

PeopleSoft®

PeopleSoft EnterpriseOne Tools
8.95 PeopleBook: PeopleSoft
Virtual AutoPilot

August 2005

PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: PeopleSoft Virtual AutoPilot
SKU E1_TOOLS895TVA-B 0805
Copyright © 2005, Oracle. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are “commercial computer software” or “commercial technical data” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software–Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee’s responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Open Source Disclosure

Oracle takes no responsibility for its use or distribution of any open source or shareware software or documentation and disclaims any and all liability or damages resulting from use of said software or documentation. The following open source software may be used in Oracle’s PeopleSoft products and the following disclaimers are provided.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>). Copyright © 1999-2000 The Apache Software Foundation. All rights reserved. THIS SOFTWARE IS PROVIDED “AS IS” AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contents

General Preface

- About This PeopleBook Prefacevii**
- PeopleSoft Application Prerequisites.....vii
- PeopleSoft Application Fundamentals.....vii
- Documentation Updates and Printed Documentation.....viii
 - Obtaining Documentation Updates.....viii
 - Ordering Printed Documentation.....viii
- Additional Resources.....ix
- Typographical Conventions and Visual Cues.....x
 - Typographical Conventions.....x
 - Visual Cues.....xi
 - Country, Region, and Industry Identifiers.....xii
 - Currency Codes.....xii
- Comments and Suggestions.....xii
- Common Elements Used in PeopleBooks.....xiii

Preface

- PeopleSoft EnterpriseOne Tools Virtual Autopilot Preface.....xv**
- Virtual AutoPilot Companion Documentation.....xv

Chapter 1

- Getting Started with PeopleSoft EnterpriseOne Tools Virtual Autopilot.....1**
- Virtual Autopilot Overview.....1
- Virtual Autopilot Implementation.....1
 - Virtual Autopilot Implementation Steps.....1

Chapter 2

- Understanding Data Capture for PeopleSoft Virtual AutoPilot Scripts.....3**
- Overall Testing Tools Architecture.....3
 - Data Capture Components.....3
- Understanding PeopleSoft AutoPilot.....4
 - PeopleSoft AutoPilot Tasks.....4
 - PeopleSoft Software and PeopleSoft AutoPilot Code.....5

Event Stream.....5
 PeopleSoft AutoPilot Playback Results Detail Table (F97214).....6

Chapter 3

Understanding PeopleSoft Virtual AutoPilot Components.....7

PeopleSoft AutoPilot Components.....7
 Virtual Script Editor.....8
 Virtual Script Editor Features.....8
 Event Pane.....10
 Event Graph.....11
 Parameter Detail Pane.....11
 Script List Pane.....12
 Parameter Value Linking.....13
 Source and Target Parameter Identification.....13
 Looping Identification.....15
 HRequest Handle Value Linking.....16
 Thread Identification.....17
 Timing Interval Maintenance.....17
 PeopleSoft Virtual AutoPilot Script Generation.....18
 Virtual Script Player.....18
 Virtual Script Player Features.....19
 Virtual Script Player Initialization File Parameters.....19
 Virtual Script Player Command Line.....22
 Environment Initialization.....22
 Modes of Operation.....23
 Preprocessing of Valid Values List Data.....23
 Valid Values List Processing.....23
 Date Formatting.....23
 Script Failure.....24
 Virtual Script Player Limitations.....24
 VSMEditor.....24
 VSM Editor Features.....24
 All Virtual Scripts List Box.....25
 Master Scripts List Box.....25
 VSM Files.....25
 Virtual Runner.....25
 Virtual Runner Features.....25
 Player Session Columns.....26
 Actions Tools.....26

Chapter 4

Creating Virtual Scripts.....29
 Understanding Script Creation.....29
 Capturing and Importing Test Results.....29
 Understanding Test Results.....30
 Capturing Test Results.....30
 Importing Test Results.....31
 Viewing Test Results.....31
 Editing Virtual Scripts.....32
 Understanding Virtual Script Generation.....32
 Using the Find Feature.....32
 Adding Loops.....33
 Value-Linking Parameters.....33
 Linking Values in Inquiry Scripts.....34
 Linking Values in Entry Scripts.....35
 Generating PeopleSoft Virtual AutoPilot Scripts.....35
 Creating Master Scripts.....36

Chapter 5

Running Virtual Scripts.....37
 Understanding Virtual Script Runs.....37
 Running Virtual Scripts from a Single Workstation.....37
 Understanding Virtual Script Runs from a Single Workstation.....37
 Prerequisite.....37
 Running Virtual Scripts from the Command Line.....38
 Running Virtual Scripts Using Virtual Runner.....38
 Launching and Managing Multiple Script Playback.....40
 Understanding LoadRunner.....40
 Defining Scripts.....40
 Defining the Host Machine.....40
 Defining Virtual Users.....41
 Gathering LoadRunner Results.....41
 Running Virtual Playback from the LoadRunner Controller.....41

Chapter 6

Understanding Special Considerations for Simulated Playback.....43
 Playback Problem Simulation.....43
 Playback Timing.....43

Event Synchronization.....43
 API Playback Timing.....44
 Interthread Timing.....44
 Call Level.....45
 Synchronous and Asynchronous Calls.....45
 Think Times.....46

Chapter 7

Troubleshooting PeopleSoft Virtual AutoPilot.....49
 Understanding PeopleSoft Virtual AutoPilot Troubleshooting.....49
 Reviewing the PeopleSoft Virtual AutoPilot Log File to Locate Errors.....49
 Understanding the PeopleSoft Virtual AutoPilot Log File.....49
 Finding Error Entries in the PeopleSoft Virtual AutoPilot Log File.....50
 Locating the Log File in the Event of Early Script Failure.....50
 Setting the MessageLevel Parameter.....51
 Identifying Environment Problems.....51
 Understanding Environment Problems.....51
 Diagnosing Environment Problems.....51
 Investigating System Errors.....52
 Debugging PeopleSoft Virtual AutoPilot Scripts.....52
 Understanding PeopleSoft Virtual AutoPilot Debugging.....53
 Displaying Business Function Parameters.....53
 Diagnosing Business Function Failures in PeopleSoft Explorer.....54
 Researching Value-Linking Errors in the Virtual Script Editor.....54
 Verifying That Value Linking Is Functioning.....55
 Identifying and Correcting Variable Value-Linking Errors.....56
 Verifying the Validity of PeopleSoft Virtual AutoPilot Script Data.....56
 Identifying and Correcting Duplicate Key Errors.....57
 Rectifying Irregular Transaction Times.....57
 Preventing Multiple Script Playback Problems.....58
 Correcting Uninitialized User Handle Errors.....59

Glossary of PeopleSoft Terms.....61

Index71

About This PeopleBook Preface

PeopleBooks provide you with the information that you need to implement and use PeopleSoft applications.

This preface discusses:

- PeopleSoft application prerequisites.
- PeopleSoft application fundamentals.
- Documentation updates and printed documentation.
- Additional resources.
- Typographical conventions and visual cues.
- Comments and suggestions.
- Common elements in PeopleBooks.

Note. PeopleBooks document only page elements, such as fields and check boxes, that require additional explanation. If a page 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 elements for the section, chapter, PeopleBook, or product line. Elements that are common to all PeopleSoft applications are defined in this preface.

PeopleSoft Application Prerequisites

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

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

You should be familiar with navigating the system and adding, updating, and deleting information by using PeopleSoft menus, and pages, 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 PeopleSoft applications most effectively.

PeopleSoft Application Fundamentals

Each application PeopleBook provides implementation and processing information for your PeopleSoft applications.

Note. Application fundamentals PeopleBooks are not applicable to the PeopleTools product.

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 PeopleBook. Most PeopleSoft product lines have a version of the application fundamentals PeopleBook. The preface of each PeopleBook identifies the application fundamentals PeopleBooks that are associated with that PeopleBook.

The application fundamentals PeopleBook consists of important topics that apply to many or all PeopleSoft applications across one or more product lines. 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 PeopleBooks. They provide the starting points for fundamental implementation tasks.

Documentation Updates and Printed Documentation

This section discusses how to:

- Obtain documentation updates.
- Order printed documentation.

Obtaining Documentation Updates

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

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

See Also

PeopleSoft Customer Connection, <https://www.peoplesoft.com/corp/en/login.jsp>

Ordering Printed Documentation

You can order printed, bound volumes of the complete PeopleSoft documentation that is delivered on your PeopleBooks CD-ROM. PeopleSoft makes printed documentation available for each major release shortly after the software is shipped. Customers and partners can order printed PeopleSoft documentation by using any of these methods:

- Web
- Telephone
- Email

Web

From the Documentation section of the PeopleSoft Customer Connection website, access the PeopleBooks Press website under the Ordering PeopleBooks topic. The PeopleBooks Press website is a joint venture between PeopleSoft and MMA Partners, the book print vendor. Use a credit card, money order, cashier's check, or purchase order to place your order.

Telephone

Contact MMA Partners at 877 588 2525.

Email

Send email to MMA Partners at peoplebookspres@mmapartner.com.

See Also

PeopleSoft Customer Connection, <https://www.peoplesoft.com/corp/en/login.jsp>

Additional Resources

The following resources are located on the PeopleSoft Customer Connection website:

Resource	Navigation
Application maintenance information	Updates + Fixes
Business process diagrams	Support, Documentation, Business Process Maps
Interactive Services Repository	Interactive Services Repository
Hardware and software requirements	Implement, Optimize + Upgrade, Implementation Guide, Implementation Documentation & Software, Hardware and Software Requirements
Installation guides	Implement, Optimize + Upgrade, Implementation Guide, Implementation Documentation & Software, Installation Guides and Notes
Integration information	Implement, Optimize + Upgrade, Implementation Guide, Implementation Documentation and Software, Pre-built Integrations for PeopleSoft Enterprise and PeopleSoft EnterpriseOne Applications
Minimum technical requirements (MTRs) (EnterpriseOne only)	Implement, Optimize + Upgrade, Implementation Guide, Supported Platforms
PeopleBook documentation updates	Support, Documentation, Documentation Updates
PeopleSoft support policy	Support, Support Policy
Prerelease notes	Support, Documentation, Documentation Updates, Category, Prerelease Notes
Product release roadmap	Support, Roadmaps + Schedules
Release notes	Support, Documentation, Documentation Updates, Category, Release Notes

Resource	Navigation
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 PeopleBooks:

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.
<i>Italics</i>	Indicates field values, emphasis, and PeopleSoft 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 <i>O</i> .
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.

Typographical Convention or Visual Cue	Description
“ ” (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)	<p>When placed before a parameter in PeopleCode syntax, an ampersand indicates that the parameter is an already instantiated object.</p> <p>Ampersands also precede all PeopleCode variables.</p>

Visual Cues

PeopleBooks contain the following visual cues.

Notes

Notes indicate information that you should pay particular attention to as you work with the PeopleSoft 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

PeopleBooks 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 PeopleBooks:

- 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 PeopleBooks:

- 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 are important to us. We encourage you to tell us what you like, or what you would like to see changed about PeopleBooks and other PeopleSoft reference and training materials. Please send your suggestions to:

PeopleSoft Product Documentation Manager PeopleSoft, Inc. 4460 Hacienda Drive Pleasanton, CA 94588

Or send email comments to doc@peoplesoft.com.

While we cannot guarantee to answer every email message, we will pay careful attention to your comments and suggestions.

Common Elements Used in PeopleBooks

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 allows 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: <i>Blank:</i> Batch is unposted and pending approval. <i>A:</i> The batch is approved for posting, has no errors and is in balance, but it has not yet been posted. <i>D:</i> The batch posted successfully. <i>E:</i> The batch is in error. You must correct the batch before it can post. <i>P:</i> 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. <i>U:</i> 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. PeopleSoft 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	<p>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.</p> <p>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.</p> <p>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.</p>
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	<p>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. PeopleSoft EnterpriseOne reserves these prefixes for the document types indicated:</p> <p><i>P</i>: Accounts payable documents.</p> <p><i>R</i>: Accounts receivable documents.</p> <p><i>T</i>: Time and pay documents.</p> <p><i>I</i>: Inventory documents.</p> <p><i>O</i>: Purchase order documents.</p> <p><i>S</i>: Sales order documents.</p>
Effective Date	<p>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:</p> <ul style="list-style-type: none">• 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.

PeopleSoft EnterpriseOne Tools Virtual AutoPilot Preface

This preface discusses Virtual AutoPilot companion documentation.

Virtual AutoPilot Companion Documentation

Additional, essential information describing the setup and design of PeopleSoft EnterpriseOne Tools resides in companion documentation. The companion documentation consists of important topics that apply to Virtual AutoPilot as well as other PeopleSoft EnterpriseOne Tools. You should be familiar with the contents of these companion PeopleBooks:

- Autopilot
- Foundation

See Also

PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: PeopleSoft AutoPilot, “Getting Started with PeopleSoft EnterpriseOne Tools AutoPilot”

PeopleSoft EnterpriseOne Tools 8.95 PeopleBook: Foundation, “Getting Started with PeopleSoft EnterpriseOne Tools Foundation”

CHAPTER 1

Getting Started with PeopleSoft EnterpriseOne Tools Virtual Autopilot

This chapter discusses:

- Virtual Autopilot Overview
- Virtual Autopilot Implementation

Virtual Autopilot Overview

Virtual Autopilot is an automated testing tool that is used to capture data and provide users with the raw material to build a virtual script that will accurately simulate PeopleSoft software processes.

Virtual Autopilot Implementation

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

In the planning phase of your implementation, take advantage of all PeopleSoft sources of information, including the installation guides and troubleshooting information. A complete list of these resources appears in the preface in *About These PeopleBooks* with information about where to find the most current version of each.

Virtual Autopilot Implementation Steps

This table lists the steps for the Virtual Autopilot implementation.

Step	Reference
1. Install EnterpriseOne	EnterpriseOne installation documentation for your platform.
2. Install AutoPilot	EnterpriseOne installation documentation for your platform.
3. Install Virtual AutoPilot	EnterpriseOne installation documentation for your platform.

CHAPTER 2

Understanding Data Capture for PeopleSoft Virtual AutoPilot Scripts

This chapter provides overviews of automated testing tools architecture and data capture, and discusses:

- Overall testing tools architecture.
- Data capture components.

Overall Testing Tools Architecture

These components of PeopleSoft automated testing tools architecture work together to capture, record, and store data about PeopleSoft software processes, including the parameters of all API calls and all other PeopleSoft software runtime events:

- PeopleSoft AutoPilot, which enables you to write and play back a script to test PeopleSoft software applications and to configure script playback so that PeopleSoft AutoPilot captures and saves playback data
- Hooks, or code, that reside in PeopleSoft software and in PeopleSoft AutoPilot that capture and record data generated by the playback of a PeopleSoft AutoPilot script
- Event stream, which is a time-stamped, chronological record of each PeopleSoft AutoPilot and PeopleSoft software event that occurs during script playback
- PeopleSoft AutoPilot Playback Results Detail Table (F97214), which stores the event stream

The placement of PeopleSoft software code is important to the creation of PeopleSoft Virtual AutoPilot scripts. Because this code is positioned at the boundary between the PeopleSoft software runtime engine and the PeopleSoft software middleware, it captures data passing to the JDB and CallObject APIs before the APIs are routed to servers by the OCM. Therefore, you can reuse PeopleSoft Virtual AutoPilot scripts regardless of changes to OCM mappings.

Data Capture Components

Creating a virtual script requires that you first capture data from a PeopleSoft software session in which you launch an application, press buttons, enter data to header controls, and so on. The automated testing tool architecture of which PeopleSoft AutoPilot is a part enables you to capture the events of a PeopleSoft software session by writing a script, configuring it for event capture, playing it back, and storing its results. You accomplish these tasks using these three components:

Component	Function
PeopleSoft AutoPilot	Enables you to write a script, play it back in PeopleSoft software, and save the results of the playback.
Code that resides in PeopleSoft software and in PeopleSoft AutoPilot	Captures and records the data generated by PeopleSoft AutoPilot and by PeopleSoft software during script playback.
PeopleSoft AutoPilot Playback Results Detail Table (F97214)	Stores the data generated during script playback as an event stream, which is a continuous record of every PeopleSoft software and PeopleSoft AutoPilot event that occurred during script playback.

Understanding PeopleSoft AutoPilot

This section discusses:

- PeopleSoft AutoPilot tasks.
- PeopleSoft software and PeopleSoft AutoPilot code.
- Event stream.
- PeopleSoft AutoPilot Playback Results Detail Table (F97214).

PeopleSoft AutoPilot Tasks

The process of creating a virtual script begins with PeopleSoft AutoPilot, which you use to write a script that tests PeopleSoft software processes. Playing back a PeopleSoft AutoPilot script simulates PeopleSoft software activities, but only as initiated by one user. However, you can capture the results of script playback, including the processes generated, save the data, and use it to create a virtual script that you run to simulate more than one user.

PeopleSoft AutoPilot Script Creation

To begin the process of capturing data, you first write a PeopleSoft AutoPilot script to test software processes such as launching applications, pressing buttons, entering data to header controls, and so on.

PeopleSoft AutoPilot Playback Configuration

You capture data about software processes by playing back the PeopleSoft AutoPilot script, but you must configure playback for data capture. Your configuration choices establish how much data you capture and ensure that PeopleSoft AutoPilot saves the data.

You can capture data at one of two levels:

- Level 1 captures data only for initiating API calls that run alone or call other APIs. If you select this option, you capture data about only these APIs.
- All API calls capture data not only about level 1 API calls but about any API calls spawned by a level 1 API.

You also configure script playback to save and display results data after playback.

PeopleSoft AutoPilot Script Playback

After you have written a script and configured playback to capture the results, you play back the script. PeopleSoft AutoPilot captures the playback data using internal code and code placed in the software.

PeopleSoft Software and PeopleSoft AutoPilot Code

Code is strategically placed in PeopleSoft AutoPilot and in 32 JDB functions and 1 CallObject function in PeopleSoft software to gather and store during PeopleSoft AutoPilot script playback.

The placement of PeopleSoft software code provides these advantages for creating PeopleSoft Virtual PeopleSoft AutoPilot scripts:

- Comprehensive data capture. Because code is positioned to capture both JDB and JDE CallObject API calls, you capture both database and business function activity.
- Simplified script maintenance. Because the code resides in slightly more than 30 JDB and CallObject functions combined, making changes in PeopleSoft software code is relatively easy.
- Flexibility in running scripts. Data that you capture can be run independent of platform or Object Configuration Manager (OCM) mapping considerations.

Code placed in PeopleSoft software performs these functions that lay the groundwork for the creation of a PeopleSoft Virtual AutoPilot script:

- Captures parameter data on JDB and CallObject API calls that occur during the playback of a PeopleSoft AutoPilot script
- Writes the parameter data to a file-mapping object that PeopleSoft software shares with PeopleSoft AutoPilot
- Writes data on event rules, button presses, and event timing to the file-mapping object

Code placed in PeopleSoft AutoPilot performs these functions that lay the groundwork for the creation of a PeopleSoft Virtual AutoPilot script:

- Writes data on PeopleSoft AutoPilot events to the file-mapping object
- Copies the PeopleSoft software and PeopleSoft AutoPilot data in the shared file space into a BLOB (Binary Large Object) field in the PeopleSoft AutoPilot Playback Results Detail Table (F97214)
- Enables the PeopleSoft AutoPilot user to access the PeopleSoft AutoPilot Playback Results Detail Table (F97214)

PeopleSoft Virtual AutoPilot scripts that you create using the data captured from a PeopleSoft AutoPilot script are platform independent and can be run on any operating system with any OCM mappings because PeopleSoft software code captures API call data before it reaches the OCM for mapping.

Event Stream

You generate an event stream when you run a PeopleSoft AutoPilot script that you have configured to capture playback results. The event stream is a time-stamped, chronological record of every PeopleSoft AutoPilot and PeopleSoft software event that occurs during playback, including:

- JDB and CallObject API calls
- Thread identification
- Event rules
- PeopleSoft software error and warning messages

- PeopleSoft AutoPilot events confirming that the script and PeopleSoft software are on the same form

PeopleSoft AutoPilot Playback Results Detail Table (F97214)

The PeopleSoft AutoPilot Playback Results Detail Table (F97214) contains the results of all PeopleSoft AutoPilot playback sessions that you have captured and saved. You can access the event stream for a script playback session through PeopleSoft AutoPilot, the Virtual Script Editor, or PeopleSoft EnterpriseOne Analyzer Tool.

Using PeopleSoft AutoPilot, you can view a summary of every event that occurred during script playback. For example, you can view this information for any test on the test Results form:

- Results sets
- Summary (Script, Release, Machine, and so on)
- JDE.INI and DEDEBUG.LOG
- Screen captures
- Messages
- Results

From the Virtual Script Editor or from PeopleSoft EnterpriseOne Analyzer Tool, you can view each event in more detail. For example, you can select an API call and view the input and output parameter values for the call. If you import an event stream record from the database table to the Virtual Script Editor, you can modify the record so that you can play it as a PeopleSoft Virtual AutoPilot script.

CHAPTER 3

Understanding PeopleSoft Virtual AutoPilot Components

This chapter discusses:

- PeopleSoft AutoPilot components.
- Virtual Script Editor.
- Virtual Script Player.
- VSM Editor.
- Virtual Runner.

PeopleSoft AutoPilot Components

You use components that are external to PeopleSoft Virtual AutoPilot (Virtual AutoPilot) to capture, record, and store data generated by an PeopleSoft AutoPilot playback session. These components include:

- PeopleSoft AutoPilot
- Data-capture code in PeopleSoft AutoPilot and in PeopleSoft software
- Event stream
- PeopleSoft AutoPilot Playback Results Detail Table (F97214), where the event stream from the playback session is stored

The internal components of PeopleSoft Virtual AutoPilot enable you to complete the PeopleSoft Virtual AutoPilot scripting process. These components are:

- Virtual Script Editor
- Virtual Script Player
- VSMEditor
- Virtual Runner

You use the internal components of Virtual User Tool to complete these tasks:

- Import an event stream into the Virtual Script Editor
- Modify the event stream by adding rules that govern the passing of parameters and looping (repeated JDB Fetch calls to complete a database inquiry)
- Use the Virtual Script Editor to automatically add rules to handle thread identification and hRequest handles
- Generate a PeopleSoft Virtual AutoPilot script

- Run a PeopleSoft Virtual AutoPilot script on the Virtual Script Player
- Use the VSMEEditor to concatenate a series of individual PeopleSoft Virtual AutoPilot scripts into one master script
- Use Virtual Runner to manage script playback, either from a single workstation or from multiple workstations
You also can manage script playback using LoadRunner from Mercury Interactive.

Virtual Script Editor

This section discusses:

- Virtual Script Editor features.
- Event pane.
- Event graph.
- Parameter detail pane.
- Script list pane.
- Parameter value linking.
- Source and target parameter identification.
- Looping identification.
- HRequest handle value linking.
- Thread identification.
- Timing interval maintenance.
- PeopleSoft Virtual AutoPilot script generation.

Virtual Script Editor Features

The Virtual Script Editor enables you to create and generate a PeopleSoft Virtual AutoPilot (Virtual AutoPilot) script that you can use to simulate the activity of many concurrent users. Working with the Virtual Script Editor represents the second step in a three-step process of producing a PeopleSoft Virtual AutoPilot script playback session. You create a PeopleSoft Virtual AutoPilot script playback session by completing these steps in order:

1. Capture data generated by PeopleSoft AutoPilot script playback and store the event stream in a results repository.
2. Use the Virtual Script Editor to modify an event stream and to generate a PeopleSoft Virtual AutoPilot script that contains all the information required by the Virtual Script Player to simulate the activities of the system's runtime engine.
3. Play back a modified event stream (the PeopleSoft Virtual AutoPilot script) using the Virtual Script Player.

The event stream is a chronological, time-stamped record of every event that occurs during the playback of an PeopleSoft AutoPilot script, including:

- User input
- Processing performed by the run-time engine, such as thread creation
- Event rules; informative messages

- API calls to the PeopleSoft middleware

PeopleSoft AutoPilot performs no editing during the process. The event stream represents a record of the events that have already occurred. You cannot edit it by adding, deleting, or reordering data. To change it, you must generate a new one by modifying an existing PeopleSoft AutoPilot script, or by creating a new script and then replaying it.

Using the Virtual Script Editor, you can:

- View the titles of all the scripts whose results you stored in the PeopleSoft AutoPilot Playback Results Detail Table (F97214)
- Import an event stream
- View an event stream as a single, continuous record
- View the timing of events by category, represented in a horizontal bar graph
- Select an individual API call and view the input values sent to the server and the output values returned to the client workstation
- Create links between parameters of API calls so that parameter values can be passed between calls during virtual script playback
- Identify and designate loops so that the virtual script can handle repetitive processing tasks, such as database retrieval

The Virtual Script Editor helps you address problems that you encounter when trying to create a script that you can run to simulate activities in a dynamic client/server system. Problems that you might encounter include:

- Identifying API parameters that require dynamic values
- Providing a way to pass values dynamically between API call parameters to avoid data conflict and record contention
- Making the values of hRequest handle parameter values dynamic to simulate concurrent user activity
- Synchronizing timing between events during script playback to keep processing running regardless of network stress placed on the server
- Synchronizing timing between data-dependent APIs in threads running asynchronously to avoid one API starting before another has finished processing
- Identifying repetitive processing tasks, such as database inquiry, so that the Virtual Script Player can efficiently simulate the work of the software

The Virtual Script Editor handles these virtual script creation tasks automatically:

- Linking values of parameters in separate API calls so that values can be passed, provided that the calls meet certain criteria
- Storing the values of hRequest handles as variables
- Storing identification of thread IDs
- Storing information about time gaps between events in a single thread and between interthread-dependent events

You perform these virtual script creation tasks manually:

- Linking values of parameters in separate API calls that do not meet all the criteria required for automatic value linking
- Identifying repetitive processes, such as database inquiry, as loops

After you have completed these manual tasks, you use the Virtual Script Editor to generate the virtual script. The Virtual Script Player receives from the Virtual Script Editor all of the information necessary to run the virtual script.

Event Pane

You click the Import button on the Virtual Script Editor form to import the results from an PeopleSoft AutoPilot script playback session. The Virtual Script Editor populates the event pane with the event stream. The event stream contains a time stamping of each event. Therefore, you can review PeopleSoft AutoPilot events or API calls during playback that might have taken an unusual amount of time to run.

You use the event pane to view data about these kinds of playback events:

- CallObject APIs
- JDB APIs
- PeopleSoft AutoPilot events
- Event rules
- Informative messages, including system errors and warnings
- Thread creation

The event pane also contains this columnar information about each event:

- Timing information, such as the start, end, and elapsed time of an event
- Thread identification
- hUser handle identification
- hRequest handle identification
- Call level
- Message entry identifying the event
- Message information about an event, such as JDB call to open a table from memory cache

The message entry for each event includes an abbreviation that identifies the type of event that occurred. This table summarizes the abbreviations and the type of event that each represents:

Abbreviation in Message Column of Event Stream	Type of Script Event
JDB	Database API call
RTE	CallObject API call
EVR	Event rule
LOG	System warning message
ERR	System error message
MSG	PeopleSoft AutoPilot message

Abbreviation in Message Column of Event Stream	Type of Script Event
AUT	Action in PeopleSoft AutoPilot (for example, typing to control)
THR	Thread action

You use these buttons to change the view in the event pane:

Generate	Enables you to generate a PeopleSoft Virtual AutoPilot script. Click this button only after you have finished editing the event stream in the Virtual Script Editor.
Filter	Enables you to remove unwanted events from the list by applying criteria found in the Filter form.
View Log	Enables you to look at the log produced when you generate the PeopleSoft Virtual AutoPilot script. The log includes the number of lines in the script and the number of errors, if any.

Event Graph

You can view playback events by category in a horizontal bar graph by choosing the Graph option in the Virtual Script Editor. While the event stream pane presents the events of a PeopleSoft AutoPilot script playback vertically, in a single chronological stream, the event graph presents the events horizontally across a timeline.

You can break up the chronology by message type, such as JDB API calls or event rules.

The event graph provides you with another detailed snapshot of activity that occurred during PeopleSoft AutoPilot script playback. You can focus on events of unusual duration, which can be helpful in debugging applications, analyzing network activity, or rewriting and rerunning the original PeopleSoft AutoPilot script.

Parameter Detail Pane

You can view the parameters that make up an API call by clicking an API call event line in the event pane. The pane that appears shows the name of each parameter in the call and its value, if any. For example, the detail pane might display the value of the user handle parameter that a JDB call passes to the database.

This detail pane provides a complete snapshot of each API call at a given point in time. For example, the pane shows arrows that indicate the flow of data that occurred during the call. An arrow on the left side of the box next to the name of a parameter indicates that the call passed the value from the client workstation to the JDENET or database driver. An arrow on the right side of the box indicates that the call returned data from the server. In some cases, a box contains both arrows, indicating that data flowed in both directions.

The parameter detail pane offers a before-and-after snapshot of script playback. Before playback, parameters for a CallObject API, such as BatchOpenOnInitialization, contain no batch number or batch date parameter values. After playback, these parameters contain returned values.

The parameter detail pane also displays the parameters of API calls that pass an environment handle to the database.

Finally, many API calls contain a request handle that points to a particular place in memory that the run-time engine has allocated for the call. The parameter for the request handle appears in the parameter detail pane if the API call used a request handle.

The ability of the PeopleSoft AutoPilot and PeopleSoft software hooks to capture data at this level of detail is critically important to PeopleSoft Virtual AutoPilot because the goal of PeopleSoft Virtual AutoPilot is to simulate, as closely as possible, the actual activities of the system. If the PeopleSoft Virtual AutoPilot script does not have the complete parameters of an API call, it cannot accurately model the activities of the system and its interaction with the client workstation, the database server, and the application server.

Script List Pane

The script list pane on the Virtual Script Editor form displays in chronological order the PeopleSoft AutoPilot script playback results that you saved.

The script list pane displays script result information in these columns:

Column	Description
Test	Database ID number assigned to each PeopleSoft AutoPilot script playback session
Client	ID of the workstation on which you ran the test
Start Time	Date and time at which you ran the test
Elapsed	Time it took the test to run to a successful conclusion, failure, or cancellation
Environment	System environment against which you ran the test
Release	PeopleSoft software release against which you ran the test
Script	Name that you assigned to the test
Status	Result of the test-success, failure, or cancellation

After you select a script, you select one of these buttons to manipulate the form view:

Filter	Enables you to remove PeopleSoft AutoPilot script playback results that you do not want, using criteria on the Filter form.
Import	Imports into the Virtual Script Editor the event stream from a test result that you select.
Refresh	Refreshes the script list pane from the database.
Delete	Removes one or more tests from the database.

See Also

[Chapter 6, “Understanding Special Considerations for Simulated Playback,” Call Level, page 45](#)

Parameter Value Linking

After you import an event stream into the Virtual Script Editor, you are ready to create the virtual script. Using the Target Parameters and Source Parameters panes, you complete the task of value linking. Value linking ensures that the virtual script can pass parameter values from one API to another. You identify a value-containing parameter in a source parameter API call and link the value to a target parameter in another API call. This process ensures the passing of a parameter value from one API to another API that requires the value.

In addition, the values contained in many API call parameters must be dynamic. For example, each time a user performs voucher entry, the PeopleSoft software creates a new batch number, a function that is essential to prevent the creation of duplicate keys. Value linking ensures that the Virtual Script Player can simulate this function. When you link the parameter value of two API calls, the Virtual Script Editor stores the value as a variable, and the value changes each time you run a virtual script.

For example, a script might call the business function `BatchOpenOnInitialization`. For the parameter `ICU`, which is the batch number, suppose the API returns the value 5056. In turn, the script might call the business function `BeginDoc`, which uses the value 5056 as an input to the `ICU` parameter. To simulate multiple script playback, the value 5056 must change in order to reflect the new batch numbers returned each time people using the system make these API calls. As long as you have linked the parameters, the batch number parameter value will change each time you run a virtual script.

Value linking simulates the application logic that is used to run PeopleSoft software operations. It codifies the relationship between one API call and another. When you run the virtual script, the Virtual Script Editor passes to the Virtual Script Player the ID number of the source parameter that you link to the target parameter. The Virtual Script Player uses this information to pass parameter values between API calls.

Several types of data necessary to run a virtual script are candidates for value linking:

- The client host name, which could change any time a script is played back.
- Next numbers, which must change each time a script is run in order to avoid producing duplicate data that would break the script.
- Valid values lists used in PeopleSoft AutoPilot scripts, which must be designated as such in a PeopleSoft Virtual AutoPilot script so that, during run time, the Virtual Script Player draws new values from the list rather than using the same value repeatedly.

Source and Target Parameter Identification

The Virtual Script Editor provides detailed information about API calls in the event stream when you click the Link Parameters button in the toolbar. The Virtual Script Editor identifies the API calls made during script playback as source parameters or target parameters. A source parameter contains a value that the system passes to a parameter in another API call. The parameter receiving the value is the target parameter.

Information about the source and target parameters appears in separate panes of the Virtual Script Editor form. Each of the panes contains the following information about API calls made during script playback:

Column Heading in Target Parameters Pane and in Source Parameters Pane	Information Displayed
ID	Displays in the source parameter pane a value that identifies that parameter. If you value-link the source parameter to a target parameter, the source parameter ID value appears next to the target parameter, along with a chain-link symbol indicating that you have linked the parameters.
Start Time	Specifies the time the event occurred during playback.
Thread	Identifies the thread generated by the system's run-time engine in which the event occurred.
Label	Identifies the data dictionary alias of the parameter.
Value	Shows the value of the parameter contained within the JDB or CallObject API call.
Comment	Contains the variable name of a business function parameter and the type of data that it contains.
Event	Identifies the specific JDB or CallObject API called or PeopleSoft AutoPilot event in which a value was entered.

To see the complete set of parameters for an API call that occurred during PeopleSoft AutoPilot playback, click an item in either pane. The Virtual Script Editor displays the parameter names and values for the selected call. Arrows indicate the direction of the flow of data.

The detail panes provide a snapshot of the API calls that the applications generated during PeopleSoft AutoPilot playback. You can examine the parameter values and the flow of the data to help determine, for example, a parameter value used in one API call that the system passed to another API call later in the script.

To find the parameter of an API call in an event line, you might have to click a node in the detail pane to expand a tree. For example, for a JDB call, find the Value node in the detail pane and expand the tree to expose all of the column parameters in the database table. You then can search the column parameters for the source parameter for which you are looking.

Automatic Value Linking of API Call Parameters

When you click the Link Parameters button on the Virtual Script Editor, the tool automatically links some source parameters of API calls to target parameters of other API calls. The Virtual Script Editor accomplishes this automatic linking according to a set of rules. These rules govern the automatic linking of API calls:

- Data must have been entered in PeopleSoft AutoPilot.
- The value of the target parameter must exactly match the value of the source parameter.
- The data dictionary ID of the target parameter must exactly match the data dictionary ID of the source parameter.

The Virtual Script Editor finds those parameters that meet each of these conditions and automatically links them.

Looping Identification

PeopleSoft Virtual AutoPilot requires a method to handle repetitive processing, such as that which occurs when you click the Find button on a Find/Browse form to perform a database inquiry. In this situation, a JDB Fetch call might return any number of values from the database. Looping rules provide a way to identify these repetitive retrievals. Having identified all of them in a single step, you can more easily link the values of source and target parameters.

You can specify the number of times you want the API to return to the database to retrieve values to be used as parameters in the script. This capability enables you to more accurately simulate the load placed on the system. You can increase or decrease this number, but the actual number of matches that the API returns, based on the inquiry command that you write in the PeopleSoft AutoPilot script, will likely determine the number of loops that you specify.

Manual Value Linking of API Call Parameters

If the source and target parameter do not meet all three of the following you must link them manually by clicking a parameter in each pane, and then clicking the Link Parameters button in the Virtual Script Editor tool bar.

- Source parameter must come from an Autopilot event (type to grid cell, for example)
- Data dictionary items (called *label* in Virtual Script Editor) must match in source and target
- Values must match in source and target

In deciding the target parameter value to link to a source parameter value, you:

- Match data dictionary aliases
- Match parameter values
- Select, in general, the event in the Source Parameter pane whose start time most closely matches the start time of the event in the Target Parameters pane

You do not have to link the values of source and target parameters when:

- APIs do not contain data dictionary items
- An API call returns a zero or null value for the source parameter that might be value-linked to a target
- The data flow of the source parameter is indicated as bi-directional, but the input value and the return values are the same

You code as literal the values of any parameters that meet at least one of these criteria by clicking the Mark Literal button on the Virtual Script Editor form.

The content of the PeopleSoft AutoPilot script also plays an important role in your decisions on value linking. If the PeopleSoft AutoPilot script that you write contains a literal value that the script writes to a grid column or header control, you cannot make that literal value dynamic by linking. The Virtual Script Player will be forced to use that literal value repeatedly during PeopleSoft Virtual AutoPilot script playback.

Because you cannot make literal values dynamic, avoid using them often in a PeopleSoft Virtual AutoPilot script. The entry of the same value to a grid column or header control by multiple users does not accurately simulate the way people use the system. To set up a more realistic scenario when you write the PeopleSoft AutoPilot script, create valid values lists containing more than one value. During PeopleSoft Virtual AutoPilot script playback, the Virtual Script Player goes to the `.atd` directory on your hard drive to retrieve the list's values, and then it cycles through them, entering a different value in each simulated playback session until it reaches the end of the list, when it returns to the top of the list and repeats the cycle.

Dynamic Loop Creation

You create a dynamic loop in the PeopleSoft AutoPilot script by writing a command to press the Find button on a Find/Browse form. This command triggers a string of JDB API calls, culminating in a Fetch call.

PeopleSoft AutoPilot and PeopleSoft software code records all of these events during playback; however, thousands of events might exist in an PeopleSoft AutoPilot script. The Virtual Script Editor offers an easy way to locate the PeopleSoft AutoPilot press Find button event and the repetitive processing that occurred because of the event. When you type the word Find in the locator space in the Virtual Script Editor form, the Virtual Script Editor highlights the first line in the event stream pane that contains the word Find. Then, you can scroll down the pane to the Fetch call.

Following the Fetch call, you can scroll through the event stream to locate the series of calls that resulted from the Fetch. This series of calls constitutes the loop.

Dynamic Loop Designation

The Virtual Script Editor enables you to designate dynamic loops in the PeopleSoft Virtual AutoPilot script. By doing so, you add looping rules to the script. These rules enable the Virtual Script Player to perform the repetitive processing that the system performs.

When you right-click the Fetch line in the event stream and select Add Loop, the Virtual Script Editor produces a Dynamic Loop Manager form, which you use to apply a dynamic loop and to establish the rules by which the Virtual Script Player manages the loop at virtual run time.

You can instruct the Virtual Script Player to run the inquiry loop until the data runs out, or you can instruct it to loop a specific number of times.

When you click the Apply Loop button in the Dynamic Loop Manager, you establish the looping rule. The Virtual Script Editor indicates the loop in the event stream, in the Source Parameters pane, and in the Target Parameters pane by graying the sequence of API calls that are part of the loop.

Dynamic Loop Editing

You can edit established looping rules before generating a PeopleSoft Virtual AutoPilot script. You also can undo the loop if, for example, a series of calls does not constitute a loop. The ability to edit dynamic loops provides an added measure of control over the creation of PeopleSoft Virtual AutoPilot scripts.

To edit the loop, you right-click the Fetch command line in the event pane and select Edit Loop Details. On the Dynamic Loop Manager form, you can change the number of times you want the script to loop, or you can select the Undo Loop option to remove the loop.

HRequest Handle Value Linking

The Virtual Script Editor automatically stores hRequest handle parameter values for JDB API calls. This value represents the address of a memory block that the system allocates for storing information about an open table. The address provides you with entry to the database each time you need to open a table to perform a Fetch, FetchKeyed, SelectKeyed, FetchMatchingKey, or CloseTable function. However, when you create a PeopleSoft Virtual AutoPilot script and play it back, the hRequest handle parameter value probably changes. Playback could not continue if this value were constant.

The Virtual Script Editor handles the problem by storing the hRequest handle parameter value as a variable and passing the variable to the Virtual Script Player during playback. The value of the hRequest handle variable changes to reflect the new address of a database table opened during script playback.

You can view the hRequest handle returned from the original API database call by clicking a call in the Source Parameters pane and viewing the details of the call. The Virtual Script Editor displays in the detail pane the request handle returned from the OpenTable API call.

If a database API call, such as OpenTable, leads to additional API calls, such as FetchKeyed and CloseTable, the Virtual Script Player passes the new memory address of the opened table to these subsequent calls. During virtual playback, the subsequent APIs use the new handle to run SQL statements and to close the table.

Thread Identification

The Virtual Script Editor also stores the idThread numbers that PeopleSoft AutoPilot gathers into the event stream during script playback. These identifier numbers represent the synchronous and asynchronous threads generated by the runtime engine. The runtime engine assigns each event to a thread and tags each thread with a number.

During virtual script playback, the Virtual Script Editor passes idThread parameters to the Virtual Script Player, which assigns different idThreads to each event and associates each script event with its new identifier.

Note. During Virtual User Tool script playback, the Virtual Script Player rennumbers the original threads generated during PeopleSoft AutoPilot script playback. The Virtual Script Editor's role is to store the thread identification information and to pass it on through the virtual script.

Timing Interval Maintenance

The Virtual Script Editor also automatically handles problems of timing that might emerge in the creation of a Virtual User Tool script. The time-stamped event stream log of events captures the length of time elapsed between each event. However, after you create a Virtual User Tool script, you do not know the different scenarios in which the Virtual User Tool script runs. For example:

- The workstation on which the script runs might be simulating 50 users.
- The power of the workstation might differ from the one on which the original script data was captured.
- The server against which the Virtual User Tool script runs might be more or less powerful than the server against which the original script ran.

These factors combine to make it likely that the time required by a Virtual User Tool script to run will differ from the time that the original script required to run.

The Virtual Script Editor handles this problem by preserving in the virtual script the time intervals that existed between events when you ran the original script. The time intervals represent the length of time required to carry out the processing between events. Thus, even if an API call during virtual script playback takes longer to carry out than the API call in the original script, the Virtual Script Player preserves the original time difference between one API call and the next.

The Virtual Script Player initialization file also contains timing parameters that govern the playback of the Virtual User Tool script. You can adjust, to a limited extent, some of these parameters; for example, you can adjust how fast the Virtual User Tool script plays back.

See Also

[Chapter 3, “Understanding PeopleSoft Virtual AutoPilot Components,” Virtual Script Player Initialization File Parameters, page 19](#)

[Chapter 6, “Understanding Special Considerations for Simulated Playback,” Playback Timing, page 43](#)

PeopleSoft Virtual AutoPilot Script Generation

The PeopleSoft Virtual AutoPilot script is the output from the Virtual Script Editor and the input to the Virtual Script Player. PeopleSoft Virtual AutoPilot scripts appear in text file form with a header and the edited list of events that you captured during script playback, imported into the Virtual Script Editor, and edited, both manually and automatically.

For ease and consistency of interpretation, each event in the script is structured in a particular way. For example, each event begins with the letter *e* and is followed by a unique identifying number. In addition, each script identifies the environment and the network user, and contains an open table handle. However, it is not necessary that you look at a PeopleSoft Virtual AutoPilot script in order to run it.

PeopleSoft Virtual AutoPilot classifies these three types of events and identifies them as such in the script:

Event	Description
Functions	Includes JDB and CallObject APIs
Assignment statements	Refers to values typed in PeopleSoft AutoPilot
Conditional	Tests/branches (if/then statements)

PeopleSoft Virtual AutoPilot divides each event into parts and, in turn, identifies each of the parts based on an assigned format and a unique value. In short, the PeopleSoft Virtual AutoPilot script contains the details necessary for the Virtual Script Player to simulate running the PeopleSoft software kernel.

PeopleSoft Virtual AutoPilot identifies transaction boundaries, which you can set in the original script by designating a script command as the start of the transaction and another script command as the end of the transaction. Setting transaction boundaries can help you to analyze system performance when running a series of tasks.

Virtual Script Player

This section discusses:

- Virtual Script Player features.
- Virtual Script Player initialization file parameters.
- Virtual Script Player command line.
- Environment initialization.
- Modes of operation.
- Preprocessing of valid values list data.
- Valid values processing.
- Date formatting.
- Script failure.
- Virtual Script Player limitations.

Virtual Script Player Features

The Virtual Script Player uses the PeopleSoft Virtual AutoPilot script that you generate in the Virtual Script Editor to simulate the concurrent activities of one or more PeopleSoft software users. It bypasses the presentation layer of PeopleSoft software and reproduces the PeopleSoft application calls to the JDB and CallObject middleware. This reproduction is based on the timing and the sequencing of data in the event stream that you generate with PeopleSoft AutoPilot, manipulate in the Virtual Script Editor, and generate in modified form in the PeopleSoft Virtual AutoPilot script. In essence, the Virtual Script Player assumes the role of the PeopleSoft software runtime engine.

Virtual Script Player Initialization File Parameters

The `vap.ini` file is a text file that contains the parameters that define the way that the Virtual Script Player runs. These parameters govern the paths that the Virtual Script Player follows to find files, synchronize playback timing, and set playback speed.

You can change the parameters, within established limits, to set the way the PeopleSoft Virtual AutoPilot scripts play.

Command

The Command section of the `vap.ini` file contains the parameters that are necessary for interaction between Virtual Runner, which manages script playback, and the Virtual Script Player, which runs playback. These parameters specify:

- User ID and password
- Environment
- Script name
- Log file of summary playback statistics
- Location of the Virtual Script Player executable

This table summarizes the [COMMAND] parameters and the meaning of each one:

Parameter	Meaning
UserID=	PeopleSoft Virtual AutoPilot user ID. Override on command line by entering -u and a user ID.
Password=	Password for PeopleSoft Virtual AutoPilot user. Override on command line by entering -p and a password.
Environment=	Environment for PeopleSoft Virtual AutoPilot script playback. Override on command line by entering -e and an environment.
Script=playscript.vsx	Name of PeopleSoft Virtual AutoPilot script (user can specify full path name for script here).

Parameter	Meaning
Common log=	Log file to which Virtual Script Player will write summary statistics for all playback sessions. Default folder is Vap_logs. Used only with Virtual Runner.
Binname=d:\b7\system\bin32\vapplayer.exe	Path by which Virtual Runner finds the Virtual Script Player executable.

Paths

The Path section of the `vap.ini` file identifies the directories for files that are needed by the Virtual Script Player. The contents of the needed files are:

- Log file, which gives detailed information about each PeopleSoft Virtual AutoPilot script playback session, the script name, and a line-by-line summary of each event in the script. The Virtual Script Player logs each event as it completes. The file also includes the start time and the date of the log.
- PeopleSoft Virtual AutoPilot script file, which stores all scripts that you might use for virtual playback.
- Valid values list file, which stores any valid values lists that the Virtual Script Player draws on for input values to run business functions. The Virtual Script Player uses valid values lists to get a new value each time it runs a business function.

The default file paths are as follows:

File/Contents	Parameter in vap.ini file	Default Path
Log of PeopleSoft Virtual AutoPilot playback events and messages	LogDirectory	c:\autopilot\VAP_LOGS
PeopleSoft Virtual AutoPilot scripts	VirtualScripts	c:\autopilot\VSX
Valid value lists	ValidValueLists	c:\autopilot\ATD

Timing

The Timing initialization parameters of the `vap.ini` file help you specify the terms under which PeopleSoft Virtual AutoPilot scripts play back:

- Rendezvous of multiple playback sessions, to control the amount of time the Virtual Script player delays a playback session following a rendezvous of multiple scripts running on a single workstation
- Synchronization of playback events, to set limits on the amount of time that threads can be inactive, events can occur behind the start time scheduled by the script, or that a thread has to wait for an API value or a handle parameter
- Playback speed, to adjust the amount of time between events to compensate for a fast or slow client workstation

This table lists the PeopleSoft Virtual AutoPilot timing initialization parameters, their default values, what they govern, and the kind of timing factor to which they relate:

Parameter Name	Default Value	Meaning	Timing Factor
RandomDelayMax	0 seconds; can be set as high as 3,600	Enables user to set a maximum period that the Virtual Script Player will wait after the LoadRunner OWLogin rendezvous and environment initialization to begin each playback session. The default value means that following rendezvous, each player session proceeds without delay.	Rendezvous of multiple playback sessions
IMaxSleep	10,000 milliseconds	Establishes an upper limit on thread sleep time. Inactive threads must check on system status at least this often. If errors require the Virtual Script Player to shut down all threads, the parameter also determines the maximum amount of time required for the Player to shut down.	Playback synchronization
ITooLate	200 milliseconds; set higher in for debugging	The latest that any event can be run after the script schedules its start without causing virtual script playback to terminate.	Playback synchronization
ITimeout	60 seconds	Maximum number of seconds that an event has to run. If that number exceeds the parameter, Virtual Script Player terminates the playback session.	Playback synchronization
ClientSpeedFactor	100	Controls timing between script events by a constant factor. Decreasing the value of the parameter decreases the time between events.	Playback speed

Log

You use the Log section of the vap.ini file to specify the type of messages that the Virtual Script Player writes to a log file during a PeopleSoft Virtual AutoPilot script playback session. These messages can be important for debugging purposes. This table summarizes the available log parameters and the debug message level that each one represents:

Log Parameter	Debug Message Level
31	Maximum log output; flush log file after each message (LoadRunner excluded)

Log Parameter	Debug Message Level
15	Parameter values and value substitutions
7	Error, warning, and status messages
3	Error and warning messages
1	Error messages only
0	Minimal messages

Note. You can cause the log file buffer to flush after every message by adding 16 to any parameter less than 31. However, you should not routinely do this, as flushing frequently increases file system overhead. For the same reason, you should not routinely set the log parameter at 31.

See Also

[Chapter 5, “Running Virtual Scripts,” Launching and Managing Multiple Script Playback, page 40](#)

[Chapter 6, “Understanding Special Considerations for Simulated Playback,” Playback Problem Simulation, page 43](#)

Virtual Script Player Command Line

You can launch the Virtual Script Player from LoadRunner, from Virtual Runner, or from the DOS command line. The command line must have entries that specify the user, the user’s password, the environment, and the script name with a default extension of .vsx for any PeopleSoft Virtual AutoPilot script, although this extension is not required.

These four entries are required on the command line:

Command Line Abbreviation	Meaning	Sample Entry
-u	User	ce5791892
-p	PeopleSoft software user ID	-p pwd
-e	Environment	-e PDEV_VAP
-s	Script Name	-s voucherentry100.vsx

Environment Initialization

The Virtual Script Player does not immediately begin playing a PeopleSoft Virtual AutoPilot script upon launch from the DOS command line, from Virtual Runner, or from LoadRunner. In fact, the Virtual Script Player reads the script and runs events that generate a PeopleSoft software environment structure. The data that drives the generation of the environment comes from entries in the command line. For example, one user might create a PeopleSoft Virtual AutoPilot script, but another user might play the script. During initialization, the Virtual Script Player passes in the user ID of the user playing the script, thereby creating the proper environment. Therefore, you can run the Virtual Script Player in an environment different from the one in which you or someone else created the PeopleSoft Virtual AutoPilot script.

Environment initialization takes about 15 to 30 seconds. LoadRunner regards this passage as initializing time, while the DOS command line reads it as busy activity.

Modes of Operation

The Virtual Script Player automatically detects whether you have launched a PeopleSoft Virtual AutoPilot script from the DOS command line, from Virtual Runner, or from LoadRunner. If LoadRunner launches the script, the Virtual Script Player responds to stop/pause commands and sends transaction times and log output to LoadRunner. In addition, the Virtual Script Player completes a LoadRunner rendezvous just after it has initialized the system environment.

Preprocessing of Valid Values List Data

The preprocessing capability of the Virtual User Tool works with the Virtual Script Editor and Virtual Script Player to use valid values lists during script playback. You must mark valid value lists because virtual playback requires the values contained in these lists as parameters for API database calls.

When a PeopleSoft Virtual AutoPilot script specifies that a particular value originates in an PeopleSoft AutoPilot valid values list, the Virtual Script Player reads the valid values list file. All valid values lists are identified by the extension `.atd`. Before the Virtual Script Player plays the script, it performs preprocessing that includes looking up the database values in the valid values list and storing them until they are required as parameters for API calls. When the Virtual Script Player runs the PeopleSoft Virtual AutoPilot script, the stored list supplies the parameters needed for JDB or CallObject calls.

Preprocessing plays an important role in the PeopleSoft Virtual AutoPilot scheme because it takes care of the lookup and load of the valid values that the Virtual Script Player needs for PeopleSoft Virtual AutoPilot script execution. This ensures that the required values exist before playback. If the Virtual Script Player had to run database lookups at the time of script playback, the result would be artificial load on the database, which would, in turn, distort the simulation of activity that PeopleSoft Virtual AutoPilot seeks to achieve.

Valid Values List Processing

The Virtual Script Player defines the location of any valid values lists that are part of the PeopleSoft Virtual AutoPilot script in the `vap.ini` file. The Virtual Script Player reads valid values lists that are 64K or smaller into memory. If the file is larger than 64K, the Virtual Script Player must read it from the file. During virtual playback, if the Virtual Script Player reaches the end of a valid values list, it starts back at the beginning of the list, reuses the first value, and continues in sequence until virtual playback is complete.

Date Formatting

The Virtual Script Player expects a certain format for date strings for valid value lists and for literal typed-in values from PeopleSoft AutoPilot. Therefore, the Virtual Script Player supports different date formats that might appear in the PeopleSoft Virtual AutoPilot script, including `mm/dd/yyyy` and Julian date strings (that is, `102343` or `12/09/2002`).

Important! The Virtual Script Editor correctly formats date entries for literal values but not for date entries in valid value lists.

Script Failure

Script failure might occur during the initialization process. For example, a branch event in the script might not refer to a valid event, or the events might not occur in the same thread. In the first example, the script fails before it is launched because the Virtual Script Player cannot validate the events. On initialization, the Virtual Script Player also validates function parameters. For example, a parameter such as Fetch might accept only 0 (zero) or 1 as values. If a different value is used, validation fails and, thus, the script fails before launching.

If the script fails during playback, the failure shuts down script processing. For most API calls, failure to return a success code causes the playback process to halt. The shutdown occurs without user intervention. LoadRunner, for example, returns a failure report, and the Virtual Script Player sends an error message to the log file, for example: `LoadRunner/Test Name/Local1/Subdirectory Name`. One subdirectory exists for every LoadRunner test session, which means that 50 simulated user test sessions produce 50 subdirectories.

If you launch the Virtual Script Player from a command line or from Virtual Runner and script failure occurs, no error message appears on the screen. You must open the log file that stores the test session results and examine the messages, a task you complete by searching on the keyword Error.

Virtual Script Player Limitations

The overriding consideration for PeopleSoft Virtual AutoPilot script playback is that client workstations must not impede the playback process. You must determine how many processes the workstation can realistically support, based on an analysis of workstation memory and CPU capability. Running either Task Manager or Performance Monitor can assess these capabilities.

Other Virtual Script Player limitations are hard-coded. If the Virtual Script Player gets a script that exceeds these limitations, you receive error messages that require a service pack to address. First, the Virtual Script Player supports up to 30 active user handles and 60 active request handles per session. Second, the Virtual Script Player can process only a certain number of status messages per second under LoadRunner. If the playback exceeds that number, some of the messages are lost, but the Virtual Script Player does not shut down.

VSMEditor

This section discusses:

- VSMEditor features.
- All Virtual Scripts list box.
- Master Scripts list box.
- VSM files.

VSM Editor Features

After you create a number of PeopleSoft Virtual AutoPilot scripts, the VSMEditor enables you to concatenate any number of those scripts into a single master script. Concatenating single scripts into a single master script is advantageous because you can run a series of unrelated tasks during testing.

You control the VSMEditor from the VSMEditor form, which you access by clicking the VSMEditor executable.

All Virtual Scripts List Box

The All Virtual Scripts list box contains all PeopleSoft Virtual AutoPilot script files that you have created; these files have a .vsx extension. In addition, any master scripts that you have created appear in this list box; master scripts have a .vsm extension. The location of any PeopleSoft Virtual AutoPilot script files that appear in the All Virtual Scripts list box is determined by the value of the VirtualScripts parameter of the PATHS section in the vap.ini.

You enter the path to the location of your virtual scripts to set the VirtualScripts parameter.

You can use any script in the All Virtual Scripts list box to create a master script. You create the script concatenation by choosing one of the scripts in the box and then holding down the Control key or the Shift key to select additional scripts.

You click the Add button to add the files that you chose to the Master Script list box.

Master Scripts List Box

The Master Scripts list box shows all the scripts that you have currently chosen for addition to a new .vsm (virtual script master) file. You can manipulate the script list in the Master Scripts list box by using the buttons adjacent to the box:

Remove	Deletes the chosen script from the Master Scripts list
Move Up and Move Down	Shifts the position of the selected script in the list
Remove All	Deletes all scripts from the list
Save Master Scripts	Saves the list of scripts as a .vsm file

VSM Files

The VSMEditor creates a .vsm text file when you save a master script. You can change these files only through the VSMEditor because the file contains a checksum value that verifies the file's integrity. The PeopleSoft Virtual AutoPilot scripts always run in the sequence listed in the .vsm file. However, the first script to run is chosen randomly when the RandomStart parameter in the text file is set to 1.

The actual PeopleSoft Virtual AutoPilot scripts are not included in the .vsm master file. Therefore, you should not delete scripts from the folder that contains the .vsx files.

Virtual Runner

This section discusses:

- Virtual Runner features.
- Player session columns.
- Actions tools.

Virtual Runner Features

Virtual Runner controls the Virtual Script Player sessions on a single workstation and provides the following command and control functions for Virtual Script Player testing:

- Enables users to start one or more Virtual Script Player sessions on a single workstation
- Enables users to play multiple iterations of a single script
- Reports Player session status (pass/fail) to user
- Summarizes performance statistics over all Virtual Script Player sessions in a test

You use the action tools and the columns in the detail area of the Virtual Runner form to manage the Virtual Runner session.

Player Session Columns

After you finish setting up the parameters for the Virtual Script Player session, Virtual Runner displays the names of the scripts that you want to run. Initially, the status of the script is Down, indicating that you have not yet run it.

After you run a test, Virtual Runner changes the status to indicate success or failure.

Each column displays information about the Virtual Script Player sessions. These descriptions summarize the purpose of each player session column:

- The State column indicates the current state of the player session. For example, after you successfully execute a player session, this column displays the word Success.
- The Env column indicates the specified environment for the current session. The environment is specified using the Options button or when you use the Virtual Runner Wizard.
- The User column displays the User name that you specified using the Options button or the Virtual Runner Wizard.
- The Repeat column specifies the number of times the script is repeated when you execute the player session. You specify this parameter when you use the Virtual Runner Wizard.
- The Script column specifies the path and file name of the script for the current player session. You specify these parameters when you use the Virtual Runner Wizard.

Actions Tools

You use the Actions tools to set up and launch a Virtual Runner session. You can select the scripts that you want to run as well as the number of script playback iterations. In addition, following playback you can access a log that contains pertinent information about the playback session.

The Virtual Runner toolbar contains these six buttons:

Option	Enables you to specify the user ID, password, and environment for the virtual playback session.
Wizard	Directs you through the process of specifying all the Virtual Script Player session parameters, including the number of scripts to run and the script playback iterations.
Run	Runs the virtual script playback session.
Log	Displays the Log Viewer screen, which provides information about the last completed Virtual Script Player session.
Report	Prints a copy of the Virtual Script Player session log information.

Close

Closes the Virtual Runner window after you have decided whether to save the results of the Virtual Script Player session.

CHAPTER 4

Creating Virtual Scripts

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

- Capture and import test results.
- Edit virtual scripts.

Understanding Script Creation

You use PeopleSoft Virtual AutoPilot to create a script that simulates the activities of PeopleSoft software as it handles the workload generated by many users. To achieve this goal, you use two key components of PeopleSoft automated testing tools architecture:

- PeopleSoft AutoPilot
- Virtual Script Editor

Using these two tools, you accomplish this sequence of tasks to create a virtual script:

1. Create a PeopleSoft AutoPilot script.
2. Run the PeopleSoft AutoPilot script with playback configured so that you can capture system and PeopleSoft AutoPilot data.
3. Import the event stream into the Virtual Script Editor.
4. Create value links between source parameters of API calls and the target parameters of other API calls to ensure that usable data flows between API calls when you run the virtual script.
5. Add loops to the PeopleSoft Virtual AutoPilot script to account for repetitive processing, such as data retrieval.
6. Generate the PeopleSoft Virtual AutoPilot script, which the Virtual Script Player runs.

After you create a virtual script, the Virtual Script Player runs the script. You use Virtual Runner or LoadRunner to manage the number of sessions, either from a single workstation or from multiple workstations.

Capturing and Importing Test Results

This section provides an overview of test results and discusses how to:

- Capture test results.
- Import test results.

- View test results.

Understanding Test Results

PeopleSoft AutoPilot enables you to create scripts that test PeopleSoft software applications. When you create a script, you can configure PeopleSoft AutoPilot's playback function so that it captures and saves the results of the playback session, which it stores in the PeopleSoft AutoPilot Playback Results Detail Table (F97214) as an event stream.

You can view the playback results in a variety of ways. You can view the event stream alone, you can view details of individual events, or you can view timing information about groups of events and thread identifiers, displayed in a horizontal bar graph.

The data that PeopleSoft AutoPilot captures provides the raw material for the PeopleSoft Virtual AutoPilot script. After you capture PeopleSoft AutoPilot script data, you import it to the Virtual Script Editor so that you can prepare a virtual script.

Capturing Test Results

To gather the raw data for a virtual script, you must first write and run an PeopleSoft AutoPilot script and capture the results of the playback as an event stream. You use the Tools option in the menu bar of the PeopleSoft AutoPilot form to set up the capture mechanism.

To capture test results:

1. From the desktop or the appropriate directory, double-click the PeopleSoft AutoPilot executable. Create the PeopleSoft AutoPilot script or open an existing script.
2. From the File menu, select Open to open an PeopleSoft AutoPilot script.

Important! When you run the script, it must sign on to a PeopleSoft software environment. A script that does not include this signon does not function correctly in PeopleSoft Virtual AutoPilot because it does not contain the data required for the Virtual Script Player to initialize the environment.

3. From the Tools menu, select Options.
4. On the Options form, select the Playback tab.
5. Select these options:
 - Save Results Data after Playback
 - Display Results Data after Playback
6. In the Event Stream Capture Level portion of the Playback tab, select Level 1 API calls.

Note. If you want to capture more script playback events, select the All API call levels option. Remember that you generate a much larger event stream if you select this option.

7. Click OK.
8. Save the PeopleSoft AutoPilot script.
9. In the PeopleSoft AutoPilot menu bar, click Play and select Play From Top.

PeopleSoft AutoPilot runs the script. The Play From Top command generates test results for DENPCX (where DENPCX=the name of the machine on which PeopleSoft AutoPilot resides). The PeopleSoft AutoPilot Results form displays detailed information about the playback session.

10. Click Close to close the Test Results window.
11. Click File/Exit to close PeopleSoft AutoPilot.

Importing Test Results

After you have run an PeopleSoft AutoPilot script and saved the playback results, you can import the event stream into the Virtual Script Editor. Importing the event stream enables you to use the Virtual Script Editor to forge value links between the source and target parameters of API calls; to identify, designate, and edit repetitive processing; and to generate a virtual script.

To import playback results into the Virtual Script Editor:

1. From the desktop or the appropriate directory, double-click the Virtual Script Editor executable.
2. On Virtual Script Editor, select a script to import.
3. Click the Import button.

After the Virtual Script Editor imports the script, an APEdit dialog box appears, confirming that the import was successful.

4. In the Virtual Script Editor dialog box, click OK.

Important! If you attempt to import an PeopleSoft AutoPilot script that you captured without a system signon, PeopleSoft Virtual AutoPilot displays a warning issue.

If a message appears, you should recapture the data, making sure that you sign on to the system through PeopleSoft AutoPilot. To do so, close PeopleSoft software, and then launch the PeopleSoft AutoPilot script. PeopleSoft AutoPilot handles your system signon.

Viewing Test Results

After you successfully import the results of a script playback, the event stream appears in the detail area of the Virtual Script Editor form.

Important! An exclamation point next to a start time (in the Start column) in a line of the event stream indicates that an error occurred during data capture.

If you find exclamation points in the event stream, you should investigate the possible causes for the error, and then edit and rerun the PeopleSoft AutoPilot script.

To view the event stream:

1. From the desktop or the appropriate directory, double-click the Virtual Script Editor executable.
2. In the toolbar of the Virtual Script Editor form, click one of these options:
 - Details
 - Graph
 - Both

To view the event stream alone, click the Details option in the toolbar. To view categories of playback events, thread activity, or both represented in a horizontal bar graph by duration and time of occurrence, click the Graph option, and then click the scroll bar button in the form to select either View Graph by Message Type or View Graph by Thread ID. Click the Both option to view both the linear event stream and the horizontal bar graph representation.

Editing Virtual Scripts

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

- Use the Find feature.
- Add loops.
- Value-link parameters.
- Link values in inquiry scripts.
- Link values in entry scripts.
- Generate PeopleSoft Virtual AutoPilot scripts.
- Create master scripts.

Understanding Virtual Script Generation

After you import an event stream into the Virtual Script Editor, you can create a virtual script by completing these two primary tasks:

- Adding loops
- Creating value links

After you finish these tasks, you generate the virtual script. The Virtual Script Editor passes the loop and value link information, as well as playback information that it stores automatically, to the Virtual Script Player, which runs the virtual script.

When you add loops, you define the number of data retrievals that PeopleSoft Virtual AutoPilot performs when you run the virtual script. You can limit the number of loops, or you can ensure that the Virtual Script Player loops until no more data is available.

When you create value links, you ensure that data necessary to run the virtual script flows dynamically between parameters in API calls. For example, you must value link APIs that use next numbers so that the Virtual Script Player retrieves the appropriate next number during virtual playback. If you fail to value link the next number parameter in this scenario, the Virtual Script Player passes the same value used in the original script to the API parameter that requires it, which causes a duplicate key error. When you forge a value link, the Virtual Script Editor stores the parameter value in a variable, which ensures that the value changes each time you run the script, preventing duplicate keys and data contention.

The PeopleSoft Virtual AutoPilot set also enables you to concatenate virtual scripts into a master script list using the VSMEditor. Using a master script enables you to test more than one script in a single virtual script playback session.

Using the Find Feature

You use the Find feature in the Virtual Script Editor to search for parameters that you will need to value link to create the PeopleSoft Virtual AutoPilot script.

To use the find feature:

1. Click inside the pane you want to search.
2. Type a value in the Find control.

To search for valid value list values to link, enter a list value to the Find control.

To find loops to process, search for JDB Fetch calls.

To find data dictionary aliases, enter a data dictionary alias, such as AN8.

3. Check the Case Sensitive button, if necessary.
4. Press Enter to run the search.

The Virtual Script Editor finds the first parameter with a data dictionary alias that matches the search criterion and marks it with an arrow.

Note. As you click a button to link or perform another task, you might lose the focus to the pane. Be sure to reset the focus to the pane you are searching, if necessary, by clicking inside the pane.

Adding Loops

Loops in PeopleSoft Virtual AutoPilot scripts simulate how PeopleSoft software functions when it performs inquiries. Without loops, the Virtual Script Player tries to fetch the same number of records that were retrieved during the original playback, regardless of selection criteria or available data. Loops also enable you to identify and reduce the number of events that must be value linked. Because of this advantage, you might want to generate loops before performing the value linking function.

To add a loop:

1. From the desktop or the appropriate directory, double-click the Virtual Script Editor executable.
2. Use the Find feature to search the event stream event pane for AUT: Press Button [Find].
3. From the Find statement in the event stream event pane, move the cursor down the list of events until you find a Fetch statement in the Message column.
4. Right-click and select Add Loop.

The Dynamic Loop Manager form appears.

5. On the Dynamic Loop Manager form, click one of these options:
 - Loop until No Data
Click if you want PeopleSoft Virtual AutoPilot to exhaust data retrieval.
 - Loop X times
Specifies the number of loops you want the script to perform.
6. Click Apply Loop to add the loop.
7. To undo the loop, find the loop, launch the Dynamic Loop Manager form, and click the Undo Loop button.

Note. PeopleSoft Virtual AutoPilot colors events inside a loop light gray in the event stream, source, and target panes. You do not need to consider these events when you perform value linking.

Value-Linking Parameters

Value linking enables data to flow from function to function within PeopleSoft software. For a PeopleSoft Virtual AutoPilot script to accurately simulate system activities, it must not produce any duplicate key values in the system database. Therefore, for scripts that enter new data to the database or update existing data, at a minimum, you must value link all next number, job number, and batch number parameter values in the Virtual Script Editor. You can run simple inquiry scripts without any value linking, but these scripts might not accurately simulate system operations.

The Virtual Script Editor links some values automatically, but you must link others manually.

To run scripts accurately, you should always value link the parameters that:

- Pass the name of the machine on which you ran the original PeopleSoft AutoPilot script.
- Reference the date on which the original PeopleSoft AutoPilot script ran.
- Pass Next Numbers or serialized values (possibly labels of data items DOC, JOBS, MATH06, PYID, ICU).
- Use valid value list data. Linking these parameters ensures that the Virtual Script Player will use the .atd directory, where you store valid values list data as the source from which to retrieve data during virtual script playback.
- Contain the date that the script ran.

Note. You can use the Find feature to quickly find functions containing data to be linked. Click the column header to reorder the table (usually by label, value, or ID) to group like information.

To value link parameters:

1. From the desktop or the appropriate directory, double-click the Virtual Script Editor executable.
2. Import an PeopleSoft AutoPilot script into the Virtual Script Editor.
3. Click the Link Parameters button in the toolbar.

The Source Parameters pane and Target Parameters pane appear.

4. In the Target Parameters pane, select a target parameter line item.

The source parameters for that target display in the Source Parameters pane.

Note. Do not select the Show All option in the Source Parameters pane because doing so causes the Virtual Script Editor to display all the API calls in the pane.

5. To link a single parameter line item, select it and click the Link button.
6. To link all items in the script that match the source, target, label, and value parameters, select a representative parameter line item in the source pane and click the Link All button.

Note. Some parameters in the Target Parameters pane do not have a value from a source parameter. You can mark these as Literal using the Mark Literal Button. If you do not want to see the parameters that you have marked as Literal, click Link in the menu bar and select Filter Literals.

Linking Values in Inquiry Scripts

Because an inquiry does not change or update any data in the system, you are not required to forge value links between parameters in inquiry scripts. However, you should value link parameters that contain valid values list data to ensure that the data changes during playback of the virtual script.

If the script contains valid values data, you can run the virtual script, change the data, and run it again to extend the stress testing. You can change the data in the list without creating new value links. During virtual script playback, the Virtual Script Editor passes the new valid values list data to the Virtual Script Player for use in the appropriate parameters.

To link values in inquiry scripts:

1. From the desktop or the appropriate directory, double-click the Virtual Script Editor executable.

2. In the PeopleSoft Virtual AutoPilot Script Editor form, add any required loops to the PeopleSoft Virtual AutoPilot Script.
3. Find valid values list data in the event stream.
4. Link all source parameters containing valid values list data to the appropriate target parameters.
5. Document the data dictionary aliases that the Virtual Script Editor links.

Note. You find data dictionary aliases in the Label column of the Source Parameters pane and the Target Parameters pane.

Linking Values in Entry Scripts

Because entry scripts change or update system data, you are required to link values in entry scripts before you generate a virtual script. Value linking ensures that Virtual Script Player can pass values between parameters and that key parameter values change during virtual script playback, preventing record duplication.

To link values in entry scripts:

1. From the desktop or the appropriate directory, double-click the Virtual Script Editor executable.
2. In the Virtual Script Editor form, add any required loops to the PeopleSoft Virtual AutoPilot Script.
3. Find and link any parameters that pass the machine name on which the PeopleSoft AutoPilot script originally ran (these might be marked with CTID or MKEY data dictionary aliases or labels).
4. Find and link parameters that pass the date that the PeopleSoft AutoPilot script originally ran.
5. Find and link parameters that pass Next Numbers or serialized values (possibly data dictionary aliases of DOC, JOBS, MATH06, PYID, and ICU).
6. Find and link parameters that pass valid values list values.
7. Document the data dictionary aliases that the Virtual Script Editor links.

Generating PeopleSoft Virtual AutoPilot Scripts

To generate the PeopleSoft Virtual AutoPilot script:

1. From the desktop or the appropriate directory, double-click the Virtual Script Editor executable.
2. In the event stream event pane, click the Generate button.

When you press the Generate button, the Virtual Script Editor produces a virtual script, which the Virtual Script Player uses to simulate playback. A Script Log form appears following generation, summarizing the number of lines in the script and the number of errors, if any. You must generate an error-free script before you attempt to run it.

3. Assign a file name and location to which you want to save the generated script and click OK to begin script generation.

To generate the PeopleSoft Virtual AutoPilot script for playback using the Virtual Runner program, the script that you want to generate must be open in the script list pane.

After PeopleSoft Virtual AutoPilot generates the script, the Virtual Player Script Log form appears and displays information about the script generation, including the status (complete or incomplete), the number of lines, and the number of errors. If the Script Log form indicates that errors occurred during generation, you must investigate the error summaries that appear in the form and correct them by editing and rerunning the PeopleSoft AutoPilot script, and then repeating the steps for creating a virtual script.

4. Click OK to close the Virtual Player Script Log form.

Note. After you generate a virtual script, it is static. Any script changes that you make in PeopleSoft AutoPilot require re-editing and regeneration in Virtual User Tool. Thus, careful documentation of the editing process is critical to the production of repeatable results.

Creating Master Scripts

Using the VSMEEditor tool, you can concatenate PeopleSoft Virtual AutoPilot scripts into a single master script. Concatenation gives you another testing option: you can run test series of unrelated scripts.

To create a master script:

1. From the desktop or the appropriate directory, double-click the VSMEEditor executable.
2. In the All Virtual Scripts list box, select the script files that you want to include in the master script.
3. When you have chosen all the virtual script text files that you want, click the Add button.
VSMEEditor adds the script files to the Master Script list box.
4. Manipulate the list in the Master Script list box by using the buttons adjacent to the box to remove script files or to change their order.
5. When you have decided on the content and order of the master script, click the Save Master Script button.

The VSMEEditor saves the master script as a .vsm file. The file includes:

- Master script version
- Checksum value to verify file integrity
- RandomStart parameter (a value of 1 means that the first script to run is chosen randomly)
- List of PeopleSoft Virtual AutoPilot Script files

CHAPTER 5

Running Virtual Scripts

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

- Run virtual scripts from a single workstation.
- Launch and manage multiple script playback.

Understanding Virtual Script Runs

After you generate one or more PeopleSoft Virtual AutoPilot scripts, you are ready to execute playback to simulate multiple users running processes. If you want to simulate multiple users on a single workstation, you can launch the script either from a command line or from Virtual Runner.

Using Mercury Interactive's LoadRunner tool, you can also launch one or more PeopleSoft Virtual AutoPilot script playback sessions on more than one workstation. LoadRunner manages the playback sessions. Using LoadRunner as the script playback manager enables you to more accurately simulate the actual stress that users in a business environment might impose on the system.

Running Virtual Scripts from a Single Workstation

This section provides an overview of virtual script runs from a single workstation, lists a prerequisite, and discusses how to:

- Run virtual scripts from the command line.
- Run virtual scripts using Virtual Runner.

Understanding Virtual Script Runs from a Single Workstation

You can launch the Virtual Script Player from a command line on a single machine or from Virtual Runner, which manages virtual script playback, in order to simulate more than one user on a single workstation. The Virtual Script Player accesses the .vsx file that you create when you generate a virtual script on the Virtual Script Editor. After you run the script, you check the log files for errors.

Prerequisite

Before you can use Virtual Runner, you must cut the vap.ini file from \\B9\system\Bin32 and paste it into \\WINNT.

Running Virtual Scripts from the Command Line

The Virtual Script Player accesses the .vsx file generated by the Virtual Script Editor. You can launch the Virtual Script Player from a command line on a single machine or from a LoadRunner controller when you want to run virtual scripts on more than one workstation.

The command line must have entries specifying the user, the environment, and the script name. This table summarizes the required entries on the command line:

Command Line Abbreviation	Description	Sample Entry
-u	User ID	-u JDE
-p	User password	-p JDE
-e	Environment	-e PRD733
-s	Script Name + number of script iterations to run	-s Script1.vsx

To run virtual scripts from a command line:

1. From the Start menu in Windows, select Command Prompt from the Programs menu.
2. At the C: prompt, type the Virtual Script Player command with appropriate parameters. For example: Virtual Script Player -u JDE -p JDE -e PRD733 -s5 script1.vsx.
3. Press Enter to run the command.
4. To review the progress of the program, press Ctrl-Alt-Del to access the Windows Task Manager.

Note. The Processes tab displays the executable (Virtual Script Player.exe) and the CPU activity associated with it. Otherwise, there is no indication of activity on the screen.

When playback concludes, the Virtual Script Player.exe task disappears from the Task Manager window and a log in the \\PeopleSoft AutoPilot\VAP_Logs directory displays any errors that were encountered. You can change the directory location in the section of vap.ini file.

5. To search the PeopleSoft Virtual AutoPilot log for errors, click the Search menu, select Find, and search on the keyword *error*.

If errors occur, see the documentation on debugging virtual scripts.

6. If the script contains valid values list data, change the data and play the script again.

See Also

[Chapter 3, “Understanding PeopleSoft Virtual AutoPilot Components,” Virtual Script Player Initialization File Parameters, page 19](#)

Running Virtual Scripts Using Virtual Runner

The Virtual Runner program enables you to manage the playback of virtual scripts. You use it to specify the script, the number of player sessions, and the number of iterations that you want to run in each session. You also specify the system environment against which you want to run the sessions.

Z

You use the Virtual Runner log in conjunction with the system logs and more detailed VAP logs to help debug failed sessions.

To run virtual scripts using Virtual Runner:

1. From the desktop or the appropriate directory, double-click the Virtual Runner executable.
2. Click the Option button located on the Virtual Runner toolbar.
The Option window appears.
3. Complete these fields and then click OK:
 - User ID
 - Password
 - Environment
The environment against which you want to run the test.
4. Click the Wizard button.
The Virtual Runner Add Wizard - Step 1 of 3 form appears.
5. Click the Browse for script button to select the script that you want to run.
The Choose a Virtual Player Script to run form appears.
6. Select the script that you want and click the Open button.
The name of the script appears in the Select a script to run field.
7. Specify the number of player sessions to run on the workstation.
Example: enter 4 to run four scripts simultaneously.
8. Specify the number of virtual script session iterations to run.
Example: enter 5 to run the script five iterations sequentially.
9. Click the Next button.
The Virtual Runner Add Wizard - Step 2 of 3 form appears. If you entered information into the Option window, then the Wizard pulls that information into this window.
10. If you did not enter information into the Option window, enter your User ID, Password, and Environment.
11. Click Next.
The Virtual Runner Add Wizard - Step 3 of 3 form appears.
12. If you want to add another script, click the button to add more scripts and repeat steps 4 through 10.
13. If you do not want to add additional scripts, click the Finish button to return to the Virtual Runner form.
Virtual Runner displays the script or scripts that you chose and activates the Run button.
14. Click the Run button to begin script processing.
The main Virtual Runner screen displays the message *Starting Up*, indicating that the processing of the scripts has begun. The main Virtual Runner screen displays the message *Running* when Virtual Runner is processing the script or scripts. If the scripts successfully run, the screen displays the message *Success*.
After Virtual Runner finishes running the sessions, it displays the status of each test. You can view log information about a test by clicking the Log button. If processing is not successful, a red failure message appears.
15. Click Log and click Yes to view the current test log.

The Common Log Viewer form appears.

16. Review the log for error and warning messages.

You can expand the nodes in the Common Log Viewer form to see any error or warning messages that might have been issued during the Virtual Runner session. In addition to error and warning messages, the form displays:

- Name of the test
- Number of errors
- Number of warnings
- Status of the test
- Duration of the test
- Time of completion

17. If the script contains valid values list data, change the data and play the script again.

Launching and Managing Multiple Script Playback

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

- Define scripts.
- Define the host machine.
- Define virtual users.
- Gather LoadRunner results.
- Run virtual playback from the LoadRunner controller.

Understanding LoadRunner

LoadRunner enables you to set up multiple workstations, each representing multiple users, from which you can launch playback sessions to simulate actual user load on the system. You provide LoadRunner with selected rendezvous points and transactions, which LoadRunner then reports to its controller. LoadRunner gathers and stores the results of each run. The LoadRunner controller workstation must have network connection to all of the workstations that are involved in the test, and the controller must run Windows NT.

Defining Scripts

You define the script that you want to play back so that LoadRunner can locate the Virtual Script Player and pass the Player the necessary script command line.

Defining the Host Machine

After you have defined the scripts, you define the host machine for the LoadRunner test and the platform on which the test ran.

Defining Virtual Users

After you define the script and the host machine, you define the virtual users who created the scripts that you want to run. You can define users individually or you can define a group as the virtual user.

Setting Rendezvous Points

You set the rendezvous point that defines for LoadRunner the time at which all virtual scripts pause before the tool releases them for virtual playback.

Gathering LoadRunner Results

The LoadRunner results directory typically has this structure:

- VAPI (the test directory)
- User Name (from those defined in the Users window)
- Session Number Output.txt (the rerouted VAP_log from the client workstation)

Running Virtual Playback from the LoadRunner Controller

After you have prepared virtual script playback, you are ready to run the test from the LoadRunner controller.

CHAPTER 6

Understanding Special Considerations for Simulated Playback

This chapter discusses:

- Playback problem simulation.
- Playback timing.
- Call level.
- Synchronous and asynchronous calls.
- Think times.

Playback Problem Simulation

The PeopleSoft Virtual AutoPilot solves several simulated playback problems. All of the problems in one way or another revolve around the tool's ability to simulate accurately the workings of the PeopleSoft run-time engine. The Virtual Script Editor and the Virtual Script Player work together; the Virtual Script Editor stores key playback information and passes it to the Virtual Script Player, which in turn uses the information to assume the role of the run-time engine. This section explains important simulated playback problems and the ways that PeopleSoft Virtual AutoPilot resolves them.

Playback Timing

This section discusses:

- Event synchronization.
- API playback timing.
- Interthread timing.

Event Synchronization

To accurately simulate system activities, PeopleSoft Virtual AutoPilot must keep script events synchronized during playback. This presents a challenge because PeopleSoft Virtual AutoPilot attempts to simulate multiple users who are stressing the server and the network, while the data upon which PeopleSoft Virtual AutoPilot scripts are based is captured from a single user's script playback. This means that event start times and duration might change significantly during a virtual script playback session.

To meet this challenge, PeopleSoft Virtual AutoPilot must solve two separate problems:

- Manage changes in the duration of individual API calls and the lengths of time between these calls within a single thread. This involves accurately simulating the process time required as the run-time engine handles user load.
- Handle timing differences that might affect interthread dependencies. These interdependencies occur when, for example, an API call in one thread has a data dependency on an API call in another thread.

API Playback Timing

The goal of PeopleSoft Virtual AutoPilot is to accurately simulate the stress that users place on the server and on the network. However, the PeopleSoft AutoPilot script, which contains the data upon which the PeopleSoft Virtual AutoPilot script is built, is not designed to create this stress. The time intervals between events in the PeopleSoft AutoPilot script reflect the running of a single script against the run-time engine. When you run a PeopleSoft Virtual AutoPilot script, the duration of events, and therefore the time intervals between those events, will likely change due to the server and network stress that the script is trying to simulate. The CPU power and memory capability of individual workstations can also affect the playback timing of PeopleSoft Virtual AutoPilot scripts.

PeopleSoft AutoPilot provides the base for the Virtual Script Player to time the execution of events during virtual script playback. When PeopleSoft AutoPilot processes a script, it captures each kernel function call, and it captures the start time and duration of each API call. Therefore, the script contains the gaps of time between each call, which occur as the system carries out other processes. The API calls within a thread might be represented as blocks of time of various lengths with intervening spatial gaps that symbolize the time duration between each call.

Preserving these chronological gaps during data capture provides the basis for simulating playback by many users, a situation that is likely to increase the length of time that is required to execute the same API calls.

For example, suppose that virtual playback on a single workstation simulates 10 users using a server that is not as powerful as the server that was in use when the PeopleSoft AutoPilot script playback session originally occurred. In this scenario, the duration of API call is likely to lengthen, which could cause one API call to overlap another, halting playback.

However, since the event stream has preserved the *intervals* between each call, virtual playback can proceed, regardless of the duration of any or all of the calls within a thread.

PeopleSoft AutoPilot's ability to record the duration and length of time between API calls is also important because it can accurately determine the number of virtual users who can be simulated on a single workstation. For example, lengthy API calls might indicate an underpowered server, a workstation lacking the CPU and memory capability to handle the number of virtual user sessions that you desire, or an application bug. In each of these instances, you would likely scale back the number of users you want to simulate in a PeopleSoft Virtual AutoPilot playback session.

Interthread Timing

The Virtual Script Editor also plays a role in handling script playback timing so that PeopleSoft Virtual AutoPilot can simulate a stressed environment. The run-time engine might, for example, create a thread that contains an API call with a data dependency on another call, which might, in turn, exist in a separate, asynchronously running thread.

In a stressed environment, however, the duration of API calls might lengthen unpredictably. This might result in a data-dependent API call in one thread starting before the API upon which it depends has finished.

To deal with this potential problem, the Virtual Script Editor notes the data dependency when you forge value links between two API calls and preserves the timing interval between the calls.

When you run the virtual script, the Virtual Script Player increases the interval between APIs in one thread so that an API in one thread has time to complete before a data-dependent API in a thread running asynchronously to it is called.

In this way, PeopleSoft Virtual AutoPilot preserves the necessary time interval that existed between the data-dependent calls when you originally ran the script.

Clearly, the Virtual Script Player, in this scenario, manipulates the time interval between API calls in the first thread. However, the manipulation represents an attempt to fairly simulate what the system does in reality. The run-time engine manages data-dependent APIs so that they can run without breaking the system. It is, therefore, appropriate that the Virtual Script Player, in assuming the role of the run-time engine, simulate the run-time engine's responsibility—for example, the delay of one APIs completion based on its logical relationship to another API.

Call Level

Some API calls invoke other API calls automatically within the same thread. Call level refers to an API call's position in the sequence of calls. For example, an EditLine business function might invoke a JDB Fetch call for a company number. In this example, the call level of the EditLine business function is 1, the call level of the JDB Fetch call is 2, and the PeopleSoft AutoPilot event stream records the two separate API calls.

However, while the run-time engine handles two separate API calls in this example, the processing occurs seamlessly: the second call follows immediately from the first without additional input from the user. For this reason, a PeopleSoft Virtual AutoPilot script contains only those API calls with a depth of 1. The Virtual Script Player automatically handles any API calls invoked by the original call, just as the run-time engine would.

This PeopleSoft Virtual AutoPilot capability is important for playing scripts back in batch mode. If APIs with a call level greater than 1 were treated separately, repetitive processing would occur. Such repetition would not correctly simulate system processing.

The Virtual Script Editor provides a convenient way for you to view call level in the event stream. Each line that displays an initiating API call shows a call level of 1. Any calls that are invoked by the initiating call show a level of 2 or greater.

Note. The Virtual Script Editor displays API calls with a call level greater than 1 only if you select the Capture Performance Statistics option when you configure playback in PeopleSoft AutoPilot. If you do not wish to view call levels greater than 1, select the Capture Virtual Script event stream option.

If you click the detail line of an API call that has a call level of 2 or greater, the event stream detail pane displays no parameters, meaning that you cannot value link any API call with a call level greater than 1. Therefore, no API calls with a call level greater than 1 appear in the Target Parameters pane.

Synchronous and Asynchronous Calls

As part of simulating PeopleSoft software operations, PeopleSoft Virtual AutoPilot must be able to manage synchronous and asynchronous API calls, an important management responsibility of the runtime engine. This ability ties into the Virtual Script Player's management of threads because an asynchronous call generates a separate thread.

A typical example of synchronous and asynchronous API call generation occurs when you enter data in a sales order line. You generate a synchronous call for each line edit; that is, the CallObject API for line 1 in a PeopleSoft grid precedes the CallObject API for line 2, and the CallObject API for line 2 does not occur until you have completed line 1. However, when you reach the end of a line, press the tab button, and proceed to line 2, you also generate an asynchronous API call that includes the data structure for the line that you just completed. The asynchronous CallObject API validates the data that you entered in line 1 through a series of related API calls. Meanwhile, you move ahead and begin entering sales order data in line 2.

The runtime engine manages this situation by generating a new thread for asynchronous calls and sending these calls to a queue to manage on a first-in, first-out (FIFO) basis. For example, you might enter 20 lines to the sales order entry grid. As you reach the end of each line and tab, the system will likely generate a new asynchronous call. Therefore, a number of asynchronous calls might queue for managing. When the runtime engine finishes processing the asynchronous calls, it stops the thread.

PeopleSoft Virtual AutoPilot manages the simulation of asynchronous call management through the operation of each part of its architecture. The PeopleSoft AutoPilot and PeopleSoft software hooks capture the timings of the synchronous and asynchronous calls that script playback generates. The Virtual Script Editor preserves the thread identifiers produced during playback, and the Virtual Script Player generates thread synchronization events in the PeopleSoft Virtual AutoPilot script based on the temporal relationships among events in the captured event stream.

The Virtual Script Player also manages the threads generated during virtual playback. When virtual playback yields an asynchronous call, the Virtual Script Player queues the calls in a new thread and manages them on the same FIFO terms that the runtime engine uses, thereby managing interthread synchronization as well as event timing within threads.

Synchronous and asynchronous call management provides another example of the Virtual User Tool's ability to accurately simulate the system, thereby providing you with a realistic picture of network and server stress.

See Also

[Chapter 6, "Understanding Special Considerations for Simulated Playback," Playback Problem Simulation, page 43](#)

Think Times

You insert wait periods while writing an PeopleSoft AutoPilot script in an attempt to accurately simulate the way people use PeopleSoft software. You click the Wait Before Proceeding button and insert pauses into the playback. These pauses are in millisecond increments.

One possible reason for inserting wait periods in the script is to simulate the pauses that might occur as a user enters vouchers. A user might pause to answer the phone or tend to other tasks, and then return to making the entries.

The event stream generated during PeopleSoft AutoPilot script playback records these wait times. They do not appear, however, with a label in the event stream pane if you import the event stream into the Virtual Script Editor. Rather, you recognize them by noting in the event stream pane the duration of time between the end of one event and the beginning of the next.

The PeopleSoft Virtual AutoPilot script that you create contains the waits inserted in the original script, and the Virtual Script Player manages the delays during script playback. The inclusion of think times provides another element that helps PeopleSoft Virtual AutoPilot simulate the PeopleSoft environment, which includes many users performing different tasks under a variety of circumstances.

You might want to analyze the event stream to see the length of time the events in the PeopleSoft AutoPilot script took to complete. Think times that you insert into the script do not interfere with event duration analysis because the Virtual Script Editor's event graph does not reflect any wait times. If, for example, a five-second wait occurs between a CallObject API and an OpenTable API, the event graph displays only the amount of time required to run the APIs. Thus, you get a true picture of the time that the system required to process the API calls.

CHAPTER 7

Troubleshooting PeopleSoft Virtual AutoPilot

This chapter provides an overview of PeopleSoft Virtual AutoPilot troubleshooting and discusses how to:

- Review the PeopleSoft Virtual AutoPilot log file to locate errors.
- Identify environment problems.
- Debug PeopleSoft Virtual AutoPilot scripts.

Understanding PeopleSoft Virtual AutoPilot Troubleshooting

You might encounter a script failure when you play back your PeopleSoft Virtual AutoPilot script. Troubleshooting PeopleSoft Virtual AutoPilot scripts consists of these tasks.

1. Locate the source of the script failure, which might be in either PeopleSoft Virtual AutoPilot or the system.
2. Run through a short list of script debugging techniques.

These techniques correct errors in business function and database API calls, transaction timing, and multiple playback sessions. You might also need to debug the system. In some cases, the problem lies in the original PeopleSoft AutoPilot script or in application source code.

3. Review the PeopleSoft AutoPilot script if you created it without first validating it through replay.

You cannot trace all failures of PeopleSoft Virtual AutoPilot scripts to a single source, nor can you debug all scripts using a single method. In learning tips and techniques for troubleshooting PeopleSoft Virtual AutoPilot scripts, you also learn the best solution to apply to a particular problem.

Reviewing the PeopleSoft Virtual AutoPilot Log File to Locate Errors

This section provides an overview of the PeopleSoft Virtual AutoPilot log file and discusses how to:

- Find error entries in the PeopleSoft Virtual AutoPilot log file.
- Locate the log file in the event of early script failure.
- Set the message parameter.

Understanding the PeopleSoft Virtual AutoPilot Log File

The vap.log file contains messages about each PeopleSoft Virtual AutoPilot script that you run. Therefore, it is the primary source of information about errors that might cause the script to fail.

The Virtual Script Player sends an error message to the log file when a PeopleSoft Virtual AutoPilot script fails during processing. If you launched the script from LoadRunner or from Virtual Runner, script failure halts the playback process, sending an instant signal that an error has occurred. However, if you launched the script from the DOS command line, you will not receive an error message.

The message level parameter controls the kind and number of error messages that you receive in the vap.log file when you play back a PeopleSoft Virtual AutoPilot script. You set the message level in the PeopleSoft Virtual AutoPilot initialization file. You should generally set the message parameter at 0, 1, 3, or 7 to minimize the number of messages that you collect. Setting the parameter higher causes slower playback performance and at least potentially skew playback results, thereby making performance analysis difficult. However, when you are attempting to find the source of a script failure, increasing the message level parameter temporarily can help you diagnose the problem. You might find very few messages in the log file as a result of setting the debug parameter too low. For example, if you set the parameter to 0 (zero), you will receive only a minimal number of messages.

If you fail to find the source of the script failure in Virtual User Tool, you can use several procedures to troubleshoot the system.

See Also

[Chapter 3, “Understanding PeopleSoft Virtual AutoPilot Components,” Virtual Script Player Initialization File Parameters, page 19](#)

Finding Error Entries in the PeopleSoft Virtual AutoPilot Log File

To isolate an error, go to the log file, which contains the test results, select a test, open the text file, and search on the keyword *error*.

To find entries in the PeopleSoft Virtual AutoPilot log file:

1. Locate and open the vap.ini file.
2. In the vap.ini file, go to the [PATHS] section to determine the location of the LogDirectory.
3. Follow the path to the LogDirectory.
4. Open the text file for the failed script using Notepad.
5. From the Edit menu, select Find.
6. On the Find form, complete the *Find what* field with the word error.
7. Note the line and event in which the error occurred.
8. Click Find Next to go to the next error.

See Also

[Chapter 3, “Understanding PeopleSoft Virtual AutoPilot Components,” Virtual Script Player Initialization File Parameters, page 19](#)

Locating the Log File in the Event of Early Script Failure

The Virtual Script Player reads the location of the log directory out of the vap.ini text file. However, the script might fail before the Virtual Script Player has a chance to read the location. Therefore, when you go to the location of the log file that you specified as an initialization parameter, you will not find the test log. Despite the early failure, PeopleSoft Virtual AutoPilot did log the errors.

To locate the PeopleSoft Virtual AutoPilot log file in the event of early script failure:

1. Locate and open the vap.ini file.
2. In the vap.ini file, determine the location of the LogDirectory.
3. Follow the path to the LogDirectory.
4. If the log that you are looking for is not in its usual location, go to the root of the drive and look for the log.

Note. If you do not find a log file in either location, you must examine the PeopleSoft Virtual AutoPilot setup. Make sure that PeopleSoft Virtual AutoPilot is installed completely and correctly.

Setting the MessageLevel Parameter

To set the MessageLevel parameter:

1. Locate and open the vap.ini file.
2. In the vap.ini file, find the [Log] entry.
3. If the MessageLevel parameter is set lower than you want, change the setting.
4. Save the change and close the vap.ini file.

Note. If the Virtual Script Player crashes while you are running a script, you might find very few messages in the log file. This occurs because the Virtual Script Player did not flush the log file buffer, in which messages are stored, before the crash. You can prevent this by setting the message level parameter at 31. This parameter requires that the Virtual Script Player flush the log file buffer after each message. Remember, however, that system performance decreases when you set the message level at 31, so you should not leave it at that level permanently.

Identifying Environment Problems

This section provides an overview of environment problems and discusses how to:

- Diagnose environment problems.
- Investigate system errors.

Understanding Environment Problems

If the PeopleSoft Virtual AutoPilot script fails very early, even before the system completes its initial system logon, you might not be initializing the environment. In this case, you can troubleshoot system operations rather than PeopleSoft Virtual AutoPilot operations. For example, you can try to log on to Explorer and run it through several sample tasks, such as opening an application. Use the same user ID, password, and environment name when you log onto the system that you used when logging on to the Virtual Script Player. You also can troubleshoot system errors, as these might also prevent you from replaying the PeopleSoft Virtual AutoPilot script. If you have cleared any problems that might exist in running Explorer, try running the PeopleSoft Virtual AutoPilot script again.

Diagnosing Environment Problems

Because Virtual User Tool's primary task is to simulate system operations, it must be able to initialize an environment at script playback time. For this to happen, the system itself must be initializing correctly. To exclude the Virtual Script Player as the source of script failure, you might attempt to sign on to the system to make sure that it is opening and running correctly.

To diagnose an environment problem:

1. Close the Virtual Script Player.
2. Sign on to Explorer.
3. Perform several operations, such as accessing an application, changing forms, adding data, and so on.
4. If you are certain that the system is running correctly, rerun the script.

Note. Be sure to use the same user ID, password, and environment that you use when you log on to the Virtual Script Player.

Investigating System Errors

Even if the system is initializing correctly, you might find errors that occur when you attempt to enter or edit data in an application. To isolate errors that occur in the system, you can select debugging and attempt to correct the errors.

To investigate system errors:

1. Click the Windows Start menu and select Run.
2. In the Open control of the Run form, type JDE.INI.
The JDE.INI file appears.
3. In the JDE.INI file, go to the [DEBUG] section.
4. Enter the Output parameter as File.
5. Run an PeopleSoft AutoPilot script or access the system and run applications that are failing in the PeopleSoft Virtual AutoPilot script.

Note. If error messages display in the status bar, click the stop sign. Read the error messages that appear. You can right-click error messages to display more troubleshooting information about each one.

6. Open the jdedebug.log file to evaluate any errors that occur.

Important! Change the Output parameter in the [DEBUG] section of the JDE.INI file to NONE after you have corrected errors that prevent the PeopleSoft Virtual AutoPilot script from functioning correctly.

Debugging PeopleSoft Virtual AutoPilot Scripts

This section provides an overview of PeopleSoft Virtual AutoPilot debugging and discusses how to:

- Display business function parameters.
- Diagnose business function failures in PeopleSoft Explorer.
- Research value-linking errors in Virtual Script Editor.
- Verify that value linking is functioning.
- Identify and correct variable value-linking errors.
- Verify the validity of PeopleSoft Virtual AutoPilot script data.
- Identify and correct duplicate key errors.

- Rectify irregular transaction times.
- Prevent multiple script playback problems.
- Correct uninitialized user handle errors.

Understanding PeopleSoft Virtual AutoPilot Debugging

If you have been troubleshooting problems with PeopleSoft Virtual AutoPilot script playback but are still having trouble running the scripts, a business function call is likely causing the failure. You can review the log file to locate the source of the error, and you can identify the particular business function call that failed. You should have the message level in the vap.ini file set at 15 so that the log file displays parameter values.

You might also encounter problems that complicate your performance characterization efforts. For example, transaction information that you incorrectly or incompletely enter in an PeopleSoft AutoPilot script might cause irregular transaction times in the PeopleSoft Virtual AutoPilot script, thus making it difficult to draw accurate conclusions about system performance. In this case, you should troubleshoot the PeopleSoft AutoPilot script, making sure that you have completely and accurately scripted input commands. If you modify the PeopleSoft AutoPilot script, remember to run it again, capture the playback results, and reimport the event stream into the Virtual Script Editor.

PeopleSoft Virtual AutoPilot also enables you to play back a script multiple times in succession, another important feature for performance characterization. However, doing so might cause playback to lock, again defeating your efforts to draw clearly and confidently characterize system performance. In this event, check the disk space to make sure you have enough to handle the testing.

If you have exhausted all of the debugging possibilities discussed here, you must turn your attention to debugging Explorer. Remember that if the same errors that appear in the PeopleSoft Virtual AutoPilot script appear when you run the application in PeopleSoft EnterpriseOne, you likely have a system problem that you must debug. Your debugging efforts might include a call to PeopleSoft System Support.

You can gain additional insight into potential system problems by double-clicking the stop sign that appears in the status bar of a PeopleSoft form when an error occurs. When you perform this action, the system displays explanatory text, including possible causes and solutions, that helps you diagnose the source of the error. You can get additional troubleshooting information by setting the Output parameter in the JDE.INI file to FILE. Remember that doing so will degrade system performance, so you should return the Output parameter in the JDE.INI file to NONE after you have diagnosed and corrected any problems with the script.

For a PeopleSoft Virtual AutoPilot script to run correctly, you must value link all required target parameters to the appropriate source parameters using the Virtual Script Editor. Failing to do so, or forging value links improperly, could cause the script to fail.

Displaying Business Function Parameters

Displaying the business function call parameters helps you to debug the PeopleSoft Virtual AutoPilot scripts. To do so, you set the MessageLevel parameter in the vap.ini file at 15 or at 31. At this level, the log file displays all the input and output parameter values of:

- Business function API calls in the script
- Text of any error messages
- File name of the business function
- Line number in the source code that contains the error

To display business function parameters:

1. Locate and open the vap.ini file.
2. In the vap.ini file, find the [Log] entry.
3. Set the Message Level parameter at 15 or 31.
4. Save the change and close the vap.ini file.

Important! Remember that you should not set the MessageLevel parameter permanently at 31 as this will cause performance to degrade. Leaving the MessageLevel parameter at 15 does not significantly degrade performance, but it can cause many messages and a great deal of text to accumulate in the log file. You should not leave the message level permanently set at 15, as doing so could consume a significant amount of disk space.

Diagnosing Business Function Failures in PeopleSoft Explorer

Your scripts must run properly in Explorer before PeopleSoft Virtual AutoPilot can run them properly. Therefore, you should determine early whether business function API calls are failing in the system when you run an PeopleSoft AutoPilot script. To do so, you can select PeopleSoft debugging in the JDE.INI file.

Note. When you run an application, right-click and select View System Log to view the jdedebug.log.

You might set breakpoints in the PeopleSoft AutoPilot script after commands that initialize a business function API call, which will enable you to check the jdedebug.log at these key points.

By verifying the system's ability to process the commands in the PeopleSoft AutoPilot script, you either pinpoint or exclude the system as a source of script failure. If it is causing the script failure, you work on debugging the system; conversely, if the business functions process properly when you run the script in the system, concentrate on finding the source of the script failure in the Virtual User Tool.

To diagnose business function failures:

1. Locate and open the JDE.INI file.
2. In the JDE.INI file, go to the [DEBUG] section and set the output parameter to FILE.
3. In the PeopleSoft AutoPilot script pane, right-click a command line that follows a command that runs a business function (optional).
4. Click Toggle Breakpoint (optional).
5. Play back the PeopleSoft AutoPilot script, either to the end or to a designated breakpoint.
6. Right-click inside a PeopleSoft form.
7. Select View System Log.
8. Click File.
9. In the drop-down menu, select c:\jdedebug.log.
10. Troubleshoot the jdedebug.log file, searching for business functions.

Researching Value-Linking Errors in the Virtual Script Editor

A business function API call might fail when you run the PeopleSoft Virtual AutoPilot script because you incompletely value linked the business function parameters in the event stream to the parameters in the PeopleSoft Virtual AutoPilot script while you were working in the Virtual Script Editor.

To research value linking errors in the Virtual Script Editor:

1. Locate and open the `vap.ini` file.
2. Set the `MessageLevel` parameter at 15.
3. Run the PeopleSoft Virtual AutoPilot script.
4. In the `vap.log` file, search for business function errors.
5. In the Virtual Script Editor, verify the value linking.

Remember that you must value link any parameters that do not use constant values during script playback. If you do not value link these parameters, the script fails because, typically, the script playback creates duplicate keys.

You are required to provide value links for these parameters:

- Job number
- Document number
- Batch number
- Any parameter that uses a value from a valid values list

These parameters might frequently require value linking:

- Computer identification
 - Those that require dates
6. Perform any necessary value linking in the Virtual Script Editor.
 7. Rerun the script with the `MessageLevel` parameter in the [LOG] section of the `vap.ini` file set at 15.
This setting enables you to capture parameter values and value substitutions in the log file.
 8. Recheck the `vap.log` file and look for business function API errors.

Verifying That Value Linking Is Functioning

You can verify that PeopleSoft Virtual AutoPilot is linking parameter values by creating valid values lists in the PeopleSoft AutoPilot script. In the Virtual Script Editor, you value link any parameters that use values from the valid values list. The Virtual Script Player should link the values in the valid values lists to the appropriate parameters in the PeopleSoft Virtual AutoPilot script during PeopleSoft Virtual AutoPilot script playback.

To verify that PeopleSoft Virtual AutoPilot performs the value linking, you can set the `MessageLevel` parameter at 15 and run the PeopleSoft Virtual AutoPilot script. After you run the script, you search the log file for valid values list data, identify that data, and change the data in the `.atd` file, which stores the valid values list data.

To verify that value linking is functioning:

1. Locate and open the `vap.ini` file.
2. In the `vap.ini` file, set the `MessageLevel` parameter at 15.
3. Run the PeopleSoft Virtual AutoPilot script.
4. Review the log file for valid values list data.

Note. You can search for valid values list data using the `.atd` extension. Verify that the values you expect are present and look for any error messages associated with the data.

5. In the `c:\.atd` file, change the valid values list data.

6. In the Virtual Script Editor, make sure that you have value-linked all of the new data in the valid values list to the correct parameters in the PeopleSoft Virtual AutoPilot script.
7. Rerun the PeopleSoft Virtual AutoPilot script.
When you replay the script, PeopleSoft Virtual AutoPilot should use the new data from the valid values list.
8. Review the log file for old valid values list data, to make sure that the Virtual Script Player used the new data rather than any of the old values.
9. If you find any of the old valid values list data, review the value linking in the Virtual Script Editor.

Identifying and Correcting Variable Value-Linking Errors

Another type of value linking related error occurs if you declare a value in a PeopleSoft AutoPilot script but do not set its value. In this case, if you value link the variable, the Virtual Script Editor registers errors in the script log during the virtual script generation process. To correct the errors, you must modify the PeopleSoft AutoPilot script by setting the value of the variable.

To identify and correct variable value-linking errors:

1. From the desktop or the appropriate directory, double-click the Virtual Script Editor executable.
2. Select a test and click the Generate button.
3. Review the Virtual Player Script Log form for validation error messages.
4. If validation error messages appear in the Virtual Script Log form, reopen the PeopleSoft AutoPilot script.
5. Set a value for any declared variables that do not have a value in the Virtual Script Editor.
6. Save and rerun the PeopleSoft AutoPilot script.

Important! Be sure that playback remains configured to capture the virtual script event stream.

7. Re-import the event stream into the Virtual Script Editor and regenerate the PeopleSoft Virtual AutoPilot script.

Verifying the Validity of PeopleSoft Virtual AutoPilot Script Data

Business function errors that occur in the PeopleSoft Virtual AutoPilot script might be caused by data errors. Data errors occur because the environment against which you wrote the PeopleSoft AutoPilot script differs from the environment against which you attempt to play back the PeopleSoft Virtual AutoPilot script.

To verify the validity of data in the PeopleSoft Virtual AutoPilot script:

1. Locate and open the vap.ini file.
2. In the vap.ini file, set the MessageLevel parameter to 15.
3. Run the PeopleSoft Virtual AutoPilot script.
4. Search the log file for business function errors.
5. Verify that the environment against which you wrote the PeopleSoft AutoPilot script and against which you ran the PeopleSoft Virtual AutoPilot script is the same.
6. If the two environments are different, recreate the valid values lists so that they contain values that are valid for the environment against which you are running the PeopleSoft Virtual AutoPilot script.

Note. You can also replay the PeopleSoft AutoPilot script in the same environment against which you are running the PeopleSoft Virtual AutoPilot script. In that case, follow the next two steps.

7. Re-import the event stream into the Virtual Script Editor.
8. Regenerate a PeopleSoft Virtual AutoPilot script by forging value links between the source and target parameters.

Identifying and Correcting Duplicate Key Errors

JDB Insert and Update API calls might fail in the PeopleSoft Virtual AutoPilot script because of duplicate key errors. These errors occur when you attempt to enter two records with the same value into a key column.

The duplicate key error prevents you from doing this. Failure to value link all the necessary parameters in the PeopleSoft Virtual AutoPilot script could cause duplicate key errors. You can view updated and inserted JDB API parameter values in the Virtual Script Editor.

Note. Duplicate keys could also result from an application error.

To identify and correct duplicate key errors:

1. Locate and open the JDE.INI file.
2. In the JDE.INI file, go to the [DEBUG] section.
3. Change the Output parameter to FILE.
4. Play the PeopleSoft AutoPilot script.
5. Locate and open the jdedebug.log file.

If you have duplicate key errors, you will find them in the jdedebug.log file.

6. Open the script in the Virtual Script Editor.
7. Check value linking for all JDB Insert and Update API calls.
8. When you are sure that you have value linked all JDB Insert and Update API calls, rerun the script.
9. If you continue to get duplicate key errors, review the application for errors that might be causing the problem.

Rectifying Irregular Transaction Times

You measure transaction times by choosing events as start and endpoints in the PeopleSoft AutoPilot script. For example, you might launch an application, move from one form to another by clicking the Add button, and then make entries to several header controls and grid columns in an active form before closing that form.

You might label that entire sequence of commands, from launching the application to closing the form, as a transaction. To see how efficiently the system manages this transaction, you label launching the application as the start of the transaction and closing the form as the end of the transaction. You also apply a name to the transaction and attach that name to the start and to the end. You use the Wait/Comment command in PeopleSoft AutoPilot to insert the start and end of the transaction into the script and to apply a name to the transaction.

If you do not include both a start and an end time for the transaction, you might find irregular or inexplicable transaction times in the log, or you might find that the transaction fails. Failing to ensure that the name that you applied to the start of the transaction matches precisely the name that you applied to the end of the transaction, including capital letters and any special characters, might also cause irregular transaction times or transaction failures.

Important! PeopleSoft Virtual AutoPilot transaction timing accuracy has several limitations that make broad-based performance characterization assertions impossible. Accurate timings can be achieved only on a discrete workstation, while PeopleSoft Virtual AutoPilot simulates server load.

To rectify irregular transaction times:

1. From the desktop or the appropriate directory, double-click the PeopleSoft AutoPilot executable.
2. In the script pane of the PeopleSoft AutoPilot form, place the insertion cursor directly above the command line that represents the start of the transaction.
3. In the menu bar, click Command and then select Wait/Comment.
4. In the uncompleted Comment list of the PeopleSoft AutoPilot command pane, enter Start, a space, and a name for the transaction.

5. Click the Insert button.

PeopleSoft AutoPilot inserts a command line marking the start of the transaction.

6. Place the insertion cursor after the command line that represents the end of the transaction.
7. In the PeopleSoft AutoPilot menu bar, click Command and then select Wait/Comment.
8. In the Comment list of the PeopleSoft AutoPilot command pane, enter End, a space, and a name for the transaction.

PeopleSoft AutoPilot inserts a command line marking the end of the transaction.

Important! 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.

9. Click the Insert button.

Preventing Multiple Script Playback Problems

PeopleSoft Virtual AutoPilot enables you to play back the same script in consecutive sessions or to simulate multiple users playing back scripts simultaneously. In either case, you must make sure that you have sufficient disk space to handle the load created by PeopleSoft Virtual AutoPilot script playback, particularly if you plan to run a long test involving many playback iterations or simulation of a large number of users. If you do not have sufficient disk space, you might find that PeopleSoft Virtual AutoPilot script playback locks up after only a few playbacks.

Debugging Virtual Runner

If Virtual Runner fails immediately after you click the Run button, first check the vapplayer.exe path specified in the vap.ini file. The vap.ini [COMMAND] section binname parameter specifies the full path of the VAPPlayer.exe file.

Virtual Script Player should operate the same whether you run under Virtual Runner control or from a command line. If it does not, you might be running two different copies of the vapplayer.exe. This might occur if the vap.ini [COMMAND] binname parameter is pointing to an old version of the Virtual Script Player. Make sure that binname parameter points to the correct drive and directory, and that you discard any old versions of the Virtual Script Player that you might have on the workstation.

If you set Virtual Script Player to run a virtual script multiple times in succession, and the script only runs a few times before locking up, you should review the available disk space. If you have set the JDE.INI error logging settings at a high level, the jde.log and jddebug.log can fill a disk very quickly. Make sure that enough free space is available on all relevant disk drives before you start a long test.

Debugging LoadRunner

If you have set the JDE.INI or vap.ini error logging settings at a high level, and you run many virtual user sessions, the network might become saturated, communications between LoadRunner controller and the host machines might become scrambled, or both. You can address this problem by setting the MessageLevel parameter in the vap.ini files on all machines lower. This will decrease the volume of log file traffic.

This table summarizes steps that you can take to minimize PeopleSoft Virtual AutoPilot script playback problems:

Situation Affecting Playback	Possible Solution
jde.log and jddebug.log messages fill up disk quickly during PeopleSoft Virtual AutoPilot script playback	In [DEBUG] section of the JDE.INI file, set Output parameter to NONE
PeopleSoft Virtual AutoPilot log file fills with messages, consuming disk space	In [LOG] section of the vap.ini file, set MessageLevel parameter to 0, 1, 3, or 7

Correcting Uninitialized User Handle Errors

An error labeled *Uninitialized User Handle* might cause the PeopleSoft Virtual AutoPilot script to fail. This error occurs when you attempt to create a PeopleSoft Virtual AutoPilot script using playback results that you obtained from the first run of a PeopleSoft application when just-in-time installation occurs, or when you have system debugging turned on when you capture the results of PeopleSoft AutoPilot script playback.

To correct uninitialized user handle errors:

1. From the desktop or the appropriate directory, double-click the Virtual User Tool executable.
2. In Virtual User Tool, discard the results of the script generation attempt that failed.
3. In PeopleSoft AutoPilot, rerun the script in the same environment that you created it.
4. Use the new results data to generate a new PeopleSoft Virtual AutoPilot script.

Glossary of PeopleSoft Terms

activity	A scheduling entity in PeopleSoft EnterpriseOne Form Design Aid that represents a designated amount of time on a calendar.
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 PeopleSoft EnterpriseOne tool that can be used to extract, transform, and load enterprise data. APAg supports access to data sources in the form of relational databases, flat file format, and other data or message encoding, such as XML.
application server	A server in a local area network that contains applications shared by network clients.
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.
alternate currency	<p>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.</p> <p>In PeopleSoft 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.</p>
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 PeopleSoft EnterpriseOne reports as of a specific date to determine balances and amounts of accounts, units, and so on as of that date.
back-to-back process	A process in PeopleSoft EnterpriseOne Workflow Management that contains the same keys that are used in another process.
batch processing	<p>A process of transferring records from a third-party system to PeopleSoft EnterpriseOne.</p> <p>In PeopleSoft EnterpriseOne Financial Management, batch processing enables you to transfer invoices and vouchers that are entered in a system other than EnterpriseOne to PeopleSoft EnterpriseOne Accounts Receivable and PeopleSoft EnterpriseOne Accounts Payable, respectively. In addition, you can transfer address book information, including customer and supplier records, to PeopleSoft EnterpriseOne.</p>
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	<p>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.</p> <p>See also direct connect and store-and-forward.</p>
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-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 view	A means for selecting specific columns from one or more PeopleSoft EnterpriseOne 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 PeopleSoft 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 PeopleSoft EnterpriseOne that appear on forms in the software.
connector	Component-based interoperability model that enables third-party applications and PeopleSoft EnterpriseOne to share logic and data. The PeopleSoft EnterpriseOne connector architecture includes Java and COM connectors.
contra/clearing account	A general ledger account in PeopleSoft 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 PeopleSoft EnterpriseOne General Accounting.
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.
cost assignment	The process in PeopleSoft EnterpriseOne Advanced Cost Accounting of tracing or allocating resources to activities or cost objects.
cost component	In PeopleSoft EnterpriseOne Manufacturing Management, an element of an item's cost (for example, material, labor, or overhead).
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.
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 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 PeopleSoft 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.
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 PeopleSoft 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 PeopleSoft 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 PeopleSoft EnterpriseOne or PeopleSoft World.
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 process	A software process that enables PeopleSoft 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. PeopleSoft 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.
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.
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 <i>business unit</i> .
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 user 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.
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.
interface table	See Z table.
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.
interoperability model	A method for third-party systems to connect to or access PeopleSoft EnterpriseOne.
in-your-face-error	In PeopleSoft EnterpriseOne, a form-level property which, when enabled, causes the text of application errors to appear on the form.
IServer service	Developed by PeopleSoft, 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 PeopleSoft EnterpriseOne or People World displays 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 PeopleSoft 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 PeopleSoft file (or member for iSeries) that provides the runtime settings required for EnterpriseOne initialization. Specific versions of the file or member must reside on every machine running PeopleSoft 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 PeopleSoft 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 PeopleSoft EnterpriseOne.
JDENET	PeopleSoft 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 PeopleSoft EnterpriseOne supported platforms.
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 PeopleSoft EnterpriseOne and World software runs.
MailMerge Workbench	An application that merges Microsoft Word 6.0 (or higher) word-processing documents with PeopleSoft EnterpriseOne records to automatically print business documents. You can use MailMerge Workbench to print documents, such as form letters about verification of employment.
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 PeopleSoft 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 PeopleSoft 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 PeopleSoft 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.
named event rule (NER)	Encapsulated, reusable business logic created using event rules, rather than 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.
<i>nota fiscal</i>	In Brazil, a legal document that must accompany all commercial transactions for tax purposes and that must contain information required by tax regulations.
<i>nota fiscal factura</i>	In Brazil, a nota fiscal with invoice information. See also <i>nota fiscal</i> .

Object Configuration Manager (OCM)	In PeopleSoft 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 PeopleSoft 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 PeopleSoft EnterpriseOne data for summarization and report generation.
Output Stream Access (OSA)	An interoperability model that enables you to set up an interface for PeopleSoft EnterpriseOne to pass data to another software package, such as Microsoft Excel, for processing.
package	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 snap shot 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 PeopleSoft 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.
PeopleSoft Database	See JDEBASE Database Middleware.
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 PeopleSoft EnterpriseOne environment used to test unaltered objects with PeopleSoft 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 PeopleSoft 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.
program temporary fix (PTF)	A representation of changes to PeopleSoft software that your organization receives on magnetic tapes or disks.
project	In PeopleSoft EnterpriseOne, a virtual container for objects being developed in Object Management Workbench.
promotion path	<p>The designated path for advancing objects or projects in a workflow. The following is the normal promotion cycle (path):</p> <p>11>21>26>28>38>01</p> <p>In this path, <i>11</i> equals new project pending review, <i>21</i> equals programming, <i>26</i> equals QA test/review, <i>28</i> equals QA test/review complete, <i>38</i> equals in production, <i>01</i> 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.</p>
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 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 PeopleSoft methods for replicating data to individual workstations. Such machines are set up as pull subscribers using PeopleSoft 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 PeopleSoft EnterpriseOne, the QBE line is the top line on a detail area that is used for filtering data.
real-time event	A service that uses system calls to capture PeopleSoft EnterpriseOne transactions as they occur and to provide notification to third-party software, end users, and other PeopleSoft systems that have requested notification when certain transactions occur.
refresh	A function used to modify PeopleSoft 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.
quote order	In PeopleSoft EnterpriseOne Procurement and Subcontract Management, a request from a supplier for item and price information from which you can create a purchase order.

	In PeopleSoft EnterpriseOne Sales Order Management, item and price information for a customer who has not yet committed to a sales order.
selection	Found on PeopleSoft 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.
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. It also updates the Server Plan detail record to reflect completion.
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 PeopleSoft 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.
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.
supplemental data	<p>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.</p> <p>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 PeopleSoft EnterpriseOne systems.</p>
table access management (TAM)	The PeopleSoft 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 PeopleSoft EnterpriseOne and third-party systems using non-PeopleSoft EnterpriseOne tables.
table conversion	An interoperability model that enables the exchange of information between PeopleSoft EnterpriseOne and third-party systems using non-PeopleSoft 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 PeopleSoft 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 PeopleSoft EnterpriseOne.
three-way voucher match	In PeopleSoft EnterpriseOne 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 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 voucher match	In PeopleSoft EnterpriseOne 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 Overrides merge	Adds new user override records into a customer's user override table.
variance	<p>In Capital Asset Management, the difference between revenue generated by a piece of equipment and costs incurred by the equipment.</p> <p>In EnterpriseOne Project Costing and EnterpriseOne Manufacturing Management, 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.</p>
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 PeopleSoft EnterpriseOne or World 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.
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.

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 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 PeopleSoft 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 EnterpriseOne Manufacturing Management, 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 PeopleSoft EnterpriseOne transactions as they occur and then calls third-party software, end users, and other PeopleSoft 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 PeopleSoft EnterpriseOne for responses.
XML List	An interoperability capability that enables you to request and receive PeopleSoft EnterpriseOne database information in chunks.
XML Service	An interoperability capability that enables you to request events from one PeopleSoft EnterpriseOne system and receive a response from another PeopleSoft EnterpriseOne system.
XML Transaction	An interoperability capability that enables you to use a predefined transaction type to send information to or request information from PeopleSoft EnterpriseOne. XML transaction uses interface table functionality.
XML Transaction Service (XTS)	Transforms an XML document that is not in the PeopleSoft EnterpriseOne format into an XML document that can be processed by PeopleSoft EnterpriseOne. XTS then transforms the response back to the request originator XML format.
Z event	A service that uses interface table functionality to capture PeopleSoft EnterpriseOne transactions and provide notification to third-party software, end users, and other PeopleSoft systems that have requested to be notified when certain transactions occur.
Z table	A working table where non-PeopleSoft EnterpriseOne information can be stored and then processed into PeopleSoft EnterpriseOne. Z tables also can be used to retrieve PeopleSoft 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 PeopleSoft EnterpriseOne database.

Index

A

- abbreviations in message column of event stream event pane 10
- action tools in Virtual Runner 26
- additional documentation viii
- All Virtual Scripts list box 25
- API playback timing 44
- application fundamentals vii
- automatic parameter value linking 14

B

- business function failures in PeopleSoft EnterpriseOne, diagnosing 54
- business function parameters, displaying 53
- buttons in the event stream event pane 11
- buttons in the script list pane 12

C

- capture and import test results 30
 - capturing results 30
 - importing test results 31
 - viewing test results 31
- causes of Virtual User Tool script failure
 - finding error entries in the log file 50
 - identifying a PeopleSoft EnterpriseOne environment problem 51
 - locating the log file in the event of early script failure 50
 - setting the MessageLevel parameter 51
- causes of Virtual User Tool script failures
 - identifying a PeopleSoft EnterpriseOne environment problem 51
 - See Also* diagnosing a PeopleSoft EnterpriseOne environment problem
 - identifying an PeopleSoft EnterpriseOne environment problem 52
 - See Also* investigating PeopleSoft EnterpriseOne errors
- code for data capture 3
- columns in the script list pane 12
- command line for Virtual Script Player launch 22

- Command section of initialization file parameters 19
- comments, submitting xii
- common elements xiii
- contact information xii
- create master scripts 36
- create virtual scripts 29
 - capturing and importing test results 30, 31
 - See Also* capturing test results; importing test results; viewing test results
 - editing the virtual script 32, 33, 34, 35, 36
 - See Also* adding loops; creating master scripts; linking values in entry scripts; linking values in inquiry scripts; script generation; value-linking parameters
- cross-references xi
- Customer Connection website viii

D

- data capture components 3, 4
 - event stream 5
 - functions of PeopleSoft AutoPilot hooks 5
 - functions of PeopleSoft EnterpriseOne software hooks 5
 - PeopleSoft AutoPilot 4, 5
 - See Also* script creation; script playback; script playback configuration
 - PeopleSoft AutoPilot and EnterpriseOne software code 5
 - PeopleSoft AutoPilot Playback Results Detail Table (F97214) 6
- data capture for virtual scripts 3
 - code for data capture 3
 - data capture components 4, 5, 6
 - See Also* PeopleSoft AutoPilot Playback Results Detail Table (F97214); event stream; PeopleSoft AutoPilot; PeopleSoft

- AutoPilot and EnterpriseOne software code
 - event stream 3
 - PeopleSoft AutoPilot 3
 - PeopleSoft AutoPilot Playback Results Detail Table (F97214) 3
 - data capture for virtual scripts, data capture components 3
 - date formatting 23
 - define a script using LoadRunner 40
 - define the host machine using LoadRunner 40
 - define virtual users using LoadRunner 41
 - documentation
 - printed viii
 - related viii
 - updates viii
 - dynamic loop creation 16
 - dynamic loop designation 16
 - dynamic loop editing 16
- E**
 - edit the virtual script 32
 - adding loops 33
 - creating master scripts 36
 - linking values in entry scripts 35
 - linking values in inquiry scripts 34
 - script generation 35
 - value-linking parameters 33
 - EnterpriseOne Virtual User Tool components, Virtual Script Editor 15
See Also looping identification
 - error entries in the Virtual User Tool log file 50
 - event graph 11
 - event pane 10
 - event stream 3, 5
 - event stream event pane
 - abbreviations in message column 10
 - buttons 11
- F**
 - functions of PeopleSoft AutoPilot hooks 5
 - functions of PeopleSoft EnterpriseOne software hooks 5
- G**
 - gather LoadRunner results 41
 - generate the virtual script 35
- H**
 - hRequest handle value linking 16
- I**
 - import test results 31
 - initialization file parameters for Virtual Script Player 19
 - Command section 19
 - Log section 21
 - Paths section 20
 - Timing section 20
 - interthread timing 44
- L**
 - launch and manage multiple script playback 40
 - defining a script 40
 - defining the host machine 40
 - defining virtual users 41
 - gathering LoadRunner results 41
 - running virtual playback from the LoadRunner controller 41
 - setting rendezvous points 41
 - limitations of Virtual Script Player 24
 - link parameters in entry scripts 35
 - link parameters in inquiry scripts 34
 - locate the causes of Virtual User Tool script failures 49
 - log file in the event of early script failure 50
 - Log section of initialization file parameters 21
 - looping identification 15
 - dynamic loop creation 16
 - dynamic loop designation 16
 - dynamic loop editing 16
 - loops 33
- M**
 - manual parameter value linking 15
 - Master Scripts list box 25
 - MessageLevel parameter, setting 51
 - MMA Partners viii
 - modes of operation for Virtual Script Player 23
 - multiple script playback problems
 - debugging LoadRunner 59
 - debugging Virtual Runner 58

N

notes xi

P

parameter detail pane 11
 parameter identification
 source parameters 13
 target parameters 13
 parameter value linking 13
 automatic 14
 manual 15
 paths section of initialization file
 parameters 20
 PeopleBooks
 ordering viii
 PeopleCode, typographical conventions x
 PeopleSoft application fundamentals vii
 PeopleSoft AutoPilot 3, 4
 script creation 4
 script playback 5
 script playback configuration 4
 PeopleSoft AutoPilot and EnterpriseOne
 software code 5
 PeopleSoft AutoPilot playback
 configuration 4
 PeopleSoft AutoPilot Playback Results
 Detail Table (F97214) 3
 PeopleSoft AutoPilot script creation 4
 PeopleSoft AutoPilot script playback 5
 PeopleSoft EnterpriseOne environment
 initialization by the Virtual Script
 Player 22
 PeopleSoft EnterpriseOne environment
 problem, diagnosing 51
 PeopleSoft EnterpriseOne environment
 problem, identifying 51, 52
 PeopleSoft EnterpriseOne errors,
 investigating 52
 PeopleSoft Virtual User Tool
 components 19, 24, 25
 See Also Virtual Runner; Virtual Script
 Player; VSMEditor
 creating virtual scripts 29, 30, 32
 See Also capturing and importing test
 results; editing the virtual script
 running virtual scripts 37, 38, 40
 See Also from a command line; from a
 single workstation; launching and

managing multiple script playback;
 using Virtual Runner

special considerations for simulated
 playback 43, 45, 46

See Also call level; playback timing;
 synchronous and asynchronous
 calls; think times

PeopleSoft Virtual User Tool components
 Virtual Runner 25

Virtual Script Editor 10, 16, 17, 18

See Also event pane; hRequest handle
 value linking; thread identification;
 timing interval maintenance;
 virtual script generation

Virtual Script Player 19, 22, 23, 24

See Also command line launch; date
 formatting; initialization file
 parameters; limitations; modes of
 operation; preprocessing of valid
 values list data; script failure; valid
 values list processing

VSMEditor 24, 25

See Also All Virtual Scripts list box;
 Master Scripts list box; VSM files

placement of hooks for data capture 5

playback timing 43

 API playback timing 44

 interthread timing 44

player session columns in Virtual
 Runner 26

preprocessing of valid values list data 23

prerequisites vii

printed documentation viii

R

related documentation viii

run virtual playback from the LoadRunner
 controller 41

run virtual scripts 37

 from a command line 38

 from a single workstation 37

 launching and managing multiple script
 playback 40, 41

See Also defining a script; defining
 the host machine; defining virtual
 users; gathering LoadRunner
 results; running virtual playback
 from the LoadRunner controller;
 setting rendezvous points

 using Virtual Runner 38

- run virtual scripts from a command line 38
- run virtual scripts from a single workstation 37
- run virtual scripts using Virtual Runner 38

S

- script failure 24
- script list pane 12
 - buttons 12
 - columns 12
- Set rendezvous points using LoadRunner 41
- source parameters 13
- special considerations for simulated playback 43
 - call level 45
 - playback timing 43, 44
 - See Also* API playback timing; interthread timing
 - synchronous and asynchronous calls 45
 - Think times 46
- suggestions, submitting xii
- synchronous and asynchronous calls 45

T

- target parameters 13
- Think Times 46
- Thread identification 17
- timing interval maintenance 17
- timing section of initialization file parameters 20
- tips and techniques for troubleshooting
 - Virtual User Tool scripts 49
 - debugging Virtual User Tool scripts 52, 53, 54, 56, 57, 58, 59
 - See Also* correcting uninitialized user handle errors; diagnosing business function failures in PeopleSoft EnterpriseOne; displaying business function parameters; identifying and correcting duplicate key errors; preventing multiple script playback problems; rectifying irregular transaction times; troubleshooting value-linking errors; verifying the validity of Virtual User Tool script data

- locating the causes of Virtual User Tool script failures 49, 50, 51
 - See Also* finding error entries in the log file; identifying a PeopleSoft EnterpriseOne environment problem; locating the log file in the event of early script failure; setting the MessageLevel parameter
- typographical conventions x

V

- valid values list processing 23
- value-linking errors
 - identifying and correcting variable value-linking errors 56
 - researching value-linking errors in the Virtual Script Editor 54
 - verifying that value linking is functioning 55
- value-linking errors, troubleshooting 53
- value-linking parameters 33
- view test results 31
- Virtual Runner 25
 - components 26
 - See Also* Actions tools; Player session columns
- Virtual script editor
 - parameter identification 13
 - See Also* target parameters
- Virtual Script Editor
 - event graph 11
 - event pane 10
 - hRequest handle value linking 16
 - looping identification 15, 16
 - See Also* dynamic loop creation; dynamic loop designation; dynamic loop editing
 - parameter detail pane 11
 - parameter identification 13
 - See Also* source parameters
 - parameter value linking 13, 14, 15
 - See Also* automatic; manual
 - script list pane 12
 - thread identification 17
 - timing interval maintenance 17
 - virtual script generation 18
- virtual script generation 18
- Virtual Script Player 19
 - command line launch 22
 - date formatting 23

- initialization file parameters 19, 20, 21
 - See Also* Command section; Log section; Paths section; Timing section
 - limitations 24
 - modes of operation 23
 - PeopleSoft EnterpriseOne environment initialization 22
 - preprocessing of valid values list data 23
 - script failure 24
 - valid values list processing 23
 - Virtual User Tool
 - capturing data for virtual scripts 3
 - components 7, 8
 - See Also* Virtual Script Editor
 - data capture for virtual scripts 3
 - See Also* code for data capture; data capture components; event stream; PeopleSoft AutoPilot; PeopleSoft AutoPilot Playback Results Detail Table (F97214)
 - Virtual Script Editor 13
 - See Also* parameter value linking
 - Virtual User Tool components 7
 - Virtual Script Editor 8, 11, 12
 - See Also* event graph; parameter detail pane; script list pane
 - Virtual User Tool scripts
 - correcting uninitialized user handle errors 59
 - identifying and correcting duplicate key errors 57
 - preventing multiple script playback problems 58, 59
 - See Also* debugging LoadRunner; debugging Virtual Runner
 - rectifying irregular transaction times 57
 - troubleshooting value-linking errors 54, 55, 56
 - See Also* identifying and correcting variable value-linking errors; researching value-linking errors in the Virtual Script Editor; verifying that value linking is functioning
 - verifying the validity of Virtual User Tool script data 56
 - Virtual User Tool scripts, debugging 52
 - diagnosing business function failures in PeopleSoft EnterpriseOne 54
 - displaying business function parameters 53
 - troubleshooting value-linking errors 53
 - visual cues xi
 - VSM files 25
 - VSMEditor 24
 - All Virtual Scripts list box 25
 - Master Scripts list box 25
 - VSM files 25
- W**
- warnings xi

