value later in the script, enter a default value so that JD Edwards Autopilot has an original value to supplement by the addition. To add the value of one variable to another variable, you must first declare and set the value of the variable that contains the value that you want to add.

To add a value to a variable:

1. On the Command menu, select Variables.
2. Select a variable from the Existing Variable list.
3. Select one of these options from the Source of Value list:
   - Add Variable
   - Add Literal
4. To add a variable value, select 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 declare the variable.

To subtract a value from a variable:

1. On the Command menu, select Variables.
2. Select a variable from the Existing Variable list.
3. Select one of these from the Source of Value list:
   - Subtract Variable
   - Subtract Literal
4. To subtract a variable value, select 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 multiple variables. You can construct a concatenated variable from other variables or from literal values. This table illustrates the principle of variable concatenation:

Variable Name	Variable Value
X	1

Variable Name	Variable Value
Y	1
X concatenated with Y	11

Note. Before concatenating a variable, declare at least one variable, and either set its value or enter a default value.

To concatenate a variable:

1. On the Command menu, select Variables.
2. Select a variable from the Existing Variable list.
3. Select one of these from the Source of Value list:
   - Concatenate Literal
   - Concatenate Variable
4. To concatenate a literal value, enter a literal value in the Literal Value field.
5. To concatenate a variable, select a variable from the Variable list.
6. Click the Insert button.

   JD Edwards Autopilot creates a concatenated value for the variable selected 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 select Validation Success as a source of value. When you select the Validation Success option, JD Edwards Autopilot displays the names of the declared validations in the value selection list. When you run the validation, JD Edwards Autopilot sets the value of the variable to *1* if the validation succeeds. If the validation fails, JD Edwards Autopilot sets the value of the variable to *0*. If you declare the validation and assign values but do not run it, JD Edwards Autopilot sets the value of the validation variable to *2*.

Select the Watch List option on the JD Edwards Autopilot View menu to determine whether the data validation is successful following playback. The Watch list displays the name of the validation variable in the Variable column. If the validation is successful, JD Edwards Autopilot displays *1* in the Value column.

Note. Before creating a variable to confirm validation success, declare and associate a validation. You must run the validation to have JD Edwards Autopilot indicate success or failure by returning a value of 1 or 0.

See Chapter 6, “Scripting Actions,” Using the Database Validation Command, page 97.

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

If the validation runs successfully, JD Edwards Autopilot displays a value of *1* for the validation variable in the Watch list.

Creating a Variable to Store a Valid Values List Count

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

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

1. On the Command menu, select Variables.
2. Enter the name of a variable in the New Variable list, or select the name of a variable from the Existing Variable list.
3. Select the Valid Values Count option from the Source of Value list.
4. Select 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 to set the repeat count for a node, select the node in the Script pane that contains the valid values list used to determine the value of the Valid Values Count variable.
7. In the Define Repeat Count list, select Variable.
8. In the Repeat Count list, select the variable that stores the valid values count value.
9. Click the Update button.

   JD Edwards Autopilot updates the repeat count value, which now matches the item number value in the valid values list. When you run the script, JD Edwards 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 select the UDC Visual Assist Value option 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 button identifies the UDC visual assist, but JD Edwards EnterpriseOne software also identifies other visual assists in this way. The UDC Visual Assist Value options 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 an input source:

1. On the Command menu, select Set Header Control Value or Set Grid Cell Value.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. In the command pane of the JD Edwards Autopilot window, select 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 button for a control or column in a form. If the selection does not have a visual assist button, the UDC Visual Assist Value option does not appear in the Source of Input list.

4. Select the UDC Visual Assist Value option from the Source of Input list.
5. In the UDC Visual Assist Value list, select a UDC value.
6. Click the Insert button.

   JD Edwards Autopilot runs the code path 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 the current application and access a new one. If you have selected a header control or grid column that includes this type of visual assist–such as Address Book Master Search–JD Edwards Autopilot displays the Form Interconnect Visual Assist option in the Source of Input list. If you select this input source, JD Edwards Autopilot inserts a command to click the visual assist 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 select this option. After you access a new application and form, you can write the additional commands that you need as part of the script.

To use a form-interconnect visual assist as a source of input:

1. On the Command menu, select either Set Header Control Value or Set Grid Cell Value.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. In the command pane of the JD Edwards Autopilot window, select a header control or grid column from the Header Control or Grid Column list.
4. Select the Form Interconnect Visual Assist option from the Source of Input list.
5. Click the Insert button.

   JD Edwards Autopilot automatically writes a command to click the visual assist button in the active form, and it then also creates 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. JD Edwards Autopilot enables you to do this by selecting the Clear option from the Source of Input list in the command pane.

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

1. On the Command menu, select Set Header Control Value, Set Grid Cell Value, or Set QBE Cell Value.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. In the JD Edwards Autopilot command pane, select the name of a header control or grid column from the Header Control or Grid Column list.
4. In the Source of Input list, select Clear.
5. Click the Insert button.

   JD Edwards Autopilot clears the value from the selected header control or grid column.

See Also

Chapter 6, “Scripting Actions,” Using a Valid Values List , page 55

Using the Value Selection List

After you select a header control or grid column and an input source for it, you complete the Type To command by selecting a value from the value selection list. JD Edwards Autopilot displays this list when you select a source of input and provides a caption according to the source of input that you select. You can also delete an entry in a control in a header or a processing option form or a grid column. To do this, select Clear from the Source of Input list.

To use the value selection list:

1. To input a literal value in the header control or grid column, type that value in the unpopulated value selection list.
2. To input the values that you assigned to a valid values list or variable, select the name of the list or variable from the value selection list.
3. To input a UDC visual assist value, you select one from the value selection 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 JD Edwards Autopilot window, select a header control or grid column name from the Header Control or Grid Column list.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. Select the Literal Value option from the Source of Input list.
4. In the value selection list labeled Literal Value, input the value assigned to the control or grid column.
5. 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. JD Edwards Autopilot selects the first value in the list the first time that you run the script. If you run it more than once, JD Edwards Autopilot selects 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 JD Edwards Autopilot window, select a header control or grid column name from the Header Control or Grid Column list.
2. Select the Valid Values List option from the Source of Input list.
3. In the value selection list labeled Valid Values List, select the name of a valid values list that you created.
4. Click the Insert button.

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, JD Edwards Autopilot stores it and it is available for use at any point in the script.

To assign a variable value:

1. In the command pane of the JD Edwards Autopilot window, select a header control or grid column name from the Header Control or Grid Column list.
2. Select the Variable option from the Source of Input list.
3. In the value selection list labeled Variable, select the name of a declared variable to set a value.
4. Click the Insert button.

To assign a UDC visual assist value:

1. In the command pane of the JD Edwards Autopilot window, select a header control or grid column name from the Header Control or Grid Column list.
2. Select the UDC Visual Assist Value option from the Source of Input list.
3. In the value selection list labeled UDC Visual Assist Value, select the name of a UDC value.
4. Click the Insert button.

Assigning a Form Interconnect Visual Assist Value

JD Edwards Autopilot enables you to select a form interconnect visual assist value for header controls or grid columns in forms that contain a visual assist that accesses a different application. You do not use the value selection list when you select the Form Interconnect Visual Assist option as a source of input. Instead, after you access the other 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 JD Edwards Autopilot window, select a header control or grid column name from the Header Control or Grid Column list.
2. Select the Form Interconnect Visual Assist option from the Source of Input list.
3. Click the Insert button.

   JD Edwards Autopilot clicks the visual assist button on the selected header control or grid column, and it then writes an Application Interconnect command and a Form command to the Script pane.
4. On the Command menu, select Set QBE Cell Value.
5. In the command pane, select the name of a grid column from the Grid Column list.
6. Select an input source from the Source of Input list.
7. Enter or select a value from the value selection list.
8. Click the Insert button.
9. On the Command menu, select the Press Toolbar Button option.
10. In the command pane, select the Standard Button option.
11. Select Find.
12. Click the Insert button.

13. On the Command menu, select the Press Toolbar Button option.
14. In the command pane, select the Standard Button option.
15. Click Select.
16. Click the Insert button.

    JD Edwards Autopilot enters the value from the form interconnect visual assist in the header control of the selected form.
17. On the Command menu, select the Press Toolbar Button option.
18. In the command pane, select the Standard Button option.
19. Click Close or Cancel.
20. Click the Insert button.
21. On the Command menu, select Form.
22. In the command pane, select the name of the form that JD Edwards Autopilot exited by using the form interconnect visual assist.

    Note. This command confirms that JD Edwards Autopilot has 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 select from each of the three command pane lists, you click the Insert button to script the Type To command. JD Edwards Autopilot uses the selected information to write two command lines in the Script pane. One command line contains the context–header, grid, or QBE–and the repeat count for the node. The other contains the name of the selected header control or grid column; a symbol that indicates whether you selected a literal value, a valid values list, a variable, or a UDC visual assist value as the input source; and the assigned value.

Typing Data in a Header Control

You use the command pane lists to script inputs for the header portion of a form. Selecting the Set Header Control Value option in the Command menu establishes the header as the context in which you type data. The options that you select 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 select and the value that you type in it. JD Edwards Autopilot inserts subsequent Type To commands subordinate to the node.

To type data in a header control:

1. On the Command menu, select the Set Header Control Value option.
2. Select the name of a control from the Header Control list.

   When JD Edwards EnterpriseOne software is running, a BlueCue highlights the control in the form that corresponds to the control that you selected in JD Edwards Autopilot.
3. Select an input source from the Source of Input list.
4. In the value selection list, enter a literal value or select 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 selecting the Set Header Control Value option in the Command menu and repeating steps 1-5.

Selecting Options in a Header

Some forms contain options that you can select. You select the Check Box/Radio Button option in the Command menu to write commands for these options in a script. The command to select 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, JD Edwards Autopilot inserts the command in the Header node along with any Type To commands that you write.

To script selecting options in a header:

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

If the option is a radio button, the Source of Input list is unpopulated. JD Edwards Autopilot selects the radio button when you insert the command.

Typing Data in a Grid Cell

Before JD Edwards Autopilot can type data into a grid cell, it must know which grid cell; that is, which row and column in the grid. You specify the grid row by entering a separate Select Grid Row command into the script. This command must come immediately before the Set Grid Cell Value command. The Select Grid Row command identifies the row, and the Set Grid Cell Value command identifies the column. Furthermore, the Select Grid Row command must have the *position for Add/Edit* option set. This is vitally important and easily missed.

Important! Scripting the Select Grid Row and Set Grid Cell Value commands correctly is extremely important. It is common for new JD Edwards Autopilot users to script these commands incorrectly, which causes the script to fail. The Select Grid Row command is explained later in this chapter.

You use the command pane lists to script inputs in the grid area of a form. Selecting the Set Grid Cell Value option in the Command menu establishes the grid area as the context in which you type data. The options that you select 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 selected column and the value that you typed in it. JD Edwards 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 JD Edwards Autopilot to script the input in one row multiple times.

To enter a Type to Grid Cell command:

1. On the Command menu, select the Set Grid Cell Value option.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. In the command pane, select a grid column from the Grid Column list.

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

4. Select an input source.
5. Enter a literal value or select the name of a valid values list, variable, or UDC visual assist value.
6. Click the Insert button.
7. 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 values into additional grid cells in the same row:

1. In the Script pane in JD Edwards Autopilot, click a Detail Information command line that you inserted in the script.

   The insertion cursor is connected to the selected node.
2. On the Command menu, select the Set Grid Cell Value option.
3. In the Form/Subform Hierarchy list, select the appropriate form or subform.
4. In the command pane, select a grid column from the Grid Column list.
5. Select an input source.
6. Enter a literal value or select the name of a valid values list, variable, or UDC visual assist value.
7. Click the Insert button.
8. 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. (Optional) Deactivate the Playback button in the toolbar.
3. In the Script pane, click the Detail Information command line that you want to play multiple times.
4. In the command pane, select an input source, such as literal value or variable, in the Define Repeat Count From list.
5. Enter a literal value or select 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.

Assigning a UDC Visual Assist Value

JD Edwards Autopilot enables you to select 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 selected control or column. If the control or column does not contain a UDC value, the UDC Visual Assist Value option does not appear 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. Selecting the Set QBE Cell Value option in the Command menu establishes the QBE line as the context in which you type data. The options that you select 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 selected grid column and the value that you typed in it. JD Edwards 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 JD Edwards Autopilot.

To type data in a QBE line:

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

Using the Select Grid Row Command

This section provides an overview of the Select Grid Row command and discusses how to:

- Use the Operation Type list.
- Use the Action on Grid Row list.
- Use the Grid Columns list.
- Use the Source of Row Number list.

Understanding the Select Grid Row Command

The Select Grid Row command enables 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 records, add to a grid row, or edit the entries in a grid row.

JD Edwards Autopilot enables 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 select the Select Grid Row command, JD Edwards Autopilot also populates the command pane with a value selection list that enables you to enter a literal value or select the name of a previously created valid values list or variable. The caption of this list reflects the option that you select in the Source of Row Number list.

Using the Operation Type List

The Operation Type list in the JD Edwards Autopilot command pane enables you to select a row in a grid area either by specifying a row number or by specifying the value of a particular cell in a grid column.

Click by Row Number Option

The Click by Row Number option enables you to select a grid row by number. Selecting a grid row by number is faster than selecting a grid row by cell value, particularly if running against an HTML client and if the search involves many rows. JD Edwards Autopilot finds the designated row and performs the action selected in the Action on Grid Row list. You can designate the grid row 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 select a grid row that contains a particular value, such as an item number, use the Click by Cell Content option and select a grid column and an action that you want to perform on the cell. JD Edwards Autopilot selects the grid cell rather than the entire grid row.

Using the Action on Grid Row List

The Action on Grid Row list enables you to specify the purpose for selecting the row. In JD Edwards Autopilot, you can script these 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, it is not recommended 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 buttons. Clicking the grid row button selects the row, rather than a cell. Do not script this grid row operation in forms that do not contain a button.

Double-Click a Grid Row

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. Do not script this grid row operation in forms that do not contain grid row buttons.

Position Grid Row for Add/Edit

Before entering a value into a grid cell, you must select the grid row and position it for add/edit using the Position Grid Row for Add/Edit option. You must use this option with the commands on the Action on Grid Row list immediately preceding a Set Grid Cell Value command. Failure to script these commands in the correct order is the most common JD Edwards Autopilot scripting error.

Using the Grid Columns List

You use the Grid Columns list only when you have selected the Click by Cell Content option. When you select this option, JD Edwards 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 select Click by Cell Content, you can use the Source of Row Number list to specify a value that might exist in a particular cell of that column. JD Edwards Autopilot searches for the value and selects the first row that contains that value in the specified column.

Using the Source of Row Number List

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

Scripting the Select Grid Row Command

This section provides an overview of scripting the Select Grid Row command and discusses how to:

- Click by row number
- Click by cell content
- Perform grid row operations

Understanding 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, use the JD Edwards Autopilot command pane to select the row that you want to target for the 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 select 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 or edit, double-clicking the row, or double-clicking the row button.

To finish scripting the Select Grid Row command, select a value source for row selection. Possible 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 use the value selection list to select a valid values list or variable.

Clicking by Row Number

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

To click a grid row by row number:

1. On the Command menu, select the Press Toolbar Button option.
2. Select the Press Standard Button option.
3. Click Find.

   JD Edwards Autopilot fills the detail area in the active form.
4. On the Command menu, select the Select Grid Row option.
5. In the JD Edwards Autopilot command pane, select the Click by Row Number from the Operation Type list.
6. Select a grid row action from the Action on Grid Row list.
7. Select a value source from the Source of Row Number list.
8. In the value selection list, enter a literal value or select the name of a valid values list or variable.
9. Click the Insert button.

Clicking by Cell Content

You can have JD Edwards Autopilot 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 select a grid column as a search criterion.

To click a grid row by cell content:

1. On the Command menu, select the Press Toolbar Button option.
2. Select the Press Standard Button option.
3. Click Find.

   JD Edwards Autopilot fills the detail area in the active form.
4. On the Command menu, select the Select Grid Row option.
5. In the JD Edwards Autopilot command pane, select the Click by Cell Content from the Operation Type list.

6. Select a grid row action from the Action on Grid Row list.
7. In the Grid Columns list, select a grid column in which you want JD Edwards Autopilot to search for a value.
8. Select a value source from the Source of Row Number list.
9. In the value selection list, enter a literal value or select 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 these 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. Select 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 JD Edwards Autopilot commands. For example, single-clicking a grid row or grid row button enables you to write a command to click the Select or Delete button. Double-clicking a row enables you to access another form or application. clicking a row and positioning for add or edit enables you to edit the selected grid row

---

To single-click a grid row:

1. On the Command menu, select the Select Grid Row option.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. Select an operation from the Operation Type list.
4. Select the single-click option in the Action on Grid Row list.
5. Select a value source from the Source of Row Number list.
6. In the value selection list, enter a literal value or select a valid values list or variable.
7. Click the Insert button.

To double-click a grid row:

1. On the Command menu, select the Select Grid Row option.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. Select an operation from the Operation Type list.
4. Select the double-click option in the Action on Grid Row list.
5. Select a value source from the Source of Row Number list.
6. In the value selection list, enter a literal value or select a valid values list or variable.
7. Click the Insert button.
8. On the Command menu, select either the Form option or the Application Interconnect option.

9. In the command pane, select the next form or application that appears when you double-click the row in the form.
10. Click the Insert button.

To single-click a grid row button:

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

To double-click a grid row button:

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

To position a grid row add or edit:

1. On the Command menu, select the Select Grid Row option.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. Select an operation from the Operation Type list.
4. Select the Position for Add/Edit option in the Action on Grid Row list.
5. Select a value source from the Source of Row Number list.
6. In the value selection list, enter a literal value or select a valid values list or variable.

Note. During script playback, if the row number specified in the script is larger than the total number of rows in the grid, JD Edwards Autopilot positions itself at the bottom of the grid. A subsequent Set Grid Cell Value command will type to the blank row at the bottom of the grid, assuming that there is a blank row. By convention, many JD Edwards Autopilot users specify row *999* to go to the bottom of the grid.

---

7. In the Command menu, select the Set Grid Cell Value option.
8. Select a grid column in the Grid Column pane.
9. Select an input source in the Source of Input pane.

10. Enter a literal value or UDC value, or select a valid values list or a variable from the value selection list.
11. Click the Insert button.

    JD Edwards Autopilot adds to or edits the column in the selected grid row.

Important! For grid row operations to be successful, you must script these commands in the order shown above. Failure to do so is the most common JD Edwards Autopilot scripting error and will cause the script to fail.

Using the Press Toolbar Button Command

This section provides an overview of the Press Toolbar Button command and discusses how to:

- Use the Standard Button option.
- Use the Custom Button option.
- Use the Select Grid Tab option.
- Use the Grid Scroll Button option.

Understanding the Press Toolbar Button Command

The Press Toolbar Button command enables you to script many actions in JD Edwards EnterpriseOne applications, including clicking a toolbar button, selecting a command from an application menu, selecting a command from the exit bar, and scrolling the grid.

Using the Standard Button Option

The Standard Button option in JD Edwards Autopilot contains button-clicking selections that match the buttons in the toolbars of forms. For example, the toolbar in some forms features 10 button-clicking options. When one of these forms is active, the Standard Button tree in the Button list in JD Edwards Autopilot contains the same options.

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

When you select the Standard Button option, the Next Form list also appears. The option that you select from this list indicates the form that is 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 options in JD Edwards 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, select Submit from the Standard Button options in JD Edwards Autopilot.

See Also

Chapter 5, “Writing Scripts,” Submitting a UBE , page 44

Using the Custom Button Option

Use the Custom Button option to script a selection from the Row menu, Form menu, or Report menu. Selecting from the Row menu or the Form menu usually results in accessing 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 enables you to select the application and form in the command pane and insert the commands. JD Edwards Autopilot interconnects the applications, and the form in the new application appears.

When you select the Custom Button option from the Button list in the command pane, these options appear in the tree:

- Form
- Row
- Report

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

Note. The options in the Form and Row menus 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 click a standard button and to click 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 the Press Toolbar Button command, whereas the command line for clicking a custom button describes the command as the Select Menu Exit command. When you click a standard button, you usually access forms in the same application. When you click a custom button, you usually access a new application.

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 that you access.

Form Exit

JD Edwards Autopilot enables you to script a selection from the Form menu, just as you might select a menu option in a JD Edwards EnterpriseOne form. Typically, you perform a form exit when you want to access a form that is related to the current form. However, the new form might exist 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

JD Edwards Autopilot enables you to script a selection from the Row menu, just as you might select one directly in a JD Edwards EnterpriseOne form. Typically, you perform a row exit when you want to move from a particular 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 can 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 JD Edwards Autopilot automatically submits the UBE.

See Also

Chapter 5, “Writing Scripts,” Launching a UBE from a Row Menu , page 42

Using the Select Grid Tab Option

You use the Select Grid Tab option in JD Edwards Autopilot to test whether the system accesses customized grid tabs that you have created. In JD Edwards EnterpriseOne software, you customize the grid by selecting Preferences, Grid, New Format. To create a new tab, you select fonts, the number of grid columns, the width of grid columns, and so on. Each grid configuration creates a new tab. After you create as many custom tabs as you need, you can use JD Edwards Autopilot to script the selection of tabs. You determine whether the tab is selected, and you can also determine whether the customized changes appear.

Note. The Select Grid Tab features works with Windows-based JD Edwards EnterpriseOne applications only; it does not work with web-based JD Edwards EnterpriseOne applications.

Using the Grid Scroll Button Option

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

Scripting the Press Toolbar Button Command

This section provides an overview of scripting the Press Toolbar Button command and discusses how to:

- Click a standard button.
- Click a custom button.
- Select a grid tab.
- Click the grid scroll button.

Understanding Scripting the Press Toolbar Button Command

You use the Press Toolbar Button command in JD Edwards Autopilot to script many important functions. For example, clicking the Add button enables you to access a new form, either in the same application or a new one. By selecting the Standard Button option for the Press Toolbar Button command, you can write a script command to click the Add button. Other Press Toolbar Button options enable you to perform form and row exits, select a grid tab, or scroll through a grid.

Clicking a Standard Button

In general, you select the Standard Button option to click a toolbar button in a form. When you select this option, the options in the command pane match the toolbar buttons on the active form.

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

1. On the Command menu, select the Press Toolbar Button command.
2. Select the Standard Button option, and then select Add.
3. Select the name of a form from the Next Form list or select Unknown/None for another application.

Important! If you select the Unknown/None option and then click the Insert button, ensure that the Form command line in JD Edwards Autopilot matches the active form. On the Command menu, select Form, select the name of the active form, and then click the Insert button.

4. Click the Insert button.

   The next form appears.

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

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

To move from one form to another by using the Select button:

1. On the Command menu, select the Select Grid Row option.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. In the Source of Row Number list in the command pane, select literal value, valid values list, or variable.
4. To specify the row number to select, enter or select a value from the value selection list.
5. Click the Insert button.
6. On the Command menu, select the Press Toolbar Button option.
7. Select the Standard Button option, and then select the Select option.
8. In the Next Form list, select the form that appears next.
9. Click the Insert button.

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

To update the database:

1. On the Command menu, select the Press Toolbar Button option.
2. Select the Standard Button option, and then click OK.
3. If you are accessing a new form, select it from the Next Form list; if not, do not make a selection from the list.
4. Click the Insert button.

To fill a grid:

1. On the Command menu, select the Press Toolbar Button option.
2. Select the Standard Button option, and then select Find.
3. Click the Insert button.

To delete a record:

1. On the Command menu, select the Select Grid Row option.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. In the Source of Row Number list in the command pane, select literal value, valid values list, or variable.
4. To specify the row number to be selected, enter or select a value from the value selection list.

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

To exit a form:

1. On the Command menu, select the Press Toolbar Button option.
2. Select the Standard button option, and then select either Cancel or Close, depending on the button that is available on the form.
3. Click the Insert button.

See Also

Chapter 5, “Writing Scripts,” Launching a UBE from a Row Menu , page 42

Clicking a Custom Button

You use the Custom Button option to script selections from the Row menu and the Form menu. When you select this option, the selections in the command pane match the form and row exits in the active form.

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

To script a form exit:

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

To script a row exit:

1. On the Command menu, select the Press Toolbar Button option.
2. If the detail area in the active software form is empty, select Find.
3. Click the Insert button.

   In the application form, the detail area loads.
4. On the Command menu, select the Select Grid Row option.

Note. To display the new form, you can select Row from the menu bar in JD Edwards EnterpriseOne, click one of the forms in the list, or click Select.

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

5. Select the single-click row button from the Actions on Grid Row pane.
6. In the Source of Row Number list in the command pane, select a literal value, valid values list, or variable.
7. To specify the row number to be selected, enter or select a value from the value selection list.
8. Click the Insert button.
9. On the Command menu, select the Press Toolbar Button option.
10. From the Button list in the JD Edwards Autopilot command pane, select the Custom Button option.
11. Click Row.
12. Select a Row exit.

Note. If you run a row exit to launch a UBE, you do not make a selection from the Application or Next Form list that appears. You select the row exit only, click the Insert button, and then write a UBE command.

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

Selecting a Grid Tab

You can select customized grid tabs by using the Select Grid Tab command. JD Edwards Autopilot selects the grid tab number that you script and, in the play back while scripting mode, displays the grid with your changes.

To script the Select Grid Tab option:

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

   From left to right, the first grid tab is 1, the second grid tab is 2, and so on.
5. Click the Insert button.

Note. The Select Grid Tab features works with Windows-based JD Edwards EnterpriseOne applications only; it does not work with web-based JD Edwards EnterpriseOne applications.

Clicking the Grid Scroll Button

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

To script the Grid Scroll Button option:

1. On the Command menu, select the Select Grid Row option.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. In the command pane, under Operation Type, select the option to click by row number.
4. In the Action on Grid Row list, select single-click.
5. Select a source of row number and enter a literal value or select a variable or valid values list.
6. Click the Insert button.
7. On the Command menu, select the Press Toolbar Button option.
8. Select the Grid Scroll Button option.
9. Select one of these scrolling options in the tree:
   - Page Up
   - Page Down
10. Click the Insert button.

Using 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 buttons and clickable bitmaps. JD Edwards Autopilot displays uses the command pane to display the button options and clickable bitmaps in the active form.

This section discusses how to:

- Use the push button options.
- Use the clickable bitmap options.

Using the Push Button Options

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

Note. You cannot click these buttons using a Press Toolbar Button command because they do not appear on the toolbar. Therefore, if you use the Press Toolbar Button option in the JD Edwards Autopilot Command menu, none of these push button options appears in the command pane. Alternatively, if you select the Press Push Button option, the command pane uses the Select Button to Press list to display the button options that appear on the form.

When you write a command to click a button, you often access a new form. You can write a Form command for the new form so that the JD Edwards Autopilot script matches the actions that you take in the software. Sometimes, when you click a button, you access a new application. In that case, you must write an Application Interconnect command in JD Edwards Autopilot. You can verify whether you have accessed a new application by clicking Help and About JD Edwards in the toolbar of the active form.

Using the Clickable Bitmap Options

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

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

Note. For each clickable bitmap, JD Edwards Autopilot displays the system-assigned name, such as Bitmap 184, rather than the descriptive label that appears next to the bitmap 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 JD Edwards Autopilot to click. To help you identify the name of the bitmap, a BlueCue appears with the descriptive label on the form when you click the corresponding name in the JD Edwards Autopilot command pane.

When you script clicking a clickable bitmap, you typically access another form, and you must write a Form command in JD Edwards Autopilot to match the actions. If you access a new application when you click a clickable bitmap, you must write an Application Interconnect command in JD Edwards Autopilot so that the current application in the script matches the active application.

Scripting the Press Push Button Command

When you write a Press Push Button command, you enable a 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 buttons and clickable bitmaps that do not appear on the toolbar of the form. Writing a command to click a button or clickable bitmap enables you to access another form in the application. In some cases, this command enables you to exit to another application.

This section discusses how to:

- Click a button.
- Click a bitmap.

Clicking a Button

Suppose that you need to test an application, such as System Setup (P0000), that includes forms with oversized buttons that do not appear on the toolbar. In most cases, you click these buttons to access a new form in the same application or to exit to a new application.

Scripting the Press Push Button command in JD Edwards Autopilot enables you to click the button in the form. You cannot write a Press Toolbar Button command to perform this action because the Press Toolbar Button command does not apply to push buttons on the form.

To click a button in a form:

1. On the Command menu, select the Press Push Button option.

   JD Edwards Autopilot populates the command pane with button options only when the active form contains buttons.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. In the JD Edwards Autopilot command pane, select a button from the Select Button to Press list.
4. Click the Insert button.
5. If clicking the button accesses another form in the same application, select Form from the command menu.
6. In the Form list of the command pane, select the name of the active form.
7. Click the Insert button.
8. If clicking the button accesses a new application, select Application Interconnect from the Command menu.
9. In the command pane, select one of these options:
   - Application (to select the active application)
   - Form (to select the active form)
10. Click the Insert button.

Clicking a Bitmap

Some applications contain forms that use clickable bitmaps that do not appear on the toolbar. To test one of these applications, you must script a Press Push Button command to click a bitmap. In the command pane, JD Edwards Autopilot displays 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, JD Edwards 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 access another form or application. You cannot click the bitmap by writing a Press Toolbar Button command because the bitmaps do not appear on the toolbar.

To click a bitmap in a form:

1. On the Command menu, select the Press Push Button command.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. From the Select Button to Press list, select the name of a tab in the active form.
4. Click the node next to the tab name.
5. In the Form/Subform Hierarchy list, select the appropriate form or subform.
6. From the drop-down menu, select 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.
7. Click the Insert button.
8. If clicking the button accesses another form in the same application, select Form from the Command menu.
9. In the Form list of the JD Edwards Autopilot command pane, select the name of the active form.

10. Click the Insert button.
11. If clicking the button accesses a new application, select the Application Interconnect option in the Command menu.
12. In the command pane, select one of these:
    - Application (to select the active application)
    - Form (to select the active form)
13. Click the Insert button.

Using 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. JD Edwards 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, JD Edwards Autopilot locates the combo box and sends a message to the form to select the text string specified in the command pane.

When you select the command, JD Edwards 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, JD Edwards 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 table F0005.

Forms in some applications include hidden combo box controls that are not used. JD Edwards Autopilot displays these controls in the Choices list, just as it displays hidden header controls and grid columns in the command pane. You cannot select 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 select the Select ComboBox Item command from the Command menu of the JD Edwards Autopilot window and select a combo box from the Combo Box list in the command pane.

The Select ComboBox Item option is available in the Command menu of the JD Edwards Autopilot window only if you launch an application and form that use combo boxes.

After you select an option from the Combo Box list, JD Edwards 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 these cases, the selection that you make in the Combo Box list in the JD Edwards Autopilot command pane changes the items in the Choices list.

To script the Select ComboBox Item command:

1. From the Command menu, select Application.

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

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

   JD Edwards Autopilot enters the item from the Combo Box list to the control in the form.

Using the Build Tree Path Command

Some JD Edwards EnterpriseOne applications use tree controls and parent/child tree forms. 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 JD Edwards 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:

- 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 select *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, JD Edwards Autopilot uses the search string that you create to select the specified 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. 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, JD Edwards Autopilot displays a dialog box indicating 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

This section provides an overview of scripting the Build Tree Path command and discusses how to:

- Build a tree path using variable values.
- Build a tree path using literal values.
- Add a parent node or child to a tree path.

- Remove a parent node or a child from a tree path.

Understanding 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 select one of these 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 select *Variable* as the data type for the node.

- Set a numeric value for the variable.

  If you select *3* as the value, JD Edwards Autopilot selects the third node in the tree or the third child of a parent. In this case, you select *Ordinal* as the data type for the node.

Whether you select *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.

Note. Before building the tree path, declare a variable and set its value.

See Chapter 6, "Scripting Actions," Using a Variable as a Source of Input , page 65.

To build a tree path using variable values:

1. From the Command menu, select 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, select Build Tree Path.

   The JD Edwards Autopilot command pane contains a Tree Path list and a Data Type list.
4. In the Form/Subform Hierarchy list, select the appropriate form or subform.
5. Select one of these options from the Data Type list:
   - Text Variable

     Select Text Variable if the variable that you want to use has a text value.
   - Ordinal Variable

     Select Ordinal Variable if the variable that you want to use has a numeric value.
6. From the Select Variable list, select a variable for which you have set a value.
7. Click Add.

   JD Edwards 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 JD Edwards Autopilot window, launch an application and form that uses tree controls.
2. From the Command menu, select the Build Tree Path option.
3. In the Form/Subform Hierarchy list, select the appropriate form or subform.
4. Select one of these options from the Data Type list:
   - Text Literal

     Select Text Literal to use a text value.
   - Ordinal Literal

     Select Ordinal Literal to use a numeric value.
5. In the Enter Node list, type a literal value.
6. Click Add.

   JD Edwards Autopilot inserts the literal value as a node in the Tree Path list.

   When you play back the script, JD Edwards Autopilot finds the node with the literal value, based on the tree path that you build.

Adding a Parent Node or Child to a Tree Path

After you select a data type and enter a literal value or select a variable, you add a tree path node by:

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

Remember, 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 JD Edwards Autopilot window command pane, click a node that does not have a child.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform.
3. Select one of these options from the Data Type list:
   - Text Literal
   - Text Variable
   - Ordinal Literal
   - Ordinal Variable
4. Select a variable from the value selection list or enter a literal value in the Enter Node list.
5. Click Add.

To remove a parent node or a child from a tree path:

1. In the Tree Path list of the JD Edwards Autopilot window command pane, click a parent node or a child.
2. To remove a parent node and its child, select the parent node, and then click Remove.
3. To remove the child only, select the child, and then click Remove.

Removing a Parent Node or a Child from a Tree Path

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

Using the Database Validation Command

This section provides an overview of using the Database Validation command and discusses how to:

- Define validation.
- Use validation declaration.
- Use validation association.
- Execute validation.
- Use the Expect No Matching Records option.

Understanding Using the Database Validation Command

Use the data validation action commands to verify data in the JD Edwards EnterpriseOne database.

The data validation process in JD Edwards Autopilot consists of these steps:

- Declaration, in which you give the validation a name.
- Association, in which you pair expected values with columns in a database table.
- Execution, in which you execute your association criteria to verify whether the expected data exists in the database.

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

Defining Validation

You script validation commands to compare a data set that you created in JD Edwards Autopilot and ran in JD Edwards EnterpriseOne 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 JD Edwards Autopilot, such as century information. In most transaction fields, you make no entry for century. Using validation commands, you can verify that century data actually appears in the database.

Using 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 select
- A table to be validated, which you select 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 means to store values.

Using Validation Association

Associating a validation enables you to store data. You can gather this data from different points in the script. No action is taken when you associate a validation. You merely associate data with the declared validation command.

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

You must specify this information to script the validation association:

- Validation name (declared validation).
- Database column identification, which you select 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 select, but you must use both a key selection value and a validation value.

Key Selection and Validation Values

Each validation association must be marked as either a *key selection value*or a *validation value*. Key selection values identify the database rows to be validated. They become part of the WHERE clause that JD Edwards Autopilot generates when the validation is executed. Validation values are the values JD Edwards Autopilot compares against the specified columns in the selected records. A validation normally has at least one key selection value and at least one validation value.

Executing Validation

You run the validation using the record selection criteria that you establish in validation association. JD Edwards Autopilot retrieves the specified data from the database through SQL. You can then compare the retrieved data with the data that you expected the system 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.

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

- A SQL SELECT 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 select when you declare the variable, the validation value columns that you select during association, and the key selection that you make during association.

A SQL statement that you generate when you run the validation might contain the identity of the table that contains the expected data (for example, F0101), the columns that contain the expected data (for example, ABALPH, ABAT1 and ABAN8), and the key column that selects the records to be validated (such as ABAN8).

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

If you made an error in scripting the validation associations, JD Edwards Autopilot records an error when it executes the Validation Execution command. For example, if you do not select a key column during association, JD Edwards 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, JD Edwards 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.

Using the Expect No Matching Records Option

If you enter a record, successfully validate it, and then write a command to delete the record, you can validate that the system successfully deleted the record from the database. To do so, you run the validation again and select the Expect No Matching Records option. When you run the script, JD Edwards Autopilot again checks the database. The validation runs successfully if JD Edwards Autopilot indicates that the record that you deleted from the database no longer exists. When you select this option, JD Edwards Autopilot verifies that no records match the key selection criteria you specified.

Scripting the Database Validation Command

You enter database validation action commands as you write a 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.

This section discusses how to declare a validation.

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, select the Declare New Validation option.
2. In the command pane, enter a name in the Validation Name list.

It is conventional to name the validation the same as the table it validates.

3. Select an option in the Database Table list.

   This selection identifies the database table against which you validate data from JD Edwards Autopilot.

4. Click the Insert button.

Associating a Validation

After you declare the validation, you have a place in which you can store values. During validation association, you select values that you want to validate and you 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.

There are two types of validation associations:

Association Type	Description
Key selection value	Associations that determine which database record (row) you verify.
Validation value	Associations that specify columns within that row whose values you want to verify.

A validation normally includes at least one key selection value association and at least one validation association.

To associate a validation:

1. From the Command menu, select the Associate a Validation Column option.
2. Select a validation name that you created using the Declare New Validation option by scrolling through the list and clicking a name.
3. Select a column name from the Database Column list.
4. Make the appropriate choice in theValue Type group box.

   This option associates the selected column with a particular value.

5. (Optional) In the Currency Type group box, select one of these currency options:
   - Domestic
   - Foreign

   Note. 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 select the name of a variable, header control or grid column.

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

Note. With the play back while scripting feature activated, if you select a header control or grid column, JD Edwards Autopilot uses a BlueCue to highlight the designated control or column in the form. Be sure to select a header control or grid column to which you have previously entered a value. Likewise, if you have selected 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. Repeat steps 1 to 8 for as many key values and validation values as you need.

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

Executing a Validation

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

To execute a validation:

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

   Note. JD Edwards 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, and the key selection value.

3. Click the Insert button.

   Later in the script, you might delete the records. You can validate that these deleted records are no longer in the database.
4. Repeat steps 1-3.
5. Select the Expect No Match Records check box.
6. Click the Insert button.

   Note. You can declare and set the value of a variable to test validation success.

Using the Command Line

Sometimes, you may need to run other programs within a JD Edwards 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 JD Edwards Autopilot to continue scripting inputs.

You can complete these tasks by selecting the Command Line option from the Command menu. Type the path to the program that you want to run. (This is similar to using the Run function in Microsoft Windows.) JD Edwards Autopilot opens the program and the document or presentation that you created.

You can also send a command line message to create screen shots of JD Edwards EnterpriseOne software forms at designated points in the script playback process. You can create the screen shots 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 create a Command Line command that creates a screen shot of the current form. You specify a folder where JD Edwards Autopilot stores the screen shots.

This section discusses how to:

- Use the Command Line command.
- Create a screen shot of the current form.

Using the Command Line Command

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

To use the Command Line command:

1. From the Command menu, select the Command Line option.

   The command pane displays the unpopulated Command Line list and options.
2. Select the Command Line option.
3. Complete the Command Line field.

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

   Note. Deactivate the play back while scripting feature if you do not want the program to open while you are writing the script.
4. Click Insert.

Creating a Screen Shot of the Current Form

You can set up the path and file extension for any screen shots that you create using the Capture Current JD Edwards Window command. You do so by clicking Tools, selecting Options from the drop-down menu, and then selecting the Directories tab on the Options form.

In the Screen Capture field on the Directories tab, you type the path where JD Edwards Autopilot stores the screen shots of forms. You then select the format in which you want to save the images.

To use the Command Line command to create a screen shot of a form, type a name for the image in the File Name field. JD Edwards Autopilot creates a screen shot of the current form and saves it in the specified format in the specified directory.

To create a screen shot of the current form:

1. From the Tools menu, select Options.

2. On the Options form, select the Directories tab.
3. Complete this field:
   - Screen Capture

     Type the path where you want JD Edwards Autopilot to store the screen shots.
   - Format

     Using the drop-down menu, select the file extension, such as .tif, that you want to use for the screen shots.
4. Click OK.
5. With a script open, select Command Line from the Command menu.
6. Click the Capture Current JD Edwards Window option.
7. With a JD Edwards EnterpriseOne form active, complete the File Name field.
8. Click Insert.

   JD Edwards Autopilot stores the screen shot in the specified location.

CHAPTER 7

Working with the Script Pane

This chapter provides an overview of the Script pane and discusses how to:

- Modify scripts.
- Use script retention.
- Reuse Scripts.

Understanding the Script Pane

This section discusses:

- Script pane overview.
- Script pane structure.
- The insertion cursor.

Script Pane Overview

You can work in the Script pane in Oracle's JD Edwards EnterpriseOne Autopilot 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 exist in 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. JD Edwards 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. Any changes that you make to the parent command affect the child command.

All 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.

Script Pane Structure

As you write JD Edwards 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 scripts and adapt them to your specifications.

Context commands define the context for each scripted action. Each context command that you write and insert in the script creates a node, which appears in the Script pane with a plus or 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 any script, there are context commands and action commands, related to each other in various ways. Follow the indentation to understand the relationships of the nodes in the script. When you move a parent node, JD Edwards Autopilot preserves the structure of the nodes under the parent. 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. In that case, the next command you write is a child. Alternatively, you can make the next command a parent that is independent of changes to other nodes in the script.

Command lines illustrate the selections 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 the script or the actions that you take within the context. For example, a header in a form is a context; the action that you take within the context might be typing data in a specified field.

Context commands direct JD Edwards Autopilot to applications, universal batch engines (UBEs), processing options, interconnected applications, forms, header controls, grid columns, and Query By Example (QBE) lines. They therefore express the environment in which actions–such as typing data and clicking buttons–are carried out.

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. An expand/collapse (+/-) button accompanies this node, and every other kind of parent node, in the script pane.

Context commands can perform these functions:

- Form nodes that can be identified in the script with the node symbol or button, which appears as a plus or minus sign.

  Nodes can be expanded or collapsed by clicking the expand/collapse button. Nodes can function as the parents of action commands and, sometimes, other context commands. The child commands are indented beneath the parent context commands in the Script pane.

- Form discrete command line units.

  If two nodes are parallel, commands that you add to one node do not affect the node that is parallel to it.

- 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.

- Run 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.

After you script context commands, you script action commands to specify actions. In contrast to context commands, action commands have these characteristics:

- They cannot have repeat counts
- They are always indented beneath context commands in the Script pane.

  This indicates that they are subordinate to context commands in the command hierarchy.

The Insertion Cursor

The insertion cursor, which appears in the Script pane as a red arrow, indicates the position in the script where 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.

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 enables 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, to work with a particular form, you must first select an application; therefore, JD Edwards Autopilot inserts and indents the Form command line below the Application command line. Similarly, to add a form by clicking a button, JD Edwards Autopilot inserts and indents the Press Toolbar Button command below the Form command line.

The hierarchy of nodes expresses the logic that you follow to build JD Edwards 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:

1. Application

2. Form
3. Header
4. Type To
5. Press Toolbar Button {OK}

Because you must launch an application before you can write any of the subsequent commands, the Application command is 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 of the Header command, which is indented. Finally, the header is the context for performing the action of entering data to a control, so the Type To command is indented beneath the Header command line.

Any context or action command that you insert in the script as a child of another command is affected by changes that 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. Note these points about 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. This prevents you from accidentally creating an invalid script by moving a command to a foreign form.
- A context command node that is attached to one Application node cannot be dragged into another Application node. This prevents you from creating an invalid script by moving a command to a foreign application.
- 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.

If two nodes are parallel, you can make one node a child of the other by dragging and dropping the node. JD Edwards Autopilot indicates the parent-child relationship by indenting one node beneath the other.

Before JD Edwards Autopilot creates the parent-child relationship, it displays the dialog box that prompts you to confirm the action.

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 JD Edwards 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

Chapter 6, “Scripting Actions,” Updating the Repeat Count in a Node, page 65

Modifying Scripts

This section provides an overview of changes to scripts and discusses how to:

- Collapse the script tree.
- Expand the script tree.
- Use nodes.
- Delete command lines.
- Change the sequence of commands.
- Edit an action command line.
- Edit a context command line.

Understanding Changes to Scripts

You can configure scripts to your precise specifications to test applications. You can also alter the display of the script tree by expanding or collapsing nodes, or you can make substantive changes 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 the JD Edwards Autopilot command pane to add, delete, or edit commands. You can also modify the structure of the script tree by moving 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, with long scripts, it is useful to view 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.

You can select any point at which to collapse the tree. For example, when you click the expand/collapse button on a parent node, any nodes that you have inserted in it also collapse. When you click a child node, only that node collapses.

Adding Command Lines

You can insert a command in a script after you have passed the point where you want to insert it. For example, suppose 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 where 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 toolbar are active. If they are not active, press and hold the mouse button, drag the insertion cursor on top of a command line, and release the mouse button. Click Yes or No, depending on where you want to insert.

Moving Command Lines

You can also 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 JD Edwards Autopilot. During playback, JD Edwards 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.

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 selecting a command line in the Script pane and then using the command pane to make new selections from lists to update the content of the command.

Note. The command lines for the Press Toolbar Button command cannot be edited. These must be deleted or added to the script as necessary.

Collapsing the Script Tree

Access the JD Edwards Autopilot window.

1. In the Script pane, expand nodes by clicking node buttons that displays a plus sign.
2. Click a node that shows a minus sign.

   The node collapses and displays only the context command line.

   Note. You can also collapse all nodes in a branch by right-clicking the first node of the branch and choosing Collapse All. You can collapse all nodes in the script by right-clicking the Begin Script line and choosing Collapse All.

Expanding the Script Tree

Access the JD Edwards Autopilot window.

1. In the Script pane, collapse nodes by clicking any node button that displays a minus sign.
2. Click a node that displays a plus sign.

   The node 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 the first node of the branch and selecting Expand All. You can expand all nodes in the script by right-clicking the Begin Script line and selecting Expand All.

Using Nodes

Access the JD Edwards Autopilot window.

1. In the Script pane, click the insertion cursor.
2. Press and hold the mouse button.
3. Drag the insertion cursor to the point in the script where you want to insert a new command.

   As you drag the cursor, an indicator arrow appears. The arrow, which points up or down, indicates the placement of the cursor above or below a selected command line.
4. Release the mouse button at the appropriate point.
5. Follow the steps required to insert the command in the script.

Note. If you cannot insert a command at a particular point in the script, JD Edwards Autopilot displays a disallow symbol 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.

Access the JD Edwards Autopilot window.

1. In the Script pane, select a command by clicking it in the script.
2. Right-click and select Delete, or press the Delete key.

Changing the Sequence of Commands

Access the JD Edwards Autopilot window.

1. Select a node in the script pane by clicking it.
2. Press and hold the mouse button, and drag the node.

   A disallow symbol indicates that you cannot drop the node in a particular spot in the script.
3. When the target node is highlighted release the mouse.

   The mouse pointer indicates whether the command appears above or below the targeted command line in the Script pane.
4. In the dialog box, click Yes or No when the system prompts you to insert as a child.

Editing an Action Command Line

Although 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 selecting options in the command pane.

To edit an action command line:

1. In the Script pane of the JD Edwards Autopilot window, 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. JD Edwards Autopilot highlights the original options that you made in the command pane.

2. In the command pane lists, select any new options.
3. Click Update.

   JD Edwards Autopilot updates the command line in the Script pane to reflect the changes.

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 JD Edwards Autopilot loops through the node during script playback.

To edit a context command line:

1. In the Script pane of the JD Edwards Autopilot window, click the context command line.
2. In the command pane, select 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 JD Edwards Autopilot to loop through the node during playback.
4. Click Update.

Using Script Retention

This section provides an overview of script retention and discusses how to:

- Save scripts.
- Use the Include command.
- Link variables between scripts.
- Share scripts.

Understanding Script Retention

JD Edwards Autopilot enables you to save, modify, reuse, combine, and send scripts. These capabilities broaden the scope of and audience for tests. With JD Edwards Autopilot, you can perform these tasks, which are integral to building a system of scripts:

- Save scripts, which you can reuse or modify.
- Include scripts within other scripts to broaden the scope of testing.
- Pass variables between scripts in a master script that consists of a parent script and one or more children.
- Share scripts.

You can save scripts either on your local drive or in the JD Edwards 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 other scripts can pass variable values; you accomplish this by declaring variables as external and creating links between variables in separate scripts. In addition, you can send an email message to colleagues with any script that you create.

Saving Scripts

You can save scripts as you work by using the File menu. 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 it as you work.

If JD Edwards EnterpriseOne or JD Edwards Autopilot fails during a JD Edwards Autopilot session, the tool saves any scripts that are open. You can use the Configure tab on the Options form to set the conditions under which JD Edwards Autopilot auto-saves scripts as you work.

Using the Include Command

You can expand the testing scope by including one or more scripts on your local drive or in the script repository within a parent script. Do so by selecting the Include Local Script option or the Include Reposited Script option from the Tools menu. JD Edwards 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 from your local drive or a script from the repository, JD Edwards Autopilot displays a form that includes all of the scripts that you have stored locally or a form that you can use to select from the repository. You select the scripts to include, and JD Edwards 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 within another. For example, if you script 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 JD Edwards Autopilot plays back the included script, it loads data into a form twice, which results in an error. You can open a script that has scripts included in it and edit any of the included scripts. JD Edwards Autopilot reloads the original script with the changes that you make to the included script.

Each time that you write a command to include a script, JD Edwards Autopilot displays a dialog box that prompts you to specify whether to continue script playback on an Include branch error. If you click Yes, when an error occurs during the playback of the included script, JD Edwards Autopilot reports the error but continues with playback of any other included scripts that exist. This feature is particularly useful if you are running long scripts or batches of scripts.

See Also

Chapter 6, “Scripting Actions,” Using a Variable as a Source of Input , page 65

Linking Variables Between Scripts

When you include scripts within a parent script, you can also share variable values between the scripts. You use *variable linking* to do this. When linking variables, you declare a variable in a parent script. When you write a script that you include in the parent script, you also declare a variable; however, to link the variable, you declare it as external. JD Edwards Autopilot enables you to link the externally declared variable to any variable that you declare in the parent script. With the link, the value that you set for the variable can be passed between scripts.

To increase the versatility of scripting, JD Edwards Autopilot enables you to designate a default value for any variable that you designate as external. Doing so enables you to run an included script in standalone mode. JD Edwards 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, JD Edwards 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 standalone mode. For example, you might write a script that tests one set of functions by itself. You might then include this script within 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 need to play back the original script in standalone mode. If you assign a default value to the variable and run the script in standalone mode, JD Edwards Autopilot uses the default value to run the script. If you leave the default value of the external variable blank, JD Edwards Autopilot reads the value as a null string.

When you create a script with an included script and linked variables, JD Edwards 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, JD Edwards Autopilot does not pass this value to the linked variable in the parent script. In this case, JD Edwards 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.

This table summarizes three scenarios and the results that occur when you write scripts with default variable values:

Location Where Default Value Is Set for External Variable	Variable to which Link Is Created	Result
Parent script	External variable in child (included) script	JD Edwards Autopilot passes the default value to linked variables in any child (included) scripts.
Child script	Variable in parent script	JD Edwards 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.
Standalone script	Not applicable	The variable with the default value behaves as a local variable. JD Edwards Autopilot uses the default value wherever the script indicates.

External Variables for Script Linking

You use external variables to 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 enables you to designate a variable as external.

For example, suppose that 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 within A. Script A is the parent script, and script B is the child. JD Edwards Autopilot prompts you to link the externally declared variable to a variable in script A. You link variable X to variable X. In this case, 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, JD Edwards Autopilot can now pass this value to script B when you run the two scripts together.

Variable Links

When other scripts that contain external variables are included within a parent script, you must define the links to each external variable so that JD Edwards Autopilot can pass values between scripts. Defining the links enables JD Edwards 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, JD Edwards Autopilot displays a message box that prompts you to continue the script playback on an Include branch error.

Continuing the script on include branch error means that, if JD Edwards 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. Clicking Yes is recommended if you intend to test a long series of included scripts.

After you respond to the prompt, JD Edwards 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.

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 following path indicates that the included script is child.ats and that its externally declared variable <x> is linked to the variable <x> in the parent script.

```
Link C:\Autopilot\ats\child.ats <x> to <x>
```

To change the link path, you click the name of a different variable from the parent script.

When you select a variable, JD Edwards Autopilot establishes the link and inserts a Link script object and command line in the Script pane. JD Edwards 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, JD Edwards 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

JD Edwards Autopilot searches recursively for links. This means that the search continues until it meets one of these conditions:

- The linked variable in the parent script is not external.
- The linked variable is in the master parent script; JD Edwards Autopilot has searched until it reached the top of the script tree.

JD Edwards 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, JD Edwards Autopilot searches the parent script for the link to the external variable. When JD Edwards 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, JD Edwards Autopilot stops its search. If the declared variable is external, JD Edwards 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 within B. When you insert the Include command, JD Edwards Autopilot prompts you to create the links between the parent script (B) and the included scripts (C and D). When you run the script, JD Edwards 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 within script A. Script A now becomes the master parent script. You now declare the variable in script B as external and include it within script A. When you run the script, JD Edwards Autopilot searches for the link in the parent script. When JD Edwards Autopilot reaches script B, it reads the declared variable. Because 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, JD Edwards Autopilot can maintain as many links as you need.

The master parent script–that is, the script at the top of the tree–should not contain variables that you declare as external. These are unresolved variables. They have been declared as external, but they have not been linked. Declaring these variables as external causes the script playback to fail. In a well-written script, all external variables link to other variables in a parent script.

Sharing Scripts

After you create a script and modify it as necessary, you can email the script to other people. However, only people who have JD Edwards Autopilot installed 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. If you find 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 does not exist, the link is broken.

JD Edwards Autopilot notifies you that broken links exist 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, JD Edwards 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 case, you must repair the broken links.

Reusing Scripts

This section provides an overview of script reuse and discusses how to:

- Include scripts.
- Include one local script within another.
- Include a reposited script within another script.
- Edit an included script.
- Create variable links.
- Declare a variable as external.
- Assign a default value to an external variable.

Understanding Script Reuse

After you create and modify scripts in JD Edwards Autopilot, you can save them and then reuse them, as necessary. You can include scripts that you have created in other scripts that you are working on, and you can email saved scripts to other script writers and testers.

Saving and reusing scripts enables you to reduce the workload of other people who test the same applications that you are testing.

JD Edwards Autopilot also enables you to declare variables as external. 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. Because you can pass values between scripts and change those values, scripts are versatile, reusable, and dynamic.

Including Scripts

Including scripts enables you to broaden your testing scope. You can include one or more scripts within another script, either from your local drive or from the script repository. Each time that you include scripts within 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; however, any changes that you make to an included script affect the master.

Including One Local Script within Another

Access the JD Edwards Autopilot window.

1. In the Script pane, place the insertion cursor at the point where you want to insert the included script.
2. From the Tools menu, select the Include Local Script option.
3. On the Select Files to Include form, click the name of the script that you want to include.

   To include more than one script, press the CTRL 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.

   JD Edwards Autopilot inserts the script in the Script pane of the open script at the point of the insertion cursor. JD Edwards 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 clearing the Continue Playback on Error check box in the command pane.

Including a Reposited Script within Another Script

Access the JD Edwards Autopilot window.

1. In the Script pane, position the insertion cursor at the point where you want to include a script.
2. From the Tools menu, select the Include Reposited Script option.
3. On the Select Script form, complete any of the fields on any of the tabs to narrow the search, and then click OK.
4. On the Include Repository Script form, 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.

   JD Edwards Autopilot places the included script in the Script pane of the open script at the point of the insertion cursor.

Editing an Included Script

Access the JD Edwards Autopilot window.

1. In the Script pane, select the Include command line of the script that you want to edit.
2. Right-click and select Edit for a local script, or select Check Out & Edit for a reposited script.

   Important! JD Edwards 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 select Edit again.
3. Perform the necessary edits to the included script, and then select File, Close.
4. Save the changes to the included script.

   If you edited a reposited script, JD Edwards Autopilot prompts you to reload the script.
5. To load the changes to the included script, click Yes.

   JD Edwards 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 enable JD Edwards Autopilot to pass variable values between two or more scripts. You declare a variable and set its value just as you do in a standalone script. However, by declaring the variable as external, you indicate that it can be linked to a variable in another script. JD Edwards 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. If you break links between variables, you must recreate the links before the script will run.

To declare a variable as external:

1. From the Command menu, select the Variables option.
2. Type the name of a new variable in the unpopulated New Variable list.
3. Select the External option.
4. If you are declaring the variable but not setting a value, select Unknown/None from the Source of Value list, and click the Insert button.
5. If you are setting a value for the variable, select from the Source of Value list and the value selection list.
6. Click the Insert button.

Note. JD Edwards Autopilot includes the word External in the Declare command line in the Script pane.

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 standalone script. In JD Edwards Autopilot, you can select 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 within another script.

Assigning a Default Value to an External Variable

You can set a default value for a variable, regardless of whether you select to declare it as external. By assigning a default value, you ensure that you can run the script in standalone 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, JD Edwards 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, select the Variables option.
2. In the JD Edwards Autopilot command pane, type the name of a variable in the New Variable list.
3. Select the External option.

4. In the Default list, type a value for the variable.
5. Click the Insert button.

CHAPTER 8

Playing Back the Script

This chapter provides an overview of script playback and discusses how to:

- Configure automatic script playback.
- Use manual script playback options.
- Run script playback.

Understanding Script Playback

You can play back the script at any point, regardless of whether you are finished writing commands.

Oracle's JD Edwards EnterpriseOne Autopilot offers these options for playing back the script:

- From the beginning to the end without interruption.
- From a selected cursor position to the end without interruption.
- From the beginning of a selected script branch (node) to the end of the branch.
- From a selected cursor position line-by-line.
- From a selected position to a designated breakpoint.

You can stop playback at any time by selecting Stop from the Play menu or by clicking the Stop button on the toolbar.

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 configure playback by selecting Options from the Tools menu and selecting the Playback tab. For example, you can set JD Edwards 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 JD Edwards Autopilot and JD Edwards EnterpriseOne software event that occurs during the session.

When you play the script, JD Edwards Autopilot stops the playback if an error occurs in the software. If you have configured playback to save and display test results, appears in the Test Results form to indicate the playback was unsuccessful. If JD Edwards Autopilot encounters no errors during playback, it displays a message indicating success.

Configuring Automatic Script Playback

This section provides an overview of automatic script playback and discusses how to:

- Use play back during script creation.
- Store and display playback data.
- Handle breakpoints.
- Set playback speed.
- Set the Cancel Playback on Comm Error (cancel playback on communication error) option.
- Set the Log Variables on Script Failure option.
- Set event stream.

Understanding Automatic Script Playback

At any time, you can set or change the script playback configuration to have JD Edwards Autopilot play back certain features without your intervention. To configure script playback, you use the Options form, which you access by selecting Options from the Tools menu in the JD Edwards Autopilot window.

Using Play Back During Script Creation

When you select the Play Back while Creating Script option, JD Edwards Autopilot plays each command after you insert it in the script. To write a script without any delay caused by script playback, you should clear this option. Alternatively, to observe each command that you script, select this option.

Storing and Displaying Playback Data

JD Edwards Autopilot enables you to store and display the results of each script playback. If you save results data, JD Edwards Autopilot saves script playback results as a binary large object in table F97214. If you display test results after playback, JD Edwards 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 select to save script playback results, you can view a history of playback results by selecting Results from the Tools menu.

See Also

Chapter 10, “Storing Scripts and Test Results,” Understanding Script Reporting, page 159

Handling Breakpoints

A breakpoint is a point in the script that halts playback until you manually continue it or cancel it. To play the script uninterrupted but keep any breakpoints that you have inserted, select the Ignore Breakpoints during Playback option.

Setting Playback Speed

If you select the Accelerated Playback option, the system notifies JD Edwards Autopilot as soon as processing is complete so that playback can immediately continue. In general, you should select this option only when you are running relatively simple scripts that do not require the system to perform a large amount of processing.

Setting the Cancel Playback on Comm Error Option

If you select the Cancel Playback on Comm Error option, and you experience software communication issues, JD Edwards Autopilot cancels the current script. If you do not select the Cancel Playback on Comm Error option, JD Edwards Autopilot attempts to complete the script.

Setting the Log Variables on Script Failure Option

If you select the Log Variables on Script Failure option, JD Edwards 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 JD Edwards Autopilot results output. You use this output to analyze script failures. In the results database, you can view variable contents and immediately draw conclusions about the cause of a script failure. For example, the journal date variable value of *06/03/02* causes the script to fail if the year is not 2002. This is typical of the variable data that you might view. It is recommended that you select this option.

Setting Event Stream

The term *event stream* refers to the flow of information from the software to JD Edwards Autopilot that occurs during playback. You select one of these options to configure script playback to capture this information:

- None
- JD Edwards warning and error messages
- Level 1 API calls
- All API call levels

Using Manual Script Playback Options

This section provides an overview of manual script playback options and discusses how to:

- Use the Play from Top option.
- Use the Play from Cursor option.
- Use the Play Branch option.
- Play the script from a selected line command.
- Play the script to the next line command.
- Toggle a breakpoint.

- Use Wait Before Proceeding.
- Use script comment.
- Ignore breakpoints during playback.
- Stop playback.

Understanding Manual Script Playback Options

After you configure script playback, you run playback using the Command menu or the playback buttons on the toolbar. You can play the script from the top without interruption, play the script from any selected spot to the end, play only a selected 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 stop and resume script playback.

Using the Play from Top Option

When you select Play from Top in the Play menu, script playback begins with the first command line and continues until the end of the script unless JD Edwards 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 toolbar. Before you play back the entire script, remove any breakpoints that you inserted in the script.

Using the Play from Cursor Option

The Play from Cursor option enables you to select any spot in the script and then play the script to completion, if JD Edwards Autopilot does not encounter an error or a breakpoint.

Using the Play Branch Option

The Play Branch option enables you play only a single script node that consists of one or more context commands and a series of action commands.

Playing the Script from a Selected Line Command

To manually control playback from a chosen point in the script, you select the Stepping On option. You can play back either from the top, from a selected cursor position, or from a chosen branch of the script. You can then play the script a line at a time or from breakpoint to breakpoint.

Playing the Script to the Next Line Command

After you select Stepping On and select to play back either from the top, from a selected cursor position, or from a selected branch, you select the Step Next option to play the script one command line at a time. Script playback then proceeds step by step as you click Step Next or Continue To Breakpoint on the toolbar or Play menu.

Toggling a Breakpoint

A breakpoint is a command line that you select to stop playback until you resume or cancel playback. You can insert as many breakpoints in the script as you need. This enables you to isolate areas of the script and observe the playback.

You toggle the breakpoint by right-clicking the line in the script where you want playback to break and selecting Toggle Breakpoint. You can script multiple playback breakpoints. Script execution is suspended at each breakpoint unless Ignore Breakpoints is selected. Another way to stop JD Edwards Autopilot script execution is by clicking the Stop button. While the use of breakpoints allows script execution to be resumed later, clicking the Stop button ends the playback session.

Using Wait Before Proceeding

You use the Comment/Wait option 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*.

Using Script Comment

Using the Comment/Wait option, you can write brief comments that, for example, explain the reason for a Wait command. If you exchange scripts with someone else, you can use the comments to explain the actions that occur at a particular point in the script or to explain what the script is designed to test. You type a comment in the Comment field in the command pane and insert it in the script.

Note. JD Edwards Autopilot truncates the comment in the Script pane at 54 characters, including spaces.

JD Edwards Autopilot also enables 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 select the Comment/Wait option from the Command menu, the command pane also displays these options:

- Wait until message window is closed
- Log To Test Manager
- Fail Script

Select the *Wait until message window is closed* option if the script clicks the Delete button in a form. If you select the option, JD Edwards Autopilot does not proceed with script playback until it has clicked OK on the Confirm Delete form.

Select 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 select 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.

Select the Fail Script option if a critical event in the script caused the script to fail. If you select this option, JD Edwards Autopilot automatically fails the script at the point where you insert the command.

If you fail the script, JD Edwards Autopilot inserts a comment symbol in red.

You can use the Fail Script option in conjunction with logging comments to Test Manager by selecting both options. When you run a batch test, JD Edwards Autopilot fails the script and generates a summary report in Test Manager that lists any comments that you include about the failure.

See Also

Chapter 10, "Storing Scripts and Test Results," Understanding Script Testing, page 165

Ignoring Breakpoints During Playback

You can preserve breakpoints that you have inserted in the script, but run the script one or more times without breakpoints. Rather than toggle the breakpoints on and off, you can select the Ignore Breakpoints during Playback option. When you want to run the script to the breakpoint that you designated, clear the Ignore Breakpoints during Playback option and play back the script.

Stopping Playback

At any point during playback, you can stop the process by selecting the Stop option in the Play menu or by clicking the Stop button on the toolbar. If the script is playing, the Stop button appears red. When you click the Stop button, JD Edwards Autopilot records a Script Playback Canceled message, and the Stop button appears gray.

Running Script Playback

This section provides an overview of script playback and discusses how to:

- Play the script from the top.
- Play the script from a particular cursor position.
- Play a branch of the script.
- Play the script to the next command.
- Toggle a breakpoint.
- Play the script to a breakpoint.
- Continue playback to a breakpoint.
- Ignore breakpoints during script playback.
- Insert a Wait command in the script.

- Fail a script.
- Set transaction times in the script.
- Insert a comment in the script.

Understanding Script Playback

You run the various script playback functions in JD Edwards Autopilot using the options in the Play menu or the playback buttons on the toolbar. You can also use the Comment/Wait option to script one or more pauses in the playback and to insert comments in selected command lines in the script or as standalone lines. In addition, you can right-click a command line in the Script pane to toggle a breakpoint on and off.

Ignoring Breakpoints in the Script

You can ignore breakpoints that you activate in the script if you want to play back the script without interruptions but do not want to clear the breakpoints. To do so, select the Ignore Breakpoints option from the Play menu. If you need to stop playback at the breakpoints, select the Ignore Breakpoints option again to disable it.

Inserting a Comment in the Script

You can also use the Comment/Wait option to insert into the Script pane comments about the command line that you chose or general comments about the script, including its purpose. 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, select the Log To Test Manager option.

Playing the Script from the Top

Before you play the script from the top, close all open JD Edwards EnterpriseOne applications. To avoid interference between JD Edwards Autopilot and JD Edwards EnterpriseOne, do not type on the keyboard or click a JD Edwards EnterpriseOne window while a script is executing unless you want to intentionally change the course of the script execution.

Note. Simply moving the mouse will not affect the execution of a script, nor will clicking anywhere in the JD Edwards Autopilot window.

To play the entire script from the top, in the Script pane of the JD Edwards Autopilot window, verify that you have completed these steps:

1. Remove any breakpoints or select Ignore Breakpoints from the Play menu.
2. Disable the Stepping On button.
3. In the Play menu, select the Play from Top option.

   Note. You can also start the playback process by pressing the F5 key on the keyboard.

Playing the Script from a Particular Cursor Position

You use the Play From Cursor option to begin playback from the position of the cursor, rather than from the top of the script. Before you play back the script from cursor, ensure that the JD Edwards EnterpriseOne software is open to the appropriate place. For example, if you are starting playback on a particular form, ensure that the form is open.

To play back the script from a particular cursor position:

1. In the Script pane in the JD Edwards Autopilot window, select a command line from which you want to play back the script.
2. Select the command line.
3. In the Play menu, select the Play From Cursor option.

Playing a Branch of the Script

You can use the Play Branch feature to play a selected branch of a script. JD Edwards Autopilot plays the selected node in the script and all nodes subordinate to it. Coincidentally, the insertion cursor always marks the endpoint of branch execution.

To play a branch of the script:

1. In the Script pane in the JD Edwards Autopilot window, select the first command line in a branch.
2. In the Play menu, select the Play Branch option.

Playing the Script to the Next Command

After you select the Stepping On option, you select a command line from which to play back the script–from the top, from a branch of the script, or from a particular 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 JD Edwards Autopilot window, select a command line from which you want to play back the script.
2. In the Play menu, select the Stepping On option.
3. In the Play menu, select one of these buttons:
   - Play from Top
   - Play Branch
   - Play from Cursor

   JD Edwards 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.

Toggling a Breakpoint

To have the script 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 select and right-click a command line to remove a breakpoint.

To toggle a breakpoint:

1. In the Script pane of the JD Edwards Autopilot window, select a playback breakpoint by selecting a line in the script.
2. Right-click.
3. Select the Toggle Breakpoint option.

   JD Edwards Autopilot inserts the breakpoint to the script.
4. To remove the breakpoint, select a command line where you entered a breakpoint.
5. Right-click.
6. Select the Toggle Breakpoint option.

   JD Edwards Autopilot removes the breakpoint.

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 particular cursor position. When JD Edwards Autopilot reaches the command line that contains the breakpoint, playback halts. However, JD Edwards Autopilot does not cancel playback. To continue scripting, or to play back the script differently, you must click the Stop button to cancel playback.

To play the script to a breakpoint:

1. In the Script pane in the JD Edwards Autopilot window, toggle a breakpoint in the script.
2. In the Play menu, select the Play from Top option.
3. If the script plays to the breakpoint and you want to continue scripting, toggle off the breakpoint.
4. In the Play menu, select Stop.

Continuing Playback to a Breakpoint

After you toggle one or more breakpoints and select stepping, you can play the script back from breakpoint to breakpoint by clicking the Continue to Breakpoint button on the toolbar.

To continue playback to a breakpoint:

1. In the Script pane in JD Edwards Autopilot, select a playback breakpoint by selecting a line in the script.
2. Right-click.

3. Select Toggle Breakpoint to insert the breakpoint in the script.
4. Select a point in the script where you want to play back the script.
5. In the Play menu, select the Stepping On option.
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 necessary and click Continue to Breakpoint each time that playback reaches one.

Ignoring Breakpoints During Script Playback

Access the JD Edwards Autopilot window.

1. In the Script pane, toggle on one or more breakpoints in the script.
2. Click the Ignore Breakpoints button on the JD Edwards Autopilot toolbar.

Inserting a Wait Command in the Script

If you insert a breakpoint in the script, playback halts when JD Edwards Autopilot reaches the breakpoint. Playback does not resume or stop without your intervention. Alternatively, if you select 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 JD Edwards Autopilot window, select a command line in the script.
2. In the Command menu, select the Comment/Wait option.
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.

Failing a Script

You can automatically fail a script by activating the Fail Script option. This option appears in the command pane when you click the Comment/Wait button. Activate 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 JD Edwards Autopilot window, place the insertion cursor at the point that you want to fail the script.
2. In the Command menu, select Comment/Wait.
3. In the command pane, select the Fail Script option and click the Insert button.

JD Edwards 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 insert comments in the script to measure playback transaction time. Setting transaction times in the script provides important information about the time that the software requires to run a series of commands. Although these transaction times have no meaning within JD Edwards Autopilot, they are used by the JD Edwards Virtual Autopilot tool to analyze application performance. Creating a transaction in a JD Edwards Autopilot script causes an entry to be written to the results data, which can be imported into JD Edwards Virtual Autopilot for analysis.

Important! JD Edwards Virtual Autopilot requires a JD Edwards EnterpriseOne Windows client. You can use JD Edwards Virtual Autopilot with JD Edwards EnterpriseOne Tools 8.97 and JD Edwards EnterpriseOne Applications 8.10 and prior. You cannot use JD Edwards Virtual Autopilot with JD Edwards EnterpriseOne Applications 8.11 and later releases, as these releases are on a web client only.

You use the Comment/Wait option 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 the program).

To set transaction times in the script:

1. In the Script pane of the JD Edwards Autopilot window, determine the command line that represents the start of the transaction, and then place the insertion cursor directly below it.
2. In the Command menu, select the Comment/Wait option.
3. In the unpopulated Comment list of the JD Edwards Autopilot command pane, enter *Start,* press SPACEBAR, and then enter a name for the transaction.
4. Click the Insert button.

   JD Edwards 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. In the Command menu, select the Comment/Wait option.
7. In the unpopulated Comment list of the JD Edwards Autopilot command pane, type *End,* press SPACEBAR, and type the name of the transaction.

   JD Edwards 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.

Inserting a Comment in the Script

Access the JD Edwards Autopilot window.

1. In the Script pane, place the insertion cursor at the point in the script in which you want the comment to appear.

2. In the Command menu, select the Comment/Wait option.
3. In the unpopulated Comment list of the JD Edwards Autopilot command pane, type a comment.
4. Select the Log to Test Manager option to have JD Edwards Autopilot include the comment in a summary report after testing.
5. Click Insert.

See Also

JD Edwards EnterpriseOne Tools 8.97 Virtual Autopilot Guide

CHAPTER 9

Creating a Sample JD Edwards Autopilot Script

This chapter provides an overview of a sample script that you can create with Oracle's JD Edwards EnterpriseOne Autopilot and discusses how to create the script.

Understanding the Sample JD Edwards Autopilot Script

Although JD Edwards Autopilot can be used to create a script to verify any application, this sample script uses the A/P Standard Voucher Entry (accounts payable standard voucher entry) application (P0411). This section includes step-by-step instructions for developing a sample script for the application. This sample script does not provide examples of every function or feature of JD Edwards Autopilot. For example, this script tests an interactive application and does not launch a universal batch engine (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 depend on your use of a particular application.

Creating the Sample JD Edwards Autopilot Script

This section provides an overview of sample script creation and discusses how to:

- Launch an application and form.
- Declare a variable.
- Add a new form.
- Type data in a header control.
- Create a valid values list.
- Type data in a grid column.
- Update the repeat count.
- Update the database.
- Set the value of a variable.
- Return to a previous form.
- Enter data to a Query By Example (QBE) line.
- Find records.

- Select records and delete them from the database.
- Complete the script.

Understanding Sample Script Creation

You use JD Edwards Autopilot to create customized scripts that apply to the applications that you most often use. The sample script presented in this chapter illustrates how you use many of the commands that are included in JD Edwards Autopilot. However, the sample script does not include all context and action commands, and it does not represent a definitive method for using JD Edwards Autopilot. For example, this script tests functions that you perform in an interactive application. You can also test the launch and submission of a UBE, which requires that you write a different set of commands.

Launching an Application and Form

Launching the application from JD Edwards Autopilot establishes one basic scripting context that you need for many other scripting commands that you insert and run. Launching an application does not have to be the first command that you script in a JD Edwards Autopilot session. However, the Application command is often at least one of the first commands that you script.

Selecting an application in JD Edwards Autopilot also requires that you select a form that is part of the application. After you launch an application from JD Edwards 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 clicking buttons to access different forms or to perform functions within the active form.

To launch an application and form:

1. From the desktop or the appropriate directory, launch JD Edwards Autopilot.

   The JD Edwards Autopilot splash screen appears, followed by the JD Edwards Autopilot window.

2. In the File menu, select the New option.

   The command pane and Script pane are unpopulated.

3. In the Command menu, select Application.

   The Application list is populated, while the Menu list remains unpopulated.

4. Click an application code, such as P0411 for the A/P Standard Voucher Entry application.

   JD Edwards Autopilot populates the Menu list with items that appear under these headings:

   - Fast Path, which corresponds to the fast path command that JD Edwards 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, such as 3/G0411, Standard Voucher Entry, version ZJDE0001.
6. Click the Insert button in the command pane.

Note. If the Playback button on the toolbar is activated, JD Edwards Autopilot launches the application, and the form specified in the Fast Path appears (in this case, the Supplier Ledger Inquiry form).

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 can place it at the top of the script to make it global. That way, 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 declare a variable after you launch an application, you can 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. In the Command menu, select Variables.
2. Type a name for the variable in the New Variable field.

   For the sample script, call the variable *Previous Document #*.
3. In the Source of Value list, select Unknown/None.

   Note. Selecting Unknown/None indicates that you merely want to give the variable a name at this point. You do not yet set its value.
4. Click the Insert button.

   JD Edwards 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. 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 move from the Supplier Ledger Inquiry form to the Enter Voucher - Payment Information form. To do so, you script clicking the Add button to change forms. You also select Enter Voucher - Payment Information from the Next Form list so that the Form command line in JD Edwards Autopilot matches the active form.

To add a new form:

1. From the Command menu, with the Playback button activated, select the Press Toolbar Button option.
2. In the Button list, select the Press Standard Button option.

   Note. The options listed for Press Standard Button in JD Edwards Autopilot correspond to the buttons on the toolbar in the form.

3. Click Add.
4. In the Next Form list, select the name of the form that appears in the software when you click Add–the Enter Voucher - Payment Information form in this example.
5. Click the Insert button.

   JD Edwards Autopilot adds the commands that you insert in the Script pane. In the playback mode, the Enter Voucher - Payment Information form appears in the software.

   Note. If you select the Unknown/None option from the Next Form list (step 4), you must complete these 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 JD Edwards 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 Playback button activated so that the software is active, note the name of the software form that is active–the Enter Voucher - Payment Information form in this example.
7. In the Command menu, select Form.

   The system displays a list of all forms within the current application.
8. Select the name of the active form from the Form list.

   In this example, select Enter Voucher - Payment Information.
9. Click Insert.

   In the Script pane, a Form command line appears that includes the name of the selected form.

Typing Data in a Header Control

Continue the sample script by adding commands to type inputs in header controls. Like the Application and Form commands, the Header command is a context command; it establishes the context in which you take further actions.

To type data in a header control:

1. In the Command menu, select the Set Header Control Value option.
2. In the Header Control list, select the name of a header control where you want to input data–for example, Company.

   Note. The header control that you select in JD Edwards Autopilot is highlighted with a BlueCue.

   ---

3. Select one of these as the input source for the control:
   - Literal Value
   - Valid Values List
   - Variable
   - UDC Visual Assist Value
   - Form Interconnect Visual Assist Value
   - Clear

- Set Focus

  In this sample script, select Literal Value. When you select an input source, JD Edwards Autopilot provides a caption for the value selection list. The caption depends on the source of input.

4. Click inside the unpopulated Literal Value list and type *1*.
5. Click the Insert button.

   JD Edwards Autopilot types the scripted input in the Company header control in the Enter Voucher - Payment Information form. JD Edwards Autopilot encloses literal values in quotes in the Script pane.
6. Continue to script inputs in header controls by selecting the Set Header Control Value option in the Command menu, and then selecting a control, a source of input, and a value.

Creating a Valid Values List

For this sample script, you script inputs in the header control Long Address Number. However, instead of entering a literal value in the control, use a valid values list as the input source.

A valid values list consists of values that you collect and store under a specified name. You use a valid values list if, for example, you need to input more than one value in a header or grid column. These 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, JD Edwards Autopilot draws the values from a database that you select. 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; JD Edwards 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, select a literal value. However, if the value is likely to change, you can create a valid values list so that JD Edwards 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. In the Tools menu, select Generate Valid Values List.
2. Select the List of Literal Values option in the Select Data File Type dialog box.
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 appear vertically stacked in the box.
6. Click Finish.
7. In the Command menu, select Set Header Control Value.
8. In the Form/Subform Hierarchy list, select the appropriate form or subform.
9. In the Header Control list, select the name of a header control where you want to input data; for example, Long Address Number.

10. Select Valid Values List as the source of input.
11. Select the name Vendors, which is the name of the valid values list that you created.
12. Click the Insert button.

    JD Edwards Autopilot types the first value in the list in the Long Address Number control in the Enter Voucher - Payment Information form. JD Edwards 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.
13. Complete these 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 named 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

Next, script inputs in the grid columns. For this sample script, make voucher payment inputs in the grid columns. The Grid command, like the Header command, is a context command; it establishes the grid column in the form as the environment where you take additional actions.

The names of the grid columns in the Grid Column list of the JD Edwards Autopilot command pane match the names of the grid columns on the form. When you select 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, select Set Grid Cell Value.

   The command pane includes these lists:
   - Grid Column
   - Source of Input
3. In the Form/Subform Hierarchy list, select the appropriate form or subform
4. Select the name of a grid column from the Grid Column list; for example, select Gross Amount.
5. Select Valid Values List as the source of input.
6. Select Random Dollar Amounts.
7. Click the Insert button.
8. Repeat steps 1 through 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 the source of input includes two lists of literal values that contain five values each, you 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 activated, disable it by clicking the Playback button in the toolbar.
2. In the Script pane, select Detail Information, which is the node that you need to update.

   In the command pane, review these lists:

   - Define repeat count from
   - Repeat Count
3. Select Literal Value from the Define repeat count list.
4. In the Repeat Count list, type a number.

   In this case, you type *5,* because you want to script entering five separate values that you included in the valid values lists.
5. Click Update.

   Note. The repeat count in the Detail Information node is now 5. If you play back the script, JD Edwards Autopilot loops through this node five times. Each time it loops, JD Edwards Autopilot inserts in the Gross Amount and Remark grid columns a different value from the valid values lists.
6. Click the Playback button on the toolbar to select the playback function.

Updating the Database

After you script the entries in the Enter Voucher - Payment Information form, you 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.

Note. Before preforming these steps, 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 Chapter 6, “Scripting Actions,” Creating a List of Literal Values , page 64.

To update the database and confirm a new form:

1. In the Command menu, select the Press Toolbar Button option.

   Review these 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 these buttons on a form:

   - OK
   - Delete
   - Cancel
2. Select the OK option from the Button list.
3. From the Next Form list, select the form that follows when a user clicks OK for this application and version.

   For the sample script, you select 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 these tasks:

   a. Enter an account number in the appropriate grid column in the G/L Distribution form.
   b. Click the OK button to update the database.
   c. 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. In the Command menu, select Set Grid Cell Value.
3. In the Grid Column list, select Account Number.
4. Select Valid Values List as the source of input.
5. Select 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. In the Command menu, select the Press Toolbar Button option.
8. Select the Standard Button option.
9. Select OK.
10. In the Next Form list, select 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. You accomplish this 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 select.

To set the value of a declared variable:

1. In the Command menu, select Variables.
2. In the Existing Variable list, select the name of the variable that you declared earlier in the script.
3. In the Source of Value list, select one of these:
   - 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, select Header Control Data as the source of value.
4. Select or type a value.

   In this case, JD Edwards 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. Select Previous Document.
5. Click Insert.

   You have now assigned a value to the variable that you named Previous Document #. You have told JD Edwards Autopilot to derive that value from the header control Previous Document in the form Enter Voucher - Payment Information. JD Edwards Autopilot stores that value in the declared variable. Because you previously made the variable global, you can use this value at any point that you select 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. Select the Press Toolbar Button option from the Command menu and select Cancel from the Press Standard Button options in the Button list. You then select the next form that appears when you click the Cancel button. Therefore, you select Supplier Ledger Inquiry from the Next Form list.

To return to a previous form:

1. In the Command menu, select Press Toolbar Button.
2. Select the Cancel option from Standard Button in the Button list.

3. Select the Supplier Ledger Inquiry form from the Next Form list.
4. Click the Insert button.

   In the software, the Supplier Ledger Inquiry form becomes active. In the JD Edwards Autopilot Script pane, the Form command line displays 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 clicking 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. In the Command menu, select Set QBE Cell Value.
2. Select the grid column Document Voucher Invoice Entry from the Grid Column list.
3. Select Variable from the Source of Input list.
4. In the value selection list, select the name of the variable that you declared; remember, it also now contains the value that you set.
5. Click the Insert button.

   JD Edwards 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 select another form from the Next Form list before clicking the Insert button.

To find records:

1. In the Command menu, select the Press Toolbar Button option.
2. Select Standard Button in the Button list, and then select Find.

   You do not need to select an option from the Next Form list because you are remaining on the Supplier Ledger Inquiry form.
3. Click the Insert button.

   The grid contains 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

Next, script the deletion of records from the database. These records appear in the detail area of the Supplier Ledger Inquiry form because you scripted clicking 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. JD Edwards Autopilot automatically clicks OK in 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. In the Command menu, select the Select Grid Row option.
2. In the Form/Subform Hierarchy list, select the appropriate form or subform
3. In the command pane, select from the Source of Row Number list.
4. Type a literal value of *1* in the value selection list.

   By typing this value, you select the row that contains the record that you want to delete.
5. Select the single-click option in the Action on grid row list.

   This command selects the row in the detail area of the grid.
6. Click the Insert button.
7. In the Command menu, select the Press Toolbar Button option.
8. Click Select Standard Button in the Button list, and then select Delete.
9. Click the Insert button.

   JD Edwards 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 need to delete.

Completing the Script

At the conclusion of the scripting, exit all open applications. For the sample script, click the Cancel button. Because you started on the Supplier Ledger Inquiry form, this command enables you to exit the A/P Standard Voucher Entry program (P0411) and complete the script.

To complete the script:

1. In the Command menu, select the Press Toolbar Button option.
2. Select Standard Button, and then select Cancel.

   Because you are exiting the application, do not select an option from the Next Form list.
3. Click Insert.

Note. By clicking Cancel, you return to the starting point of JD Edwards EnterpriseOne software. You should end each scripting session by clicking Cancel or Close, especially if you intend to create additional scripts. Leaving windows open can impede playback of scripts and make it difficult to write additional scripts.

---

4. In the File menu, select 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, selected a form and version associated with the application, and accessed forms by clicking 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 selected 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. At the end, you canceled the application and saved the script.

CHAPTER 10

Storing Scripts and Test Results

This chapter provides an overview of storing scripts and test results and discusses how to:

- Work with the script repository.
- Work with script reporting.
- Work with JD Edwards Autopilot Test Manager.
- Manage script testing.

Understanding Storing Scripts and Test Results

Scripts that you write are reusable, dynamic objects that continue to be useful after you complete them. Oracle's JD Edwards EnterpriseOne Autopilot enables you to write and run scripts. You can also include those scripts in a base of knowledge about the software and manage batch testing of scripts.

The script repository is a key component of the JD Edwards Autopilot knowledge base. The repository is a database of scripts. It 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 can contribute to it. Finally, the database is organized because you can assign defining properties to each script that you reposit. These properties enable you to categorize scripts by application, for example.

Capturing and storing test results is another important way in which JD Edwards Autopilot enables you to build a knowledge base about the software. If you configure JD Edwards Autopilot to capture playback results, it generates an event stream during playback. The event stream is a chronological, time-stamped record of JD Edwards Autopilot and JD Edwards EnterpriseOne events that occur during playback. JD Edwards Autopilot stores these test results locally and in a repository, the F97214 table. You can use these results to troubleshoot JD Edwards EnterpriseOne processes. For example, you might identify a processing error or isolate an error message.

The results repository is an important part of the automated testing process. To analyze playback events in detail, you can import an event stream to JD Edwards Virtual Autopilot Script Editor, which is part of JD Edwards Virtual Autopilot. Using JD Edwards Virtual Autopilot Script Editor, you can generate from the event stream a virtual script. You can use the virtual script on a single workstation to simulate multiple users.

Important! JD Edwards Virtual Autopilot requires a JD Edwards EnterpriseOne Windows client. You can use JD Edwards Virtual Autopilot with JD Edwards EnterpriseOne Tools 8.97 and JD Edwards EnterpriseOne Applications 8.10 and prior. You cannot use JD Edwards Virtual Autopilot with JD Edwards EnterpriseOne Applications 8.11 and later releases, as these releases are on a web client only.

JD Edwards Autopilot also enables you to manage the testing of scripts. Using Test Manager, you can create playlists of locally saved and reposited scripts and conduct batch testing. This frees you from the time-consuming task of running one test at a time. You can use Test Manager to conduct testing of an entire suite of applications in one session.

JD Edwards Autopilot Repositories

Scripts that you write are reusable, dynamic objects that continue to be useful after you complete them. JD Edwards Autopilot enables you to write and run scripts. You can also include those scripts in a base of knowledge about the software and manage batch testing of scripts.

The script repository is a key component of the JD Edwards Autopilot knowledge base. The repository is a database of scripts. It 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 can contribute to it. Finally, the database is organized because you can assign defining properties to each script that you reposit. These properties enable you to categorize scripts by application, for example.

Capturing and storing test results is another important way in which JD Edwards Autopilot enables you to build a knowledge base about the software. If you configure JD Edwards Autopilot to capture playback results, it generates an event stream during playback. The event stream is a chronological, time-stamped record of JD Edwards Autopilot and JD Edwards EnterpriseOne events that occur during playback. JD Edwards Autopilot stores these test results locally and in a repository, the F97214 table. You can use these results to troubleshoot JD Edwards EnterpriseOne processes. For example, you might identify a processing error or isolate an error message.

The results repository is an important part of the automated testing process. You can import an event stream from the repository to JD Edwards Virtual Autopilot Script Editor, which is part of JD Edwards Virtual Autopilot. Using JD Edwards Virtual Autopilot Script Editor, you can generate from the event stream a virtual script. You can use the virtual script on a single workstation to simulate multiple users.

JD Edwards Autopilot also enables you to manage the testing of scripts. Using Test Manager, you can create playlists of locally saved and reposited scripts and conduct batch testing. This frees you from the time-consuming task of running one test at a time. You can use Test Manager to conduct testing of an entire suite of applications in one session.

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, like a library, the script repository acquires new materials. Each time that you create 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 use.

Unlike a library, you can modify the materials that you remove from the repository. For example, you might check out a script from the repository to modify it. The JD Edwards Autopilot script repository enables a dynamic interchange between the people who use it.

You can use the script repository to build a database of scripts. You use JD Edwards Autopilot to:

- Categorize scripts according to a set of user defined 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.

See Also

JD Edwards EnterpriseOne Tools 8.97 Virtual AutoPilot Guide

Script Categorization

Like a librarian who adds books to a library 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 users who check out a script, and they provide a way for you to categorize scripts and make them easier to find.

Categorizing scripts also facilitates running batch test scripts using 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 selecting Properties from the File menu. The Script Properties form contains controls with scroll buttons.

The combo boxes contain user defined values. You select from these values to categorize a 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 table F0004 and 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). This table lists the relevant user defined codes (UDCs) that JD Edwards 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 JD Edwards 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.

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 also enables you to classify a script as part of a large-scale testing effort. For example, you can designate these script properties to include the script in a suite of scripts:

- System code.
- Department.
- General use, such as benchmarking.
- Detailed use, such as batch applications.

You document the properties of a script by entering information in the Script Properties form. When you save the script, JD Edwards Autopilot saves the information along with the script.

When you add the script to the repository, JD Edwards 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 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 the script.

The information that you enter on this tab provides baseline information about the script and its origins. You enter data in these fields on the General tab of the Script Properties form:

Field	Description
Title	Script title that JD Edwards 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 the 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.

Field	Description
General	The general testing purpose of the script, such as benchmarking. The values for this parameter are user defined.
Detail	The particular 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.

These fields enable 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 activating the Address by Effective Date feature. The values for this parameter are user defined.

Select one of these 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

Select the automatic validation option if simply running the script successfully means that the functions worked, and you do not need to further test the results. However, in some cases, such as when you test universal batch engines (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 select the Manual option, JD Edwards Autopilot enables these additional options:

- Report
- Screen Prints
- Log Comments

Selecting one or more of these options reminds people who run the script to manually review the output after the script runs.

In addition, you can indicate whether the script resets changes that you made to constants or to master file data, such as additions to or deletions from the F4101 table.

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 is relevant.

Categories Tab

The Categories tab enables you to categorize the object of that you are testing. A 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–can indicate the type of verification testing that the script performs.

Naming Conventions for Saved Scripts

No predetermined rules exist for naming scripts that you add to the repository. Use a particular naming convention for all scripts. For example, you might give the script a title that identifies the application, release, or function that it tests. In addition, you might specify whether the test is new or a retest. Following a naming convention enables other users to identify the purpose of a script. Assigning properties to the script enables you to accurately subcategorize it; adhering to a naming convention furthers that goal.

See Also

Chapter 7, “Working with the Script Pane,” Understanding Script Retention, page 112

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. However, you cannot specify the owner because JD Edwards 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 enables you to search for scripts in the repository. You can make the search as narrow or as broad as necessary. After you have found the script that you need, you can create a copy of it or check it out of the repository.

You use these two forms with the Browse Repository Scripts command:

- Select Script
- Browse Scripts

You use the Select Script form to establish search criteria. The Browse Scripts form contains information about the scripts in the repository that have properties that match the criteria specified in the Select Script form.

Select Script Form

The Select Script form contains the same tabs and fields as the Script Properties form. However, it is a query form rather than a form for entering script properties.

You can select any of the tabs, except Comments, and enter data in the fields or select options to establish search criteria for a particular type of script. Alternatively, you can enter the exact title of a script. JD Edwards Autopilot matches the criteria that you set and the entries in the Script Properties pages. If you do not enter any field information, JD Edwards Autopilot includes all 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–enable you to use wildcards (asterisks) to have JD Edwards Autopilot include all scripts in the search.

You can use the asterisks alone or with an entry in a field. For example, to find all scripts that tested the 8.11 release, you might enter **811** in the Title field. JD Edwards Autopilot includes in its search all scripts that contain 811 in the title, regardless of any text that comes before or after 811.

If you enter information in a field, JD Edwards Autopilot automatically appends the wildcard to the text string. For example, if you enter *P0911* in the Main Application field, JD Edwards 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, JD Edwards 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 all scripts that match the data specified in the Select Script form. The Browse Scripts form contains these 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 field identifies the workstation that the user used to check out the script. The Security field 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 the owner and security level cannot be changed.
No Checkout/in	People who do not own the script can only create a copy of it and save changes locally.
No Access	People who do not own the script can only see that it resides in the repository.

The form also contains these buttons:

- Get Copy, which enables you to create a copy of a script.
- Checkout, which enables you to check out a script from the repository.
- Undo Checkout, which enables you to undo a script checkout.
- Delete, which enables you to delete a script from the repository if you are authorized to do so.
- Close, which enables you to exit from the Browse Scripts form.

JD Edwards Autopilot disables the Get Copy and Checkout buttons until you select the title of a script. After you select 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 selecting 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 create a copy of or check out a single script. You can also create multiple copies or check out more than one script. You can open the checked-out copy, check it in, or close the form. If you close the form, 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.

Deletion of Scripts

You can delete from the repository any script that you own. However, these restrictions exist:

- You cannot delete a script that you do not own unless you have authorization.
- You must enter a password that changes daily 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 create a copy of a script that resides in the repository. JD Edwards Autopilot enables you to view and run a copy of a script that you obtain from the repository, but you cannot make any permanent changes to it without first saving it as a local copy in your script directory.

JD Edwards 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 create a copy of a reposited script, the copy opens in JD Edwards Autopilot. The window title bar contains the word Repository, and the title and description of the reposited script.

If you modify the copy and then click Save, JD Edwards Autopilot displays an error message indicating that the script is not checked out and that you must either check it out or save it as a local file before the changes take effect.

Checkout Command

You might retrieve a script from the repository to modify it. To do so, you must check out the script, make and save the changes, and then check it back into the repository. JD Edwards Autopilot returns the script to the repository with the changes intact. The repository then contains a new version of the script.

When you check out a script from the repository, you check out the latest version of the script. A script cannot be checked out to more than one person at a time. This prevents two people from making changes to the script simultaneously.

You can also run the Checkout command when you are browsing scripts. In the Select Script form, you can limit the script search by selecting options and entering information on the various tabs.

Undo Checkout Command

You can undo the Checkout command if, for example, you change an existing script but decide to cancel the changes. If you check in the script, the changes take effect, and the repository contains a new version of the script. If you use the Undo Checkout command, none of the changes that you make to the script take effect. To undo the checkout, select File, Repository, Check In/Check Out, Undo Check Out. A form confirms the undo checkout.

My Checkouts Form

You can find scripts that you have checked out by using the My Checkouts form. The My Checkouts form enables you to keep track of the scripts that you check out to ensure that you check them back in. The My Checkouts form contains the same headings as the Browse Scripts form. Each document button next to the script title displays 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 displays all of your checkouts on the current machine. If you check out a script on a machine other than the one on which you are currently working, the document button 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 make any necessary changes to the reposited version. When you are finished with the changes, you must check in the script for them to take effect. When the script is checked in to the repository, the new version is available to other people. When you check in a script, it automatically closes and enters the repository, just as when you add a script to the repository.

Where Included Command

You sometimes need to determine if a script is included by 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, the Where Included form displays the titles and descriptions of any reposited scripts that include the script title.

From the Where Included form, you can select a script and create a copy of it or check it out. The script that you select from this form is a master script; 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.

Working with the Script Repository

This section provides an overview of the script repository and discusses how to:

- Assign properties to a script.
- Add a script to the repository.
- Browse for repository scripts.
- Delete a script from the repository.
- Assign security to a reposited script.
- Get a copy of a script.
- Check out a script.
- Undo script checkout.
- Check in a script.
- Query for included scripts.
- Use a Command Line to Load a Repository Script.

Understanding the Script Repository

The JD Edwards Autopilot script repository enables you to retrieve scripts for study, playback, and modification. You can add scripts that you create, and others can 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 other people's scripts for the same purpose. You can also check out scripts from the repository, change them, and store the new versions by checking them in.

The script repository works in conjunction with the other components of JD Edwards 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 can 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 assign to it. These identifying features facilitate 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 JD Edwards Autopilot window, open a script.
2. In the File menu, select the Properties option.

   The Script Properties form appears.
3. In the Script Properties form, enter information or select options in the fields on each of these tabs:
   - General
   - Details
   - Comments
   - Categories
4. Click OK.
5. In the File menu, select Save or Save As and assign a title that follows your predetermined naming convention.

Adding a Script to the Repository

After you save a script (and any assigned properties), you can add it to the repository. The script must be open before you can add it. When you select the Add to Repository command, JD Edwards Autopilot enables you to assign script properties for the first time or to modify previously assigned properties. After you add the script to the repository, JD Edwards Autopilot identifies the script by the title and description that you enter.

To add a script to the repository:

1. In the JD Edwards Autopilot window, open the script that you want to add to the repository.
2. In the File menu, select Repository, Add to Repository.

   The Add Script to Repository form appears.
3. On the Add Script to Repository form, enter any script properties that you want to assign.
4. Click OK.

   JD Edwards 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 to run, use as a template, or modify. You can view all of the scripts in the repository or enter search criteria in the Select Script form to search for scripts of a certain type. You enter information and select options to establish search criteria. JD Edwards Autopilot uses the criteria to display any scripts that have matching properties.

To browse for scripts:

1. In the File menu of the JD Edwards Autopilot window, select 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. On the Select Script form, establish criteria for the scripts to retrieve.
3. Click OK.

   The Browse Scripts form appears and displays the titles of the scripts that match the specified criteria.

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 precaution, JD Edwards Autopilot prompts 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. In the File menu of the JD Edwards Autopilot window, select Repository, and then Browse Repository Scripts.
2. On 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 of the Select Script form.

   ---
3. On the Browse Scripts form, select the title of at least one script that you want to delete from the repository and click the Delete button.

   A JD Edwards Autopilot window prompts you to confirm the deletion of the script. If you click Yes when the script is open, JD Edwards Autopilot closes the script.
4. Click OK.

   Note. The deletion fails if the script 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 other people's access to 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 by using the My Checkouts form if you have checked out the script from the repository.

To assign security to a reposited script:

1. In the File menu of the JD Edwards Autopilot window, select Repository, and then Browse Repository Scripts.
2. On the Select Script form, enter your user ID in the Owner field on the General tab and click OK.
3. On the Browse Scripts form, select one or more scripts and right-click.

   The system displays a pop-up menu that contains four security-level options.
4. From the pop-up menu, select one of these security levels:
   - No Restrictions
   - Owner Locked
   - No Checkout/in
   - No Access

Getting a Copy of a Script

You can use the Browse Scripts form to create a copy of a script to play back or to use as a template for creating another script. Note these points about script copies:

- 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 create copies of more than one script if one of the scripts that you select is checked out.
- If you select only the checked-out script, you create a copy of the script, including the changes made since the last checkout.

To create a copy of a script:

1. From the File menu of the JD Edwards Autopilot window, select Repository, and then select Browse Repository Scripts.
2. On the Select Script form, establish criteria for the types of scripts to retrieve.
3. Click OK.
4. In the Browse Scripts form, select at least one script.
5. Click the Get Copy button.

Checking Out a Script

You can check out scripts that are 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, and then check it back in. JD Edwards Autopilot saves all changes and creates a new version without prompting you to add the script to the repository.

If someone has checked out a script, the document button 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 select multiple scripts and one of them has already been checked out, JD Edwards Autopilot disables the Get Copy and Checkout commands. If you select 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 JD Edwards Autopilot window, select Repository and Browse Repository Scripts.
2. On the Select Script form, establish criteria for the types of scripts to retrieve.
3. Click OK.
4. On Browse Scripts, select one or more scripts; and then click Check Out.

Undoing Script Checkout

You can undo a script checkout if, for example, you make changes to the script but then decide not to save the changes.

To undo a script checkout:

1. From the File menu of the JD Edwards Autopilot window with a checked-out script open, select Repository, Check In/Check Out, Undo Check Out.

   A JD Edwards Autopilot window appears prompting you to confirm the checkout cancellation.
2. Click Yes.

Checking in a Script

If you check out a script and make changes to it, you can check it back in to save the changes in the repository.

From the File menu of the JD Edwards Autopilot window, with a checked-out script open, select Repository, Check In/Check Out, Check In.

Note. You can identify all of the scripts that you have checked out by selecting File, Repository, My Checkouts.

The script closes. JD Edwards Autopilot checks in the new version of the script.

Querying for Included Scripts

You can use the Where Included form to search the repository for any scripts in which a particular script is included. If a script is included in another reposited script, you cannot delete it from the repository.

To query for included scripts:

1. From the File menu of the JD Edwards Autopilot window, select Repository, Where Included.
2. On the Where Included form, enter the title of a script and click OK.

   JD Edwards Autopilot displays the titles of all scripts that include the script for which you entered the title. You can check out or create a copy of these scripts.

Using a Command Line to Load a Repository Script

You can load any repository script into JD Edwards 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 JD Edwards Autopilot executable file and identifies the specific repository script to load into JD Edwards Autopilot. The .atr extension indicates to JD Edwards 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.

Working with Script Reporting

This section provides an overview of script reporting and discusses how to:

- Use the event stream.
- Use the Test Results form.

Understanding Script Reporting

The script repository contains information about scripts that test particular applications and processes. The architecture of JD Edwards Autopilot also contains a results repository, the F97214 table. This repository contains information about the actual JD Edwards Autopilot and JD Edwards EnterpriseOne events that occur during script playback.

If you configure JD Edwards Autopilot to capture and store playback events, it records each event using internally placed code and code in JD Edwards EnterpriseOne software. When playback finishes, JD Edwards Autopilot sends the record of events, called the *event stream,* to the repository. You can view each event stream on the Test Results form.

Using the Event Stream

The event stream provides a timeline 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 enables you to troubleshoot problems that occur during playback.

You also capture an event stream when you want to use JD Edwards Virtual Autopilot to generate a virtual script that you can use to simulate multiple users on a single workstation.

Important! JD Edwards Virtual Autopilot requires a JD Edwards EnterpriseOne Windows client. You can use JD Edwards Virtual Autopilot with JD Edwards EnterpriseOne Tools 8.97 and JD Edwards EnterpriseOne Applications 8.10 and prior. You cannot use JD Edwards Virtual Autopilot with JD Edwards EnterpriseOne Applications 8.11 and later releases, as these releases are on a web client only.

Using the Test Results Form

If you have configured script playback to capture, save, and display results, the Test Results form appears when playback finishes. The Test Results form displays playback data such as the event stream, which is a chronological listing of each event that occurs 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 these tabs:

- Browse Result Sets
- Summary
- JDE.INI
- JDE.LOG
- JDEBUG.LOG
- Screen Captures
- Messages
- Results

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 check mark beside a test indicates that it was successful; an *X* indicates a test that failed or was canceled.

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 enables you to filter the saved test results using the columns in the form as criteria. You use the Filter form to select a filter criterion.

Summary Tab

Clicking the Summary tab displays these properties for each test that you run:

- Script
- Machine
- Release
- Environment
- User
- Start time
- End time
- Elapsed playback time
- Status of the playback

JDE.INI Tab

The JDE.INI tab enables you to view the initialized settings for JD Edwards EnterpriseOne software that existed before JD Edwards Autopilot played the script. JD Edwards 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.

Jde.log Tab

After script playback, JD Edwards 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 to track errors that occur during processing.

Screen Captures Tab

If a script fails, JD Edwards Autopilot captures the screen that is active when the script fails and lists the screen shot on the Screen Captures tab.

Jdedebug.log Tab

The jdedebug.log tab displays the jdedebug.log file that JD Edwards 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 occur during script playback.

Messages Tab

Data that appears on the Messages tab of the Test Results form summarizes the script that JD Edwards Autopilot played back.

You can review each context command and action command that you write in the script, as well as any error messages that JD Edwards Autopilot generates. In addition, you can review any error messages that JD Edwards EnterpriseOne software generates during playback.

On the Messages tab, you can filter, print, and export test results. You can filter test results for:

- A particular point during playback that an event occurred
- A particular type of event, such as a message or an action in JD Edwards Autopilot
- A text description of the event in the Test Results form

You can print the test results, provided that you have set up a default printer to do so. Using the Export button, you can export test results to a spreadsheet.

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 enables 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.

See Also

Options for Configuring JD Edwards Autopilot in the *JD Edwards EnterpriseOne Autopilot Guide*

Results in the *JD Edwards EnterpriseOne Autopilot Guide*

Working with JD Edwards Autopilot Test Manager

This section provides an overview of Test Manager and discusses how to:

- Use the script display pane.
- Use the script storage pane.
- Use the test results pane.
- Use the Test Manager toolbar.

Understanding Test Manager

JD Edwards Autopilot Test Manager enables you to test multiple JD Edwards Autopilot scripts in a batch, so that you can review results quickly and gather test results for archiving. You use Test Manager to create a playlist that contains scripts that you save on your local drive, scripts that you retrieve from the script repository, or a combination of both.

After you assemble a playlist, you run it. Test Manager launches JD Edwards Autopilot, which in turn launches JD Edwards EnterpriseOne software. Test Manager runs the playlist to completion, closing JD Edwards Autopilot each time a script completes and opening JD Edwards Autopilot immediately after the next script appears in the queue. Test Manager displays a test status of Failure, Success, or Incomplete for each script that runs.

You can view the results of each script playback. These results include messages that enable you to analyze the cause of a script failure. In addition, you can save the playlists that you create, edit them, and replay them.

Using the Script Display Pane

The script display pane in Test Manager is the area where you select scripts to assemble a playlist. The pane contains two tabs, Local and Repository. When you select the Local tab, Test Manager displays all the scripts that you have stored on the local drive.

When you select 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 in to 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. 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. 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

Chapter 10, “Storing Scripts and Test Results,” Property Pages for Scripts, page 148

Chapter 10, “Storing Scripts and Test Results,” Assigning Properties to a Script, page 155

Using the 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, which means that you have added it to the playlist but have not yet run it.

You can remove scripts from the script storage pane. When you remove a script from the playlist, Test Manager prompts you to confirm the action.

After you assemble the playlist, you run it by clicking the Run button on the toolbar. Test Manager launches JD Edwards Autopilot and runs the scripts in the order that they appear in the script storage pane. After the script runs, Test Manager displays one of these states, depending on the results of the test: Success, Failure, Cancellation, or Incomplete.

Using the 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 on the toolbar. The test result summary displays this information about the test:

- Total number of tests generated.
- Status breakdown, including the number of scripts that failed, succeeded, were canceled, or did not finish.
- Name of the client machine.
- Environment in which the test was run.
- 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 that 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 appeared during playback.

In addition, Test Manager provides information about warning messages. This table lists some of the message types that can appear in the test results pane and summarizes their meanings:

Message Type	Description
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 JD Edwards 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. 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. JD Edwards Autopilot failed to set processing option text, which might cause a failure later in the script.

Using the Test Manager Toolbar

The Test Manager toolbar enables you to control a test session and view its results. The Test Manager toolbar contains these 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 JD Edwards Autopilot.
Log	Display the Test Results form, which contains detailed summaries of each test that you ran and saved in JD Edwards Autopilot.
Report	Populate 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

This section provides an overview of script testing and discusses how to:

- Create a playlist.
- Save a playlist.
- Run a test.
- View test results.
- Reset a test.

Understanding Script Testing

You use JD Edwards Autopilot Test Manager to create a playlist from scripts that reside on your local drive or in the script repository. Test Manager enables you to run a playlist multiple times, without intervention. The playlist can contain a combination 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

The first task in Test Manager is to create a playlist. You retrieve the scripts for your playlist from your local drive, the script repository, or both. 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 the desktop or in the directory where you store Test Manager, click the Test Manager executable file.

   The Test Manager splash screen appears, followed by the JD Edwards Autopilot Test Manager form.

   Note. When the Test Manager form appears, the script display pane and the script storage pane might not appear. Drag the splitter bar that separates them by using the grabber, which is represented by a pair of vertical bars.

2. In the script display pane of the JD Edwards Autopilot Test Manager form, select either the Local or the Repository tab.
3. If you select the Local tab, select a local script from the script display pane and click the Add button on the toolbar.

   Test Manager adds the selected test to the script storage pane.

Note. You can select more than one test by selecting a script in the script display pane, holding down either the CTRL or the SHIFT key, and selecting additional scripts.

4. Select the Repository tab.
5. Click the Repository Filter button.

   Test Manager displays the Select Script form.
6. In the Select Script form, select script criteria to narrow the number of scripts that you copy from the repository, and then click OK.

   Test Manager displays in the script display pane any repository scripts that match the search criteria.
7. Select one or more repository scripts from the script display pane, and then click the Add button on the toolbar.
8. Continue adding local and repository scripts until you have created the playlist.
9. To change the sequence of the scripts, click a script in the script storage pane and click the Up or Down button on the toolbar.
10. To remove a script from the playlist, select it in the script storage pane and click the Remove button on the toolbar.

Saving a Playlist

After you create a playlist, you can save it, or you can run the test before you save it. Remember, if you do not save the playlist, Test Manager prompts you to do so when you exit the form.

To save a playlist:

1. In the File menu of the JD Edwards Autopilot Test Manager form, select Save or Save As.
2. Assign the playlist a filename and save it to a drive and directory, and click Save.

Note. 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 JD Edwards Autopilot, and then runs each script in the queue in the order that you set up in the script storage pane.

Test Manager launches JD Edwards Autopilot, minimizes the JD Edwards Autopilot window, and then begins running the first script in the queue. As each test finishes, Test Manager displays its result in the script storage pane. When Test Manager finishes running a script, it closes JD Edwards Autopilot, and then relaunches it with the beginning of the next script in the queue.

As each script finishes running, Test Manager displays the result of Success, Failure, Canceled, or Incomplete.

To run a test:

1. In the File menu of the JD Edwards Autopilot Test Manager form, select Open.
2. Open the drive, directory, and file in which you store your playlists, select one or more playlists, and click Open.

3. On the toolbar of the JD Edwards Autopilot Test Manager form, click Run.

   Note. During playback, JD Edwards EnterpriseOne software remains open, unless a script contains an Exit JD Edwards command, in which case the software closes. In this case, with the beginning of the next script in the queue, JD Edwards Autopilot Test Manager launches JD Edwards Autopilot, which launches JD Edwards EnterpriseOne software.

4. To stop the test, click the Stop button on 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 run and save, click the Log button.

To view test results:

1. On the JD Edwards Autopilot Test Manager form, open a saved playlist.
2. After JD Edwards Autopilot Test Manager completes the playlist, click the Report button on the toolbar.

   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 on the toolbar.

   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 on the Test Results form, and then select 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 fails and you make changes to correct the failure.

To reset a test:

1. On the JD Edwards Autopilot Test Manager form, open a playlist that you have already run.
2. On the toolbar, click the Reset button.

   If Test Manager displays a form warning you that resetting the test overwrites the existing results, click Yes.
3. On the toolbar, click the Run button to rerun the test

Glossary of JD Edwards EnterpriseOne Terms

Accessor Methods/Assessors Java methods to "get" and "set" the elements of a value object or other source file.

activity rule The criteria by which an object progresses from one given point to the next in a flow.

add mode A condition of a form that enables users to input data.

Advanced Planning Agent (APAg) A JD Edwards EnterpriseOne tool that can be used to extract, transform, and load enterprise data. APAg supports access to data sources in the form of rational databases, flat file format, and other data or message encoding, such as XML.

alternate currency A currency that is different from the domestic currency (when dealing with a domestic-only transaction) or the domestic and foreign currency of a transaction.

In JD Edwards EnterpriseOne Financial Management, alternate currency processing enables you to enter receipts and payments in a currency other than the one in which they were issued.

Application Server Software that provides the business logic for an application program in a distributed environment. The servers can be Oracle Application Server (OAS) or WebSphere Application Server (WAS).

as if processing A process that enables you to view currency amounts as if they were entered in a currency different from the domestic and foreign currency of the transaction.

as of processing A process that is run as of a specific point in time to summarize transactions up to that date. For example, you can run various JD Edwards EnterpriseOne reports as of a specific date to determine balances and amounts of accounts, units, and so on as of that date.

Auto Commit Transaction A database connection through which all database operations are immediately written to the database.

back-to-back process A process in JD Edwards EnterpriseOne Supply Management that contains the same keys that are used in another process.

batch processing A process of transferring records from a third-party system to JD Edwards EnterpriseOne.

In JD Edwards EnterpriseOne Financial Management, batch processing enables you to transfer invoices and vouchers that are entered in a system other than JD Edwards EnterpriseOne to JD Edwards EnterpriseOne Accounts Receivable and JD Edwards EnterpriseOne Accounts Payable, respectively. In addition, you can transfer address book information, including customer and supplier records, to JD Edwards EnterpriseOne.

batch server A server that is designated for running batch processing requests. A batch server typically does not contain a database nor does it run interactive applications.

batch-of-one immediate A transaction method that enables a client application to perform work on a client workstation, then submit the work all at once to a server application for further processing. As a batch process is running on the server, the client application can continue performing other tasks.

See also direct connect and store-and-forward.

best practices Non-mandatory guidelines that help the developer make better design decisions.

BPEL Abbreviation for Business Process Execution Language, a standard web services orchestration language, which enables you to assemble discrete services into an end-to-end process flow.

BPEL PM Abbreviation for Business Process Execution Language Process Manager, a comprehensive infrastructure for creating, deploying, and managing BPEL business processes.

Build Configuration File Configurable settings in a text file that are used by a build program to generate ANT scripts. ANT is a software tool used for automating build processes. These scripts build published business services.

build engineer An actor that is responsible for building, mastering, and packaging artifacts. Some build engineers are responsible for building application artifacts, and some are responsible for building foundation artifacts.

Build Program A WIN32 executable that reads build configuration files and generates an ANT script for building published business services.

business analyst An actor that determines if and why an EnterpriseOne business service needs to be developed.

business function A named set of user-created, reusable business rules and logs that can be called through event rules. Business functions can run a transaction or a subset of a transaction (check inventory, issue work orders, and so on). Business functions also contain the application programming interfaces (APIs) that enable them to be called from a form, a database trigger, or a non-JD Edwards EnterpriseOne application. Business functions can be combined with other business functions, forms, event rules, and other components to make up an application. Business functions can be created through event rules or third-generation languages, such as C. Examples of business functions include Credit Check and Item Availability.

business function event rule See named event rule (NER).

business service EnterpriseOne business logic written in Java. A business service is a collection of one or more artifacts. Unless specified otherwise, a business service implies both a published business service and business service.

business service artifacts Source files, descriptors, and so on that are managed for business service development and are needed for the business service build process.

business service class method A method that accesses resources provided by the business service framework.

business service configuration files Configuration files include, but are not limited to, interop.ini, JDBj.ini, and jdelog.properties.

business service cross reference A key and value data pair used during orchestration. Collectively refers to both the code and the key cross reference in the WSG/XPI based system.

business service cross-reference utilities Utility services installed in a BPEL/ESB environment that are used to access JD Edwards EnterpriseOne orchestration cross-reference data.

business service development environment A framework needed by an integration developer to develop and manage business services.

business services development tool Otherwise known as JDeveloper.

business service EnterpriseOne object A collection of artifacts managed by EnterpriseOne LCM tools. Named and represented within EnterpriseOne LCM similarly to other EnterpriseOne objects like tables, views, forms, and so on.

business service framework Parts of the business service foundation that are specifically for supporting business service development.

business service payload An object that is passed between an enterprise server and a business services server. The business service payload contains the input to the business service when passed to the business services server. The business service payload contains the results from the business service when passed to the Enterprise Server. In the case of notifications, the return business service payload contains the acknowledgement.

business service property Key value data pairs used to control the behavior or functionality of business services.

Business Service Property Admin Tool An EnterpriseOne application for developers and administrators to manage business service property records.

business service property business service group A classification for business service property at the business service level. This is generally a business service name. A business service level contains one or more business service property groups. Each business service property group may contain zero or more business service property records.

business service property categorization A way to categorize business service properties. These properties are categorized by business service.

business service property key A unique name that identifies the business service property globally in the system.

business service property utilities A utility API used in business service development to access EnterpriseOne business service property data.

business service property value A value for a business service property.

business service repository A source management system, for example ClearCase, where business service artifacts and build files are stored. Or, a physical directory in network.

business services server The physical machine where the business services are located. Business services are run on an application server instance.

business services source file or business service class One type of business service artifact. A text file with the .java file type written to be compiled by a Java compiler.

business service value object template The structural representation of a business service value object used in a C-business function.

Business Service Value Object Template Utility A utility used to create a business service value object template from a business service value object.

business services server artifact The object to be deployed to the business services server.

business view A means for selecting specific columns from one or more JD Edwards EnterpriseOne application tables whose data is used in an application or report. A business view does not select specific rows, nor does it contain any actual data. It is strictly a view through which you can manipulate data.

central objects merge A process that blends a customer's modifications to the objects in a current release with objects in a new release.

central server A server that has been designated to contain the originally installed version of the software (central objects) for deployment to client computers. In a typical JD Edwards EnterpriseOne installation, the software is loaded on to one machine—the central server. Then, copies of the software are pushed out or downloaded to various workstations attached to it. That way, if the software is altered or corrupted through its use on workstations, an original set of objects (central objects) is always available on the central server.

charts Tables of information in JD Edwards EnterpriseOne that appear on forms in the software.

check-in repository A repository for developers to check in and check out business service artifacts. There are multiple check-in repositories. Each can be used for a different purpose (for example, development, production, testing, and so on).

connector Component-based interoperability model that enables third-party applications and JD Edwards EnterpriseOne to share logic and data. The JD Edwards EnterpriseOne connector architecture includes Java and COM connectors.

contra/clearing account A general ledger account in JD Edwards EnterpriseOne Financial Management that is used by the system to offset (balance) journal entries. For example, you can use a contra/clearing account to balance the entries created by allocations in JD Edwards EnterpriseOne Financial Management.

Control Table Workbench An application that, during the Installation Workbench processing, runs the batch applications for the planned merges that update the data dictionary, user-defined codes, menus, and user override tables.

control tables merge A process that blends a customer's modifications to the control tables with the data that accompanies a new release.

correlation data The data used to tie HTTP responses with requests that consist of business service name and method.

cost assignment The process in JD Edwards EnterpriseOne Advanced Cost Accounting of tracing or allocating resources to activities or cost objects.

cost component In JD Edwards EnterpriseOne Manufacturing, an element of an item's cost (for example, material, labor, or overhead).

credentials A valid set of JD Edwards EnterpriseOne username/password/environment/role, EnterpriseOne session, or EnterpriseOne token.

Cross-reference utility services Utility services installed in a BPEL/ESB environment that access EnterpriseOne cross-reference data.

cross segment edit A logic statement that establishes the relationship between configured item segments. Cross segment edits are used to prevent ordering of configurations that cannot be produced.

currency restatement The process of converting amounts from one currency into another currency, generally for reporting purposes. You can use the currency restatement process, for example, when many currencies must be restated into a single currency for consolidated reporting.

cXML A protocol used to facilitate communication between business documents and procurement applications, and between e-commerce hubs and suppliers.

database credentials A valid database username/password.

database server A server in a local area network that maintains a database and performs searches for client computers.

Data Source Workbench An application that, during the Installation Workbench process, copies all data sources that are defined in the installation plan from the Data Source Master and Table and Data Source Sizing tables in the Planner data source to the system-release number data source. It also updates the Data Source Plan detail record to reflect completion.

date pattern A calendar that represents the beginning date for the fiscal year and the ending date for each period in that year in standard and 52-period accounting.

denominated-in currency The company currency in which financial reports are based.

deployment artifacts Artifacts that are needed for the deployment process, such as servers, ports, and such.

deployment server A server that is used to install, maintain, and distribute software to one or more enterprise servers and client workstations.

detail information Information that relates to individual lines in JD Edwards EnterpriseOne transactions (for example, voucher pay items and sales order detail lines).

direct connect A transaction method in which a client application communicates interactively and directly with a server application.

See also batch-of-one immediate and store-and-forward.

Do Not Translate (DNT) A type of data source that must exist on the iSeries because of BLOB restrictions.

dual pricing The process of providing prices for goods and services in two currencies.

duplicate published business services authorization records Two published business services authorization records with the same user identification information and published business services identification information.

embedded application server instance An OC4J instance started by and running wholly within JDeveloper.

edit code A code that indicates how a specific value for a report or a form should appear or be formatted. The default edit codes that pertain to reporting require particular attention because they account for a substantial amount of information.

edit mode A condition of a form that enables users to change data.

edit rule A method used for formatting and validating user entries against a predefined rule or set of rules.

Electronic Data Interchange (EDI) An interoperability model that enables paperless computer-to-computer exchange of business transactions between JD Edwards EnterpriseOne and third-party systems. Companies that use EDI must have translator software to convert data from the EDI standard format to the formats of their computer systems.

embedded event rule An event rule that is specific to a particular table or application. Examples include form-to-form calls, hiding a field based on a processing option value, and calling a business function. Contrast with the business function event rule.

Employee Work Center A central location for sending and receiving all JD Edwards EnterpriseOne messages (system and user generated), regardless of the originating application or user. Each user has a mailbox that contains workflow and other messages, including Active Messages.

enterprise server A server that contains the database and the logic for JD Edwards EnterpriseOne.

Enterprise Service Bus (ESB) Middleware infrastructure products or technologies based on web services standards that enable a service-oriented architecture using an event-driven and XML-based messaging framework (the bus).

EnterpriseOne administrator An actor responsible for the EnterpriseOne administration system.

EnterpriseOne credentials A user ID, password, environment, and role used to validate a user of EnterpriseOne.

EnterpriseOne object A reusable piece of code that is used to build applications. Object types include tables, forms, business functions, data dictionary items, batch processes, business views, event rules, versions, data structures, and media objects.

EnterpriseOne development client Historically called "fat client," a collection of installed EnterpriseOne components required to develop EnterpriseOne artifacts, including the Microsoft Windows client and design tools.

EnterpriseOne extension A JDeveloper component (plug-in) specific to EnterpriseOne. A JDeveloper wizard is a specific example of an extension.

EnterpriseOne process A software process that enables JD Edwards EnterpriseOne clients and servers to handle processing requests and run transactions. A client runs one process, and servers can have multiple instances of a process. JD Edwards EnterpriseOne processes can also be dedicated to specific tasks (for example, workflow messages and data replication) to ensure that critical processes don't have to wait if the server is particularly busy.

EnterpriseOne resource Any EnterpriseOne table, metadata, business function, dictionary information, or other information restricted to authorized users.

Environment Workbench An application that, during the Installation Workbench process, copies the environment information and Object Configuration Manager tables for each environment from the Planner data source to the system-release number data source. It also updates the Environment Plan detail record to reflect completion.

escalation monitor A batch process that monitors pending requests or activities and restarts or forwards them to the next step or user after they have been inactive for a specified amount of time.

event rule A logic statement that instructs the system to perform one or more operations based on an activity that can occur in a specific application, such as entering a form or exiting a field.

explicit transaction Transaction used by a business service developer to explicitly control the type (auto or manual) and the scope of transaction boundaries within a business service.

exposed method or value object Published business service source files or parts of published business service source files that are part of the published interface. These are part of the contract with the customer.

facility An entity within a business for which you want to track costs. For example, a facility might be a warehouse location, job, project, work center, or branch/plant. A facility is sometimes referred to as a "business unit."

fast path A command prompt that enables the user to move quickly among menus and applications by using specific commands.

file server A server that stores files to be accessed by other computers on the network. Unlike a disk server, which appears to the user as a remote disk drive, a file server is a sophisticated device that not only stores files, but also manages them and maintains order as network users request files and make changes to these files.

final mode The report processing mode of a processing mode of a program that updates or creates data records.

foundation A framework that must be accessible for execution of business services at runtime. This includes, but is not limited to, the Java Connector and JDBj.

FTP server A server that responds to requests for files via file transfer protocol.

header information Information at the beginning of a table or form. Header information is used to identify or provide control information for the group of records that follows.

HTTP Adapter A generic set of services that are used to do the basic HTTP operations, such as GET, POST, PUT, DELETE, TRACE, HEAD, and OPTIONS with the provided URL.

instantiate A Java term meaning "to create." When a class is instantiated, a new instance is created.

integration developer The user of the system who develops, runs, and debugs the EnterpriseOne business services. The integration developer uses the EnterpriseOne business services to develop these components.

integration point (IP) The business logic in previous implementations of EnterpriseOne that exposes a document level interface. This type of logic used to be called XBPs. In EnterpriseOne 8.11, IPs are implemented in Web Services Gateway powered by webMethods.

integration server A server that facilitates interaction between diverse operating systems and applications across internal and external networked computer systems.

integrity test A process used to supplement a company's internal balancing procedures by locating and reporting balancing problems and data inconsistencies.

interface table See Z table.

internal method or value object Business service source files or parts of business service source files that are not part of the published interface. These could be private or protected methods. These could be value objects not used in published methods.

interoperability model A method for third-party systems to connect to or access JD Edwards EnterpriseOne.

in-your-face-error In JD Edwards EnterpriseOne, a form-level property which, when enabled, causes the text of application errors to appear on the form.

IServer service This internet server service resides on the web server and is used to speed up delivery of the Java class files from the database to the client.

jargon An alternative data dictionary item description that JD Edwards EnterpriseOne appears based on the product code of the current object.

Java application server A component-based server that resides in the middle-tier of a server-centric architecture. This server provides middleware services for security and state maintenance, along with data access and persistence.

JDBNET A database driver that enables heterogeneous servers to access each other's data.

JDEBASE Database Middleware A JD Edwards EnterpriseOne proprietary database middleware package that provides platform-independent APIs, along with client-to-server access.

JDECallObject An API used by business functions to invoke other business functions.

jde.ini A JD Edwards EnterpriseOne file (or member for iSeries) that provides the runtime settings required for JD Edwards EnterpriseOne initialization. Specific versions of the file or member must reside on every machine running JD Edwards EnterpriseOne. This includes workstations and servers.

JDEIPC Communications programming tools used by server code to regulate access to the same data in multiprocess environments, communicate and coordinate between processes, and create new processes.

jde.log The main diagnostic log file of JD Edwards EnterpriseOne. This file is always located in the root directory on the primary drive and contains status and error messages from the startup and operation of JD Edwards EnterpriseOne.

JDENET A JD Edwards EnterpriseOne proprietary communications middleware package. This package is a peer-to-peer, message-based, socket-based, multiprocess communications middleware solution. It handles client-to-server and server-to-server communications for all JD Edwards EnterpriseOne supported platforms.

JDeveloper Project An artifact that JDeveloper uses to categorize and compile source files.

JDeveloper Workspace	An artifact that JDeveloper uses to organize project files. It contains one or more project files.
JMS Queue	A Java Messaging service queue used for point-to-point messaging.
listener service	A listener that listens for XML messages over HTTP.
local repository	A developer's local development environment that is used to store business service artifacts.
local standalone BPEL/ESB server	A standalone BPEL/ESB server that is not installed within an application server.
Location Workbench	An application that, during the Installation Workbench process, copies all locations that are defined in the installation plan from the Location Master table in the Planner data source to the system data source.
logic server	A server in a distributed network that provides the business logic for an application program. In a typical configuration, pristine objects are replicated on to the logic server from the central server. The logic server, in conjunction with workstations, actually performs the processing required when JD Edwards EnterpriseOne software runs.
MailMerge Workbench	An application that merges Microsoft Word 6.0 (or higher) word-processing documents with JD Edwards EnterpriseOne records to automatically print business documents. You can use MailMerge Workbench to print documents, such as form letters about verification of employment.
Manual Commit transaction	A database connection where all database operations delay writing to the database until a call to commit is made.
master business function (MBF)	An interactive master file that serves as a central location for adding, changing, and updating information in a database. Master business functions pass information between data entry forms and the appropriate tables. These master functions provide a common set of functions that contain all of the necessary default and editing rules for related programs. MBFs contain logic that ensures the integrity of adding, updating, and deleting information from databases.
master table	See published table.
matching document	A document associated with an original document to complete or change a transaction. For example, in JD Edwards EnterpriseOne Financial Management, a receipt is the matching document of an invoice, and a payment is the matching document of a voucher.
media storage object	Files that use one of the following naming conventions that are not organized into table format: Gxxx, xxxGT, or GTxxx.
message center	A central location for sending and receiving all JD Edwards EnterpriseOne messages (system and user generated), regardless of the originating application or user.
messaging adapter	An interoperability model that enables third-party systems to connect to JD Edwards EnterpriseOne to exchange information through the use of messaging queues.
messaging server	A server that handles messages that are sent for use by other programs using a messaging API. Messaging servers typically employ a middleware program to perform their functions.
Middle-Tier BPEL/ESB Server	A BPEL/ESB server that is installed within an application server.
Monitoring Application	An EnterpriseOne tool provided for an administrator to get statistical information for various EntepriseOne servers, reset statistics, and set notifications.

named event rule (NER) Encapsulated, reusable business logic created using event rules, rather that C programming. NERs are also called business function event rules. NERs can be reused in multiple places by multiple programs. This modularity lends itself to streamlining, reusability of code, and less work.

nota fiscal In Brazil, a legal document that must accompany all commercial transactions for tax purposes and that must contain information required by tax regulations.

nota fiscal factura In Brazil, a nota fiscal with invoice information.

See also *nota fiscal.*

Object Configuration Manager (OCM) In JD Edwards EnterpriseOne, the object request broker and control center for the runtime environment. OCM keeps track of the runtime locations for business functions, data, and batch applications. When one of these objects is called, OCM directs access to it using defaults and overrides for a given environment and user.

Object Librarian A repository of all versions, applications, and business functions reusable in building applications. Object Librarian provides check-out and check-in capabilities for developers, and it controls the creation, modification, and use of JD Edwards EnterpriseOne objects. Object Librarian supports multiple environments (such as production and development) and enables objects to be easily moved from one environment to another.

Object Librarian merge A process that blends any modifications to the Object Librarian in a previous release into the Object Librarian in a new release.

Open Data Access (ODA) An interoperability model that enables you to use SQL statements to extract JD Edwards EnterpriseOne data for summarization and report generation.

Output Stream Access (OSA) An interoperability model that enables you to set up an interface for JD Edwards EnterpriseOne to pass data to another software package, such as Microsoft Excel, for processing.

package JD Edwards EnterpriseOne objects are installed to workstations in packages from the deployment server. A package can be compared to a bill of material or kit that indicates the necessary objects for that workstation and where on the deployment server the installation program can find them. It is point-in-time snapshot of the central objects on the deployment server.

package build A software application that facilitates the deployment of software changes and new applications to existing users. Additionally, in JD Edwards EnterpriseOne, a package build can be a compiled version of the software. When you upgrade your version of the ERP software, for example, you are said to take a package build.

Consider the following context: “Also, do not transfer business functions into the production path code until you are ready to deploy, because a global build of business functions done during a package build will automatically include the new functions.” The process of creating a package build is often referred to, as it is in this example, simply as “a package build.”

package location The directory structure location for the package and its set of replicated objects. This is usually \\deployment server\release\path_code\package\package name. The subdirectories under this path are where the replicated objects for the package are placed. This is also referred to as where the package is built or stored.

Package Workbench An application that, during the Installation Workbench process, transfers the package information tables from the Planner data source to the system-release number data source. It also updates the Package Plan detail record to reflect completion.

Pathcode Directory The specific portion of the file system on the EnterpriseOne development client where EnterpriseOne development artifacts are stored.

patterns General repeatable solutions to a commonly occurring problem in software design. For business service development, the focus is on the object relationships and interactions. For orchestrations, the focus is on the integration patterns (for example, synchronous and asynchronous request/response, publish, notify, and receive/reply).

planning family A means of grouping end items whose similarity of design and manufacture facilitates being planned in aggregate.

preference profile The ability to define default values for specified fields for a user-defined hierarchy of items, item groups, customers, and customer groups.

print server The interface between a printer and a network that enables network clients to connect to the printer and send their print jobs to it. A print server can be a computer, separate hardware device, or even hardware that resides inside of the printer itself.

pristine environment A JD Edwards EnterpriseOne environment used to test unaltered objects with JD Edwards EnterpriseOne demonstration data or for training classes. You must have this environment so that you can compare pristine objects that you modify.

processing option A data structure that enables users to supply parameters that regulate the running of a batch program or report. For example, you can use processing options to specify default values for certain fields, to determine how information appears or is printed, to specify date ranges, to supply runtime values that regulate program execution, and so on.

production environment A JD Edwards EnterpriseOne environment in which users operate EnterpriseOne software.

production-grade file server A file server that has been quality assurance tested and commercialized and that is usually provided in conjunction with user support services.

Production Published Business Services Web Service Published business services web service deployed to a production application server.

program temporary fix (PTF) A representation of changes to JD Edwards EnterpriseOne software that your organization receives on magnetic tapes or disks.

project In JD Edwards EnterpriseOne, a virtual container for objects being developed in Object Management Workbench.

promotion path The designated path for advancing objects or projects in a workflow. The following is the normal promotion cycle (path):

11>21>26>28>38>01

In this path, *11* equals new project pending review, *21* equals programming, *26* equals QA test/review, *28* equals QA test/review complete, *38* equals in production, *01* equals complete. During the normal project promotion cycle, developers check objects out of and into the development path code and then promote them to the prototype path code. The objects are then moved to the productions path code before declaring them complete.

proxy server A server that acts as a barrier between a workstation and the internet so that the enterprise can ensure security, administrative control, and caching service.

published business service EnterpriseOne service level logic and interface. A classification of a published business service indicating the intention to be exposed to external (non-EnterpriseOne) systems.

published business service identification information Information about a published business service used to determine relevant authorization records. Published business services + method name, published business services, or *ALL.

published business service web service — Published business services components packaged as J2EE Web Service (namely, a J2EE EAR file that contains business service classes, business service foundation, configuration files, and web service artifacts).

published table — Also called a master table, this is the central copy to be replicated to other machines. Residing on the publisher machine, the F98DRPUB table identifies all of the published tables and their associated publishers in the enterprise.

publisher — The server that is responsible for the published table. The F98DRPUB table identifies all of the published tables and their associated publishers in the enterprise.

pull replication — One of the JD Edwards EnterpriseOne methods for replicating data to individual workstations. Such machines are set up as pull subscribers using JD Edwards EnterpriseOne data replication tools. The only time that pull subscribers are notified of changes, updates, and deletions is when they request such information. The request is in the form of a message that is sent, usually at startup, from the pull subscriber to the server machine that stores the F98DRPCN table.

QBE — An abbreviation for query by example. In JD Edwards EnterpriseOne, the QBE line is the top line on a detail area that is used for filtering data.

real-time event — A message triggered from EnterpriseOne application logic that is intended for external systems to consume.

refresh — A function used to modify JD Edwards EnterpriseOne software, or subset of it, such as a table or business data, so that it functions at a new release or cumulative update level, such as B73.2 or B73.2.1.

replication server — A server that is responsible for replicating central objects to client machines.

Rt-Addressing — Unique data identifying a browser session that initiates the business services call request host/port user session.

rules — Mandatory guidelines that are not enforced by tooling, but must be followed in order to accomplish the desired results and to meet specified standards.

quote order — In JD Edwards Procurement and Subcontract Management, a request from a supplier for item and price information from which you can create a purchase order.

In JD Edwards Sales Order Management, item and price information for a customer who has not yet committed to a sales order.

secure by default — A security model that assumes that a user does not have permission to execute an object unless there is a specific record indicating such permissions.

Secure Socket Layer (SSL) — A security protocol that provides communication privacy. SSL enables client and server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery.

SEI implementation — A Java class that implements the methods that declare in a Service Endpoint Interface (SEI).

selection — Found on JD Edwards EnterpriseOne menus, a selection represents functions that you can access from a menu. To make a selection, type the associated number in the Selection field and press Enter.

serialize — The process of converting an object or data into a format for storage or transmission across a network connection link with the ability to reconstruct the original data or objects when needed.

Server Workbench — An application that, during the Installation Workbench process, copies the server configuration files from the Planner data source to the system-release number

	data source. The application also updates the Server Plan detail record to reflect completion.
Service Endpoint Interface (SEI)	A Java interface that declares the methods that a client can invoke on the service.
SOA	Abbreviation for Service Oriented Architecture.
soft coding	A coding technique that enables an administrator to manipulate site-specific variables that affect the execution of a given process.
source repository	A repository for HTTP adapter and listener service development environment artifacts.
spot rate	An exchange rate entered at the transaction level. This rate overrides the exchange rate that is set up between two currencies.
Specification merge	A merge that comprises three merges: Object Librarian merge, Versions List merge, and Central Objects merge. The merges blend customer modifications with data that accompanies a new release.
specification	A complete description of a JD Edwards EnterpriseOne object. Each object has its own specification, or name, which is used to build applications.
Specification Table Merge Workbench	An application that, during the Installation Workbench process, runs the batch applications that update the specification tables.
SSL Certificate	A special message signed by a certificate authority that contains the name of a user and that user's public key in such a way that anyone can "verify" that the message was signed by no one other than the certification authority and thereby develop trust in the user's public key.
store-and-forward	The mode of processing that enables users who are disconnected from a server to enter transactions and then later connect to the server to upload those transactions.
subscriber table	Table F98DRSUB, which is stored on the publisher server with the F98DRPUB table and identifies all of the subscriber machines for each published table.
superclass	An inheritance concept of the Java language where a class is an instance of something, but is also more specific. "Tree" might be the superclass of "Oak" and "Elm," for example.
supplemental data	Any type of information that is not maintained in a master file. Supplemental data is usually additional information about employees, applicants, requisitions, and jobs (such as an employee's job skills, degrees, or foreign languages spoken). You can track virtually any type of information that your organization needs. For example, in addition to the data in the standard master tables (the Address Book Master, Customer Master, and Supplier Master tables), you can maintain other kinds of data in separate, generic databases. These generic databases enable a standard approach to entering and maintaining supplemental data across JD Edwards EnterpriseOne systems.
table access management (TAM)	The JD Edwards EnterpriseOne component that handles the storage and retrieval of use-defined data. TAM stores information, such as data dictionary definitions; application and report specifications; event rules; table definitions; business function input parameters and library information; and data structure definitions for running applications, reports, and business functions.
Table Conversion Workbench	An interoperability model that enables the exchange of information between JD Edwards EnterpriseOne and third-party systems using non-JD Edwards EnterpriseOne tables.

table conversion An interoperability model that enables the exchange of information between JD Edwards EnterpriseOne and third-party systems using non-JD Edwards EnterpriseOne tables.

table event rules Logic that is attached to database triggers that runs whenever the action specified by the trigger occurs against the table. Although JD Edwards EnterpriseOne enables event rules to be attached to application events, this functionality is application specific. Table event rules provide embedded logic at the table level.

terminal server A server that enables terminals, microcomputers, and other devices to connect to a network or host computer or to devices attached to that particular computer.

three-tier processing The task of entering, reviewing and approving, and posting batches of transactions in JD Edwards EnterpriseOne.

three-way voucher match In JD Edwards Procurement and Subcontract Management, the process of comparing receipt information to supplier's invoices to create vouchers. In a three-way match, you use the receipt records to create vouchers.

transaction processing (TP) monitor A monitor that controls data transfer between local and remote terminals and the applications that originated them. TP monitors also protect data integrity in the distributed environment and may include programs that validate data and format terminal screens.

transaction processing method A method related to the management of a manual commit transaction boundary (for example, start, commit, rollback, and cancel).

transaction set An electronic business transaction (electronic data interchange standard document) made up of segments.

trigger One of several events specific to data dictionary items. You can attach logic to a data dictionary item that the system processes automatically when the event occurs.

triggering event A specific workflow event that requires special action or has defined consequences or resulting actions.

two-way authentication An authentication mechanism in which both client and server authenticate themselves by providing the SSL certificates to each other.

two-way voucher match In JD Edwards Procurement and Subcontract Management, the process of comparing purchase order detail lines to the suppliers' invoices to create vouchers. You do not record receipt information.

user identification information User ID, role, or *public.

User Overrides merge Adds new user override records into a customer's user override table.

value object A specific type of source file that holds input or output data, much like a data structure passes data. Value objects can be exposed (used in a published business service) or internal, and input or output. They are comprised of simple and complex elements and accessories to those elements.

variance In JD Edwards Capital Asset Management, the difference between revenue generated by a piece of equipment and costs incurred by the equipment.

In JD Edwards EnterpriseOne Project Costing and JD Edwards EnterpriseOne Manufacturing, the difference between two methods of costing the same item (for example, the difference between the frozen standard cost and the current cost is an engineering variance). Frozen standard costs come from the Cost Components table, and the current costs are calculated using the current bill of material, routing, and overhead rates.

versioning a published business service Adding additional functionality/interfaces to the published business services without modifying the existing functionality/interfaces.

Version List merge The Versions List merge preserves any non-XJDE and non-ZJDE version specifications for objects that are valid in the new release, as well as their processing options data.

visual assist Forms that can be invoked from a control via a trigger to assist the user in determining what data belongs in the control.

vocabulary override An alternate description for a data dictionary item that appears on a specific JD Edwards EnterpriseOne form or report.

wchar_t An internal type of a wide character. It is used for writing portable programs for international markets.

web application server A web server that enables web applications to exchange data with the back-end systems and databases used in eBusiness transactions.

web server A server that sends information as requested by a browser, using the TCP/IP set of protocols. A web server can do more than just coordination of requests from browsers; it can do anything a normal server can do, such as house applications or data. Any computer can be turned into a web server by installing server software and connecting the machine to the internet.

Web Service Description Language (WSDL) An XML format for describing network services.

Web Service Inspection Language (WSIL) An XML format for assisting in the inspection of a site for available services and a set of rules for how inspection-related information should be made.

web service proxy foundation Foundation classes for web service proxy that must be included in a business service server artifact for web service consumption on WAS.

web service softcoding record An XML document that contains values that are used to configure a web service proxy. This document identifies the endpoint and conditionally includes security information.

web service softcoding template An XML document that provides the structure for a soft coded record.

Where clause The portion of a database operation that specifies which records the database operation will affect.

Windows terminal server A multiuser server that enables terminals and minimally configured computers to display Windows applications even if they are not capable of running Windows software themselves. All client processing is performed centrally at the Windows terminal server and only display, keystroke, and mouse commands are transmitted over the network to the client terminal device.

wizard A type of JDeveloper extension used to walk the user through a series of steps.

workbench A program that enables users to access a group of related programs from a single entry point. Typically, the programs that you access from a workbench are used to complete a large business process. For example, you use the JD Edwards EnterpriseOne Payroll Cycle Workbench (P07210) to access all of the programs that the system uses to process payroll, print payments, create payroll reports, create journal entries, and update payroll history. Examples of JD Edwards EnterpriseOne workbenches include Service Management Workbench (P90CD020), Line Scheduling Workbench (P3153), Planning Workbench (P13700), Auditor's Workbench (P09E115), and Payroll Cycle Workbench.

work day calendar In JD Edwards EnterpriseOne Manufacturing, a calendar that is used in planning functions that consecutively lists only working days so that component and work order scheduling can be done based on the actual number of work days available. A work

	day calendar is sometimes referred to as planning calendar, manufacturing calendar, or shop floor calendar.
workflow	The automation of a business process, in whole or in part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules.
workgroup server	A server that usually contains subsets of data replicated from a master network server. A workgroup server does not perform application or batch processing.
XAPI events	A service that uses system calls to capture JD Edwards EnterpriseOne transactions as they occur and then calls third-party software, end users, and other JD Edwards EnterpriseOne systems that have requested notification when the specified transactions occur to return a response.
XML CallObject	An interoperability capability that enables you to call business functions.
XML Dispatch	An interoperability capability that provides a single point of entry for all XML documents coming into JD Edwards EnterpriseOne for responses.
XML List	An interoperability capability that enables you to request and receive JD Edwards EnterpriseOne database information in chunks.
XML Service	An interoperability capability that enables you to request events from one JD Edwards EnterpriseOne system and receive a response from another JD Edwards EnterpriseOne system.
XML Transaction	An interoperability capability that enables you to use a predefined transaction type to send information to or request information from JD Edwards EnterpriseOne. XML transaction uses interface table functionality.
XML Transaction Service (XTS)	Transforms an XML document that is not in the JD Edwards EnterpriseOne format into an XML document that can be processed by JD Edwards EnterpriseOne. XTS then transforms the response back to the request originator XML format.
Z event	A service that uses interface table functionality to capture JD Edwards EnterpriseOne transactions and provide notification to third-party software, end users, and other JD Edwards EnterpriseOne systems that have requested to be notified when certain transactions occur.
Z table	A working table where non-JD Edwards EnterpriseOne information can be stored and then processed into JD Edwards EnterpriseOne. Z tables also can be used to retrieve JD Edwards EnterpriseOne data. Z tables are also known as interface tables.
Z transaction	Third-party data that is properly formatted in interface tables for updating to the JD Edwards EnterpriseOne database.

Index

A

B

C

D

E

F

G

O

P

Q

R

S

T

U