

ORACLE®

PEOPLESOFT

PeopleTools 8.55: Test Framework

November 2017

ORACLE®

PeopleTools 8.55: Test Framework

Copyright © 1988, 2017, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Contents

Preface: Preface.....	xiii
Understanding the PeopleSoft Online Help and PeopleBooks.....	xiii
PeopleSoft Hosted Online Help.....	xiii
Locally Installed Help.....	xiii
Downloadable PeopleBook PDF Files.....	xiii
Common Help Documentation.....	xiv
Field and Control Definitions.....	xiv
Typographical Conventions.....	xiv
ISO Country and Currency Codes.....	xv
Region and Industry Identifiers.....	xv
Translations and Embedded Help.....	xvi
Using and Managing the PeopleSoft Online Help.....	xvi
PeopleTools Related Links.....	xvi
Contact Us.....	xvii
Follow Us.....	xvii
Chapter 1: Getting Started with PeopleSoft Test Framework.....	19
Understanding PeopleSoft Test Framework.....	19
Terminology.....	20
Chapter 2: Installing and Configuring PTF.....	23
Understanding the PTF Development Environment.....	23
Configuring an Environment for PTF.....	24
Verifying Integration Broker Setup.....	24
Setting Up Security.....	26
Configuring the Web Profile.....	28
Defining PTF Configuration Options.....	28
Evaluating SSL Certification Requirements.....	31
Installing a PTF Client.....	32
Verifying Requirements.....	32
Configuring Browser Settings.....	33
Installing the PTF Client Software.....	35
Creating a Connection to a PTF Environment.....	37
Selecting a PTF Environment.....	40
Configuring Local Options.....	42
Configuring Execution Options in PTF Client.....	44
Configuring Execution Options in PeopleSoft Internet Architecture.....	54
Specifying Execution Options.....	54
Configuring Debugging Options.....	56
Defining Advanced Options.....	57
Specifying PeopleTools Options.....	58
Establish Export Log Options.....	59
Chapter 3: Using PeopleSoft Test Framework.....	63
Using PTF Explorer.....	63
Understanding PTF Explorer.....	63
Defining and Applying Filters.....	66
Using the Test Editor.....	68
Test Editor Menus.....	69

Test Editor Toolbar.....	71
Test Window.....	72
Test Window Fields.....	73
Test Window Toolbar.....	74
Test Step Fields.....	75
Using the PTF Test Recorder.....	76
Test Recorder Toolbar.....	77
Recording Action Tools.....	77
Step Insert Tools.....	78
Step Modification Tools.....	81
Recorder Utility Tools.....	81
Using the Log Viewer.....	83
Chapter 4: Creating Tests and Test Cases.....	87
Creating Tests.....	87
Creating a New Folder.....	87
Creating a New Test.....	87
Naming Tests.....	88
Copying a Test.....	88
Renaming a Test.....	88
Renaming a Folder.....	89
Recording Tests.....	89
Test Action Tools.....	90
Recording a Test.....	91
Opening Tests.....	91
Opening Tests with PTF Explorer.....	92
Opening Tests Assets with the Quick Open Dialog Box.....	92
Working with Test Cases.....	94
Creating a New Test Case.....	94
Creating a Test Case With Values.....	95
Managing Values when Pasting Test Steps.....	96
Exporting and Importing Test Cases.....	96
Exporting Test Cases.....	96
Importing Test Cases.....	97
Executing Tests.....	98
Executing a Test.....	98
Executing a Test Case.....	99
Executing a Test from a Specific Step.....	99
Executing a Test from the Command Line.....	100
Reviewing Test Logs.....	103
Exporting Test Logs to XML.....	106
Organizing Tests In PTF Explorer.....	108
Cutting and Pasting Multiple Tests or Folders.....	108
Deleting Multiple Tests or Folders.....	108
Expanding or Collapsing Tests and Folders.....	108
Chapter 5: Developing and Debugging Tests.....	109
Using the Message Tool.....	109
Using Reserved Words.....	112
Using Variables.....	113
Using Text Strings as Parameters in Functions.....	116
Using Persistent Variables.....	117
Setting Variable Option in the Test.....	117

Setting Persistent Variable Options.....	118
Managing Persistent Variables.....	118
Example of a Test that uses Persistent Variables.....	118
Using Conditional Logic.....	119
Handling Application Messages.....	120
Interpreting Logs.....	122
Incorporating Scroll Handling.....	123
Calling Tests.....	127
Understanding Calling Tests.....	128
Using Library Tests.....	128
Using Parameters with Library Tests.....	128
Using Shell Tests.....	130
Sharing Test Assets.....	130
Chapter 6: Administering PTF.....	133
Managing PTF Logs.....	133
Understanding Log Manager.....	133
Using Log Manager Toolbar.....	134
Using Log Manager Fields.....	135
Using the Selection Pane.....	135
Using the Trace Pane.....	136
Upgrading Tests.....	136
Working with Application Designer Projects in PTF Client.....	138
Managing Application Designer Projects in PTF Client.....	139
Migrating PTF Tests.....	142
Performing Mass Updates.....	142
Understanding Mass Update.....	143
Using the Mass Update Page.....	143
Finding and Replacing Test Step Data.....	145
Chapter 7: Identifying Change Impacts.....	149
Understanding Change Impacts.....	149
Defining Analysis Rules.....	149
Defining Analysis Rules.....	150
Creating Test Maintenance Reports.....	151
Step 1 of 4: Manual Tasks.....	152
Step 2 of 4: Analyze Compare Data.....	154
Step 3 of 4: Select an Analyzed Project.....	155
Step 4 of 4: Generate Report.....	156
Interpreting Test Maintenance Reports.....	159
Understanding Test Coverage Reports.....	161
Creating Test Coverage Reports.....	162
Step 1 of 2: Select a Project.....	162
Step 2 of 2: Generate Report.....	163
Using Usage Monitor Data with PTF.....	165
Configuring Usage Monitor.....	166
Generating Usage Monitor Data.....	166
Administering Usage Monitor for PTF.....	166
Interpreting Test Coverage Reports.....	167
Running Test Details Reports.....	168
Creating a Test Compare Report.....	170
Creating Test Matrix Reports.....	172
Querying PTF Report Tables.....	174

Chapter 8: Incorporating Best Practices	177
Incorporating PTF Best Practices.....	177
Keep your Desktop Simple.....	177
Adopt Naming Conventions.....	177
Record First.....	178
Document Tests.....	178
Clean Up Tests.....	179
Use Configuration and Execution Options.....	179
Use Page Prompting.....	180
Use the Process Step Type.....	180
Make Tests Dynamic.....	181
Reduce Duplication.....	182
Chapter 9: Using the PTF Test Language	183
Understanding the PTF Test Structure.....	183
PTF Test Language.....	185
Validation.....	185
Parameters.....	185
Variables.....	186
Reserved Words.....	186
System Variables.....	187
Syntax Check.....	187
Context Sensitive Help within Grid for Function Parameter Details.....	189
Chapter 10: Test Language Reference	191
Step Types.....	191
Browser.....	191
Close.....	191
CloseAll.....	191
Count.....	192
Exists.....	192
FrameExists.....	192
FrameSet.....	193
Get_Active.....	193
Set_Active.....	193
Set_URL.....	194
Start.....	194
Start_Login.....	194
WaitForNew.....	194
Button.....	195
Click.....	195
Exists.....	196
Get_Property.....	196
Get_Style.....	196
Verify.....	196
Chart.....	196
ChartGetType.....	196
GetText.....	197
SectionCount.....	197
CheckBox.....	198
Exists.....	198
Get_Property.....	199
Get_Style.....	199

GetLabel.....	199
Set_Value.....	199
Verify.....	199
ComboBox.....	199
Exists.....	200
Get_Property.....	200
Get_Style.....	200
GetLabel.....	200
Set_Value.....	200
ValueExists.....	201
Verify.....	201
Command.....	202
Exec.....	202
Conditional.....	204
Else.....	204
End_If.....	204
If_Then.....	204
DataMover.....	205
Exec.....	205
DateTime.....	206
Exists.....	206
Get_Property.....	206
Get_Style.....	206
GetLabel.....	206
Set_Value.....	207
Verify.....	207
Div.....	207
Click.....	207
Exists.....	207
Get_Property.....	207
Get_Style.....	208
ScrollIt.....	208
Execution.....	208
Set_Options.....	208
Skip_Login.....	208
Skip_PageSave.....	208
Skip_RunRequest.....	209
StopOnError.....	209
File.....	209
Download.....	209
Upload.....	210
Upload_ByLink.....	211
Header.....	211
Exists.....	211
Get_Property.....	211
Get_Style.....	211
Verify.....	211
HTMLTable.....	212
CellClick.....	212
CellClickOnChkB.....	212
CellClickOnImage.....	213

CellClickOnLink.....	213
CellExists.....	213
CellGetIndex.....	214
CellGetValue.....	215
ColCount.....	216
RowCount.....	216
Image.....	217
Click.....	217
Exists.....	217
Get_Property.....	217
Get_Style.....	217
RightClick.....	218
Label.....	218
Exists.....	218
Get_Property.....	218
Get_Style.....	218
Verify.....	218
Link.....	219
Click.....	219
Exists.....	219
Get_Property.....	219
Get_Style.....	219
SaveTargetAs.....	219
Verify.....	220
Log.....	220
Fail.....	220
Message.....	220
Pass.....	221
SnapShot.....	221
Warning.....	221
LongText.....	222
Exists.....	222
Get_Property.....	222
Get_Style.....	223
GetLabel.....	223
Set_Value.....	223
SetValue_InModal.....	223
Verify.....	223
Loop.....	223
Do.....	224
End_Loop.....	224
Exit_Loop.....	224
For.....	224
Next.....	225
While.....	225
MsgBox.....	226
Actions for MsgBox Step Type.....	226
MultiSelect.....	228
Exists.....	228
Get_Property.....	228
Get_Style.....	229

GetLabel.....	229
Set_Value.....	229
Verify.....	229
Number.....	230
Exists.....	230
Get_Property.....	230
Get_Style.....	230
GetLabel.....	230
Set_Value.....	231
Verify.....	231
Page.....	231
Expand.....	231
Go_To.....	231
Prompt.....	231
PromptOK.....	232
Save.....	233
SecPage_Close.....	233
SecPage_Open.....	233
Process.....	233
Run.....	234
Run_Def.....	235
Pwd.....	236
Exists.....	236
GetLabel.....	236
Set_Value.....	236
Query.....	236
Exec.....	237
Exec_Private.....	237
Radio.....	239
Exists.....	239
Get_Property.....	239
Get_Style.....	239
GetLabel.....	240
Set_Value.....	240
Verify.....	240
Range.....	240
Exists.....	240
Get_Property.....	240
Get_Style.....	241
GetLabel.....	241
Set_Value.....	241
Verify.....	241
RichText.....	241
GetLabel.....	241
Set_Value.....	241
Scroll.....	242
Action.....	242
Definition.....	243
Key_Set.....	245
ModalGrid_Close.....	246
ModalGrid_Open.....	246

Reset.....	246
RowCount.....	246
Span.....	247
Click.....	248
Exists.....	248
Get_Property.....	248
Get_Style.....	248
GetLabel.....	248
MouseOver.....	248
MouseOverClose.....	249
Verify.....	250
Test.....	250
Exec.....	250
Text.....	251
Exists.....	251
Get_Property.....	251
Get_Style.....	251
GetLabel.....	251
Set_Value.....	251
Verify.....	252
UsageMonitor.....	252
Start.....	252
Stop.....	252
Variable.....	252
Set_Value.....	252
Wait.....	253
For_Seconds.....	253
For_Value.....	253
Common Actions.....	254
Click.....	254
Exists.....	255
Get_Property.....	256
Get_Style.....	257
GetLabel.....	258
Set_Value.....	259
Verify.....	260
Reserved Words.....	261
#CHECK#.....	261
#DIS#.....	261
#DTTM.....	262
#EXIST#.....	262
#FAIL#.....	263
#IGNORE.....	263
#LIKEF#.....	264
#LIKEW#.....	265
#LIST#.....	265
#NOTEXIST#.....	266
#NOTHING.....	266
#PREFIX#.....	267
#TODAY.....	267
#WARN#.....	268

System Variables.....	268
Functions.....	269
Add.....	270
Concat.....	270
Date.....	271
Day.....	271
Divide.....	272
GetField.....	273
Hour.....	273
InStr.....	274
LCase.....	275
Left.....	275
Len.....	276
MakeDate.....	276
MakeTime.....	277
Minute.....	278
Month.....	279
Multiply.....	279
Now.....	280
Replace.....	281
Right.....	281
Round.....	282
Second.....	282
SubStr.....	283
Subtract.....	284
Sum.....	284
Time.....	285
Trim.....	286
UCase.....	286
Weekday.....	287
Year.....	288
Appendix A: Reserved Words Quick Reference.....	289
Reserved Words.....	289

Preface

Understanding the PeopleSoft Online Help and PeopleBooks

The PeopleSoft Online Help is a website that enables you to view all help content for PeopleSoft Applications and PeopleTools. The help provides standard navigation and full-text searching, as well as context-sensitive online help for PeopleSoft users.

PeopleSoft Hosted Online Help

You access the PeopleSoft Online Help on Oracle's PeopleSoft Hosted Online Help website, which enables you to access the full help website and context-sensitive help directly from an Oracle hosted server. The hosted online help is updated on a regular schedule, ensuring that you have access to the most current documentation. This reduces the need to view separate documentation posts for application maintenance on My Oracle Support, because that documentation is now incorporated into the hosted website content. The Hosted Online Help website is available in English only.

Note: Only the most current release of hosted online help is updated regularly. After a new release is posted, previous releases remain available but are no longer updated.

Locally Installed Help

If your organization has firewall restrictions that prevent you from using the Hosted Online Help website, you can install the PeopleSoft Online Help locally. If you install the help locally, you have more control over which documents users can access and you can include links to your organization's custom documentation on help pages.

In addition, if you locally install the PeopleSoft Online Help, you can use any search engine for full-text searching. Your installation documentation includes instructions about how to set up Oracle Secure Enterprise Search for full-text searching. See *PeopleTools Installation* for your database platform, "Installing PeopleSoft Online Help." If you do not use Secure Enterprise Search, see the documentation for your chosen search engine.

Note: Beginning with PeopleSoft PeopleTools 8.55.11, you can use Elasticsearch as the search engine for locally installed PeopleSoft Online Help.

See [Oracle Support Document 2189508.1 \(PeopleTools 8.55.11 Elasticsearch Documentation Home Page\)](#) for more information on using Elasticsearch with PeopleSoft.

Note: Before users can access the search engine on a locally installed help website, you must enable the Search field. For instructions, select About This Help, Managing Locally Installed PeopleSoft Online Help, Enabling the Search Button and Field in the Contents sidebar of the online help site.

Downloadable PeopleBook PDF Files

You can access downloadable PDF versions of the help content in the traditional PeopleBook format. The content in the PeopleBook PDFs is the same as the content in the PeopleSoft Online Help, but it has

a different structure and it does not include the interactive navigation features that are available in the online help.

Common Help Documentation

Common help documentation contains information that applies to multiple applications. The two main types of common help are:

- Application Fundamentals
- Using PeopleSoft Applications

Most product families provide a set of application fundamentals help topics that discuss essential information about the setup and design of your system. This information applies to many or all applications in the PeopleSoft product family. Whether you are implementing a single application, some combination of applications within the product family, or the entire product family, you should be familiar with the contents of the appropriate application fundamentals help. They provide the starting points for fundamental implementation tasks.

In addition, the *PeopleTools: Applications User's Guide* introduces you to the various elements of the PeopleSoft Pure Internet Architecture. It also explains how to use the navigational hierarchy, components, and pages to perform basic functions as you navigate through the system. While your application or implementation may differ, the topics in this user's guide provide general information about using PeopleSoft Applications.

Field and Control Definitions

PeopleSoft documentation includes definitions for most fields and controls that appear on application pages. These definitions describe how to use a field or control, where populated values come from, the effects of selecting certain values, and so on. If a field or control is not defined, then it either requires no additional explanation or is documented in a common elements section earlier in the documentation. For example, the Date field rarely requires additional explanation and may not be defined in the documentation for some pages.

Typographical Conventions

The following table describes the typographical conventions that are used in the online help.

<i>Typographical Convention</i>	<i>Description</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.
. . . (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.

Typographical Convention	Description
& (ampersand)	When placed before a parameter in PeopleCode syntax, an ampersand indicates that the parameter is an already instantiated object. Ampersands also precede all PeopleCode variables.
=>	This continuation character has been inserted at the end of a line of code that has been wrapped at the page margin. The code should be viewed or entered as a single, continuous line of code without the continuation character.

ISO Country and Currency Codes

PeopleSoft Online Help topics use International Organization for Standardization (ISO) country and currency codes to identify country-specific information and monetary amounts.

ISO country codes may appear as country identifiers, and ISO currency codes may appear as currency identifiers in your PeopleSoft documentation. Reference to an ISO country code in your documentation does not imply that your application includes every ISO country code. The following example is a country-specific heading: "(FRA) Hiring an Employee."

The PeopleSoft Currency Code table (CURRENCY_CD_TBL) contains sample currency code data. The Currency Code table is based on ISO Standard 4217, "Codes for the representation of currencies," and also relies on ISO country codes in the Country table (COUNTRY_TBL). The navigation to the pages where you maintain currency code and country information depends on which PeopleSoft applications you are using. To access the pages for maintaining the Currency Code and Country tables, consult the online help for your applications for more information.

Region and Industry Identifiers

Information that applies only to a specific 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 region-specific heading: "(Latin America) Setting Up Depreciation"

Region Identifiers

Regions are identified by the region name. The following region identifiers may appear in the PeopleSoft Online Help:

- 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 the PeopleSoft Online Help:

- USF (U.S. Federal)
- E&G (Education and Government)

Translations and Embedded Help

PeopleSoft 9.2 software applications include translated embedded help. With the 9.2 release, PeopleSoft aligns with the other Oracle applications by focusing our translation efforts on embedded help. We are not planning to translate our traditional online help and PeopleBooks documentation. Instead we offer very direct translated help at crucial spots within our application through our embedded help widgets. Additionally, we have a one-to-one mapping of application and help translations, meaning that the software and embedded help translation footprint is identical—something we were never able to accomplish in the past.

Using and Managing the PeopleSoft Online Help

Click the Help link in the universal navigation header of any page in the PeopleSoft Online Help to see information on the following topics:

- What's new in the PeopleSoft Online Help.
- PeopleSoft Online Help accessibility.
- Accessing, navigating, and searching the PeopleSoft Online Help.
- Managing a locally installed PeopleSoft Online Help website.

PeopleTools Related Links

[Hosted Online Help Home](#)

[PeopleSoft Information Portal](#)

[My Oracle Support](#)

[Oracle University](#)

[Oracle's PeopleSoft PeopleTools 8.55 Documentation Home Page \(Doc ID 2052626.1\)](#)

"PeopleTools Product/Feature PeopleBook Index" (PeopleTools 8.55: Getting Started with PeopleTools)

Contact Us

Send your suggestions to PSOFT-INFODEV_US@ORACLE.COM. Please include release numbers for the PeopleTools and applications that you are using.

Follow Us



[Facebook.](#)



[YouTube](#)



[Twitter@PeopleSoft_Info.](#)



[PeopleSoft Blogs](#)



[LinkedIn](#)

Chapter 1

Getting Started with PeopleSoft Test Framework

Understanding PeopleSoft Test Framework

PTF automates various tasks within the PeopleSoft application, primarily functional testing. Automating functional testing enables testers to execute more tests with greater accuracy during a shorter time.

PTF works by replicating the actions of a single user executing functional tests against the PeopleSoft browser-based application. Users can record manual test procedures and save them within the framework. Later (perhaps after an application upgrade or patch), those tests can be executed against the application to verify whether the application still behaves as expected. This method for capturing and executing tests is often called the *record and playback* approach to automation.

Test assets (tests and test cases) are stored in a database as Application Designer objects. As a result, test assets are PeopleTools-managed objects, which can be managed along with other PeopleTools-managed objects through PeopleSoft Lifecycle Management.

PTF includes a number of features not available in other commercially available record and playback automation tools, including:

- The ability to validate recorded objects against PeopleSoft object metadata definitions.

As a result, the tester is able to assertively verify the existence of test objects before running a test rather than running the test to identify invalid object definitions by trial and error.

See [Understanding Change Impacts](#).

- Features that help users manipulate data within the PeopleSoft rowset-oriented data structure.

See [Incorporating Scroll Handling](#).

- Functionality that automates numerous PeopleSoft-specific functions, such as running processes through Process Scheduler.

See [Process](#).

- Functionality that interfaces with other PeopleSoft automation tools, such as Data Mover and PsQuery.

See [Query](#).

See [DataMover](#).

You should be aware that PTF is not designed to:

- Validate certain types of information, such as image appearance and relative position of data and online objects. PTF is a functional test tool rather than a user interface or browser testing tool.
- Be a load testing tool; it replicates the experience of a single user running the application.
- Replicate certain types of user actions, such as drag-and-drop mouse actions.
- Recognize or validate certain types of objects you might find in third-party or external applications, such as Flash/Flex objects, data displayed in HTML regions, and so on. PTF is designed to validate objects in the PeopleSoft application.

Terminology

This table defines some PTF terms:

Asset	See Test Asset.
Execution Options	A list of application environments available to the tester. Execution options store application environment information such as URL, user ID, password, and Process Scheduler server, and information needed to run DataMover. PTF supplies this information to the test by default when a test does not explicitly specify such information.
Explorer	See PTF Explorer.
Grid	See Scroll.
Hook	Establish a connection between a PTF test and a PeopleSoft application browser.
Library	Similar to a test, a library contains one or more steps that together automate some discrete amount of test functionality. Unlike a test, a library is never executed by itself. Rather, libraries are meant to be called (sometimes repetitively) by tests.
Log	An object that saves the experience of a single test execution event. Logs report the success or failure of the test execution and include messages and screen shots to indicate where errors occurred.
Log Manager	A tool that enables PTF administrators to purge unneeded logs
Maintenance	The process of updating PTF tests and test cases to reflect object modifications present in upgrades or changes to the PeopleSoft application. This is done by way of a direct connection to the PeopleSoft metadata, not by executing the test. For example, if a field is renamed in an upgrade, the PTF maintenance process can warn the user that a test containing a reference to the old field name will likely fail to find the object by the old identification method. The maintenance process can help the

user find the obsolete field reference and replace it with the valid (renamed) field reference before executing the test.

Mass Updated

A tool that enables you to modify test steps across multiple tests using search and replace.

Project

A PeopleTools Application Designer project. Projects are the primary means for moving PTF Tests and Test Case objects between databases.

PTF Client

An instance of the PTF executable program installed on an individual user's machine.

PTF Environment

An instance of a PeopleSoft application that has been configured to exchange data with one or more PTF clients, enabling clients to save and retrieve test assets from the application database.

PTF Explorer

A view of the PTF test assets stored within an application database. The system stores assets in a tree structure with collapsible folders for organizing the test assets. The pane containing the tree is the first pane visible to the user after startup and will always be the leftmost pane in the PTF user interface. It is labeled with the name of the application database.

Recognition

The means that the PTF client uses to identify (or find) HTML objects within the application. Often, this is the HTML ID property of the object.

Recorder

A feature of the PTF tool that is the primary means for creating new tests. While the Recorder is active, the PTF tool converts all of the user's manual test steps into steps that can be saved as an automated test.

Screen Shot

An image generated during test execution. A screen shot can be generated automatically by PTF to show the application window immediately after an error condition, or as a result of a step that uses the Log.Snapshot step.

Scroll

A scroll represents a rowset, which is a set of rows of data uniquely identified by one or more key fields. Rows in a scroll can contain child rowsets. Scrolls are rendered on PeopleSoft pages as grids of data or as a grouping of fields in a scroll area.

Scroll Area

See Scroll.

Step

The smallest unit of test functionality in PTF. A test will contain a number of steps. A step typically corresponds to a single manual test step or test instruction.

Test Asset

An object used in PTF to automate a functional test. PTF test assets are saved in the application database and can be retrieved at any time to help automate tests. The five types of test assets are:

- Execution Options
- Libraries
- Logs
- Tests
- Test Cases

Test

The primary type of test asset in PTF. Tests contain steps that replicate the action of a tester executing a functional test against the PeopleSoft application.

Test Case

A set of data associated with a test corresponding to the values entered or verified in the application. For example, if a hire test hires three similar employees into the PeopleSoft system, a user might elect to record one test and to configure that test to call three test cases, one for each employee hired. A test can have multiple test cases associated with it.

Test Editor

A space within the PTF user interface where users can edit individual tests and test cases. The Test Editor displays a test as a series of steps presented as rows within the test. Users can open multiple Test Editor panes to edit multiple tests simultaneously.

Variable

A variable is a reference to a section of computer memory that can be used to store information that is subject to change or modification. PTF tests often use variables to store values that are not known until the test is executed. For example, you could use a variable to store an ID number generated by the PeopleSoft application during the test. Later in the test, the value of the variable could be manipulated, if necessary, and then used to set or validate other information within the PeopleSoft application.

Chapter 2

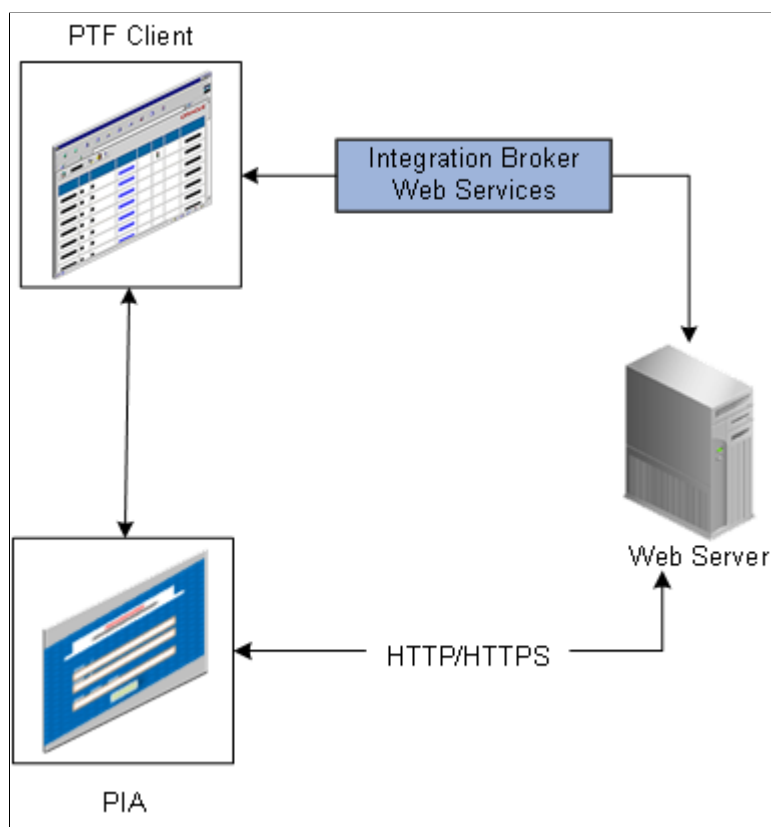
Installing and Configuring PTF

Understanding the PTF Development Environment

The following diagram illustrates the PeopleSoft Test Framework (PTF) development environment:

Image: Diagram of the PTF development environment

This diagram illustrates the PTF development environment.



A PTF development environment consists of the following elements:

- A PTF client instance.
- An internet browser instance.
- A connection to a PeopleSoft application that is to be tested through the Integration Broker Web Services.

PeopleSoft Test Framework (PTF) client is a standalone program that runs on a Microsoft Windows 64 bit machine.

Microsoft Internet Explorer 11 is required to record tests. The browsers Chrome, Firefox, Internet Explorer, or Microsoft Edge are supported to execute tests. Please refer to the Certifications tab on My Oracle Support for current information on supported browsers.

PTF client connects to PeopleSoft application database using a secure HTTPS connection through PeopleTools Integration Broker web services which runs on the web server. All the recorded tests are saved to a test repository in the application database. While executing a test, the test repository interacts with the web server.

Pure Internet Architecture (PIA) verifies a stable Integration Broker setup, and a secured access to the PTF client. You can define the PTF Configuration Options and evaluate the SSL certificate requirements using the PIA. The PIA connects to the web server through a web browser using HTTP/HTTPS.

Note: The PeopleSoft application database where test assets are stored and the PeopleSoft application that is to be tested are not required to be on the same database, but we strongly recommend you use the same database for both.

Configuring an Environment for PTF

PTF test assets (tests and test cases) are stored in tables in a PeopleSoft application database. Any application database that is certified to run on PeopleTools 8.51 or greater can be used as a PTF environment.

To configure an environment for PTF, you need to complete the following tasks:

1. Verify Integration Broker setup.
2. Set up security.
3. Configure the Web Profile.
4. Define PTF Configuration Options.
5. Evaluate SSL certificate requirements.

Verifying Integration Broker Setup

To verify that Integration Broker is set up for your application:

1. In your PeopleSoft application, navigate to PeopleTools, Integration Broker, Configuration, Gateways.
2. Verify that the Gateway URL field references the correct machine name.
3. Click the Ping Gateway button.
4. Verify that the message returns a status of ACTIVE.
5. Click the Gateway Setup Properties link.
6. Sign on to access `integrationGateway.properties` file.

7. The default user ID is *administrator*, and the default password was created when PIA was installed (please contact your security administrator for the password).
8. Verify that the Gateway Default App Server URL is specified.

This is an example of the Gateways page:

Image: Integration Broker Gateways page

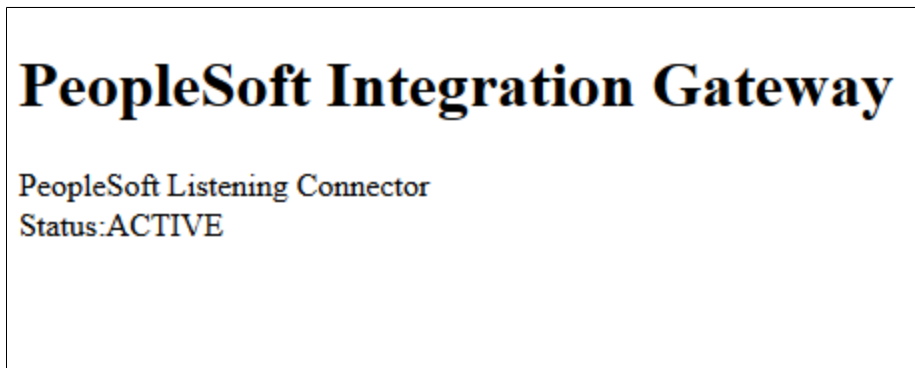
This example illustrates the fields and controls on the Integration Broker Gateways page.

The port number in the URL (8010 in this example) is the http port of the web server.

This is an example of a Ping message showing ACTIVE status:

Image: Integration Gateway Ping message

This example illustrates a successful Integration Gateway Ping message.



Click the Gateway Setup Properties link on the Gateways page to access the PeopleSoft Node Configuration page, as shown in this example:

Image: PeopleSoft Node Configuration page

This example illustrates the fields and controls on the PeopleSoft Node Configuration page.

PeopleSoft Node Configuration

URL: http://SLC02QFL:8920/PSIGW/PeopleSoftListeningConnector

Gateway Default App. Server

App Server URL	User ID	Password	Tools Release	Domain Password	Virtual Server Node
//MYSERVER:9010	PTACMADMIN	8.54-810-R1	

PeopleSoft Nodes

Node Name	App Server URL	User ID	Password	Tools Release	Domain Password		
QE_LOCAL	//MYSERVER:9211	PTACMADMIN	8.54-810-R1	Ping Node	+ -

[Advanced Properties Page](#)

The port number in the App Server URL (9010 in this example) generally corresponds with the JSL Port Number as defined in the Application Server configuration. The default port number is 9000.

When the web server is connected to more than one database you will need to enter a node name, as defined in PeopleSoft Nodes on the PeopleSoft Node Configuration page, in the Node ID field of the PTF Signon dialog box. Contact your Integration Broker administrator to determine the correct node name to use. If no node is defined in PeopleSoft Nodes on this page, leave the Node ID field of the PTF Signon dialog blank.

See [Creating a Connection to a PTF Environment](#).

Note: If you rerun the PIA installer, the PeopleSoft Node Configuration page data is cleared and needs to be reentered.

Verify that the Default User ID for the ANONYMOUS node has, at a minimum, a PTF User role.

1. Navigate to Integration Broker, Integration Setup, Nodes.
2. Select the ANONYMOUS node.
3. Note the Default User ID.
4. Navigate to PeopleTools, Security, User Profiles, User Profiles.
5. Select the User ID you identified in Step 3.
6. Access the Roles tab.
7. Verify that one of the PTF roles is present.

See [Setting Up Security](#).

If Integration Broker is not set up correctly, contact your Integration Broker administrator.

Setting Up Security

Users connecting to a PTF test environment must have one of these roles associated with their user ID:

- PTF User

- PTF Editor
- PTF Administrator

This table details the privileges associated with the PTF security roles:

Privilege	PTF User	PTF Editor	PTF Administrator
Run Tests	Yes	Yes	Yes
Create, Modify, and Delete Tests	No*	Yes	Yes
Create, Modify, and Delete Test Cases	Yes	Yes	Yes
Create or Modify Execution Options	No	No	Yes
Use Log Manager	No	No	Yes
Define Configuration Options	No	No	Yes
Create Test Maintenance Reports	No	No	Yes
Create Test Coverage Reports	No	No	Yes
Insert Tests/Test Cases into Application Designer projects	No	No	Yes

*PTF User can create, modify, and delete tests only in myFolder.

Modify Test includes these actions:

- Record tests.
- Add, modify, and delete test steps.
- Define and modify message recognition.
- Modify test properties.
- Add and modify test comments.
- Add and modify test step comments.
- Modify line information.

Note: The Default User ID for the ANONYMOUS node must have, as a minimum, a PTF User role.

If PTF security is not configured properly you may receive an error message when signing on to the PTF client indicating that the UserID and Password are not correct.

Possible causes and solutions for this error are:

- The user ID for the ANONYMOUS node does not have PTF privileges. Add at least the PTF User role to the user profile.
- The user ID you entered in the User field in the Environment Login does not have PTF privileges. Add at least the PTF User role to the user profile.

For more information on entering roles for a user profile, see "Defining Role Options" (PeopleTools 8.55: Security Administration)

Configuring the Web Profile

Complete the following steps to configure the web profile settings for the PeopleSoft application that you are testing.

1. Access the Web Profile Configuration page (PeopleTools, Web Profile, Web Profile Configuration).
2. Select the profile name for your environment. (This is the web profile that was selected during web server installation.)
3. Click the Debugging tab.
4. Check the Show Connection & Sys Info check box.

If this option is not selected PTF will not record menu, component, and page metadata correctly.

5. Check the Generate HTML for Testing check box.

If this option is not selected PTF will not record HTML objects correctly.

Image: Web Profile Configuration - Debugging page

This example illustrates the fields and controls on the Web Profile Configuration - Debugging page.

General | Security | Virtual Addressing | Cookie Rules | Caching | **Debugging** | Look and Feel | ▶

Profile Name: DEV

- Trace Monitoring Server ?
- Trace PPM Agent ?
- Show Connection & Sys Info ?
- Show Trace Link at Signon ?
- Show Layout ?
- Show Overlapping Fields ?
- Show StyleSheet Inline HTML ?
- Generate HTML for Testing ?
- Write Dump File ?
- Create File from PIA HTML Page ?
- Use Unminified JavaScript ?

Defining PTF Configuration Options

Use the Define Configuration Options page (PSPTTSTCONFIG) to:

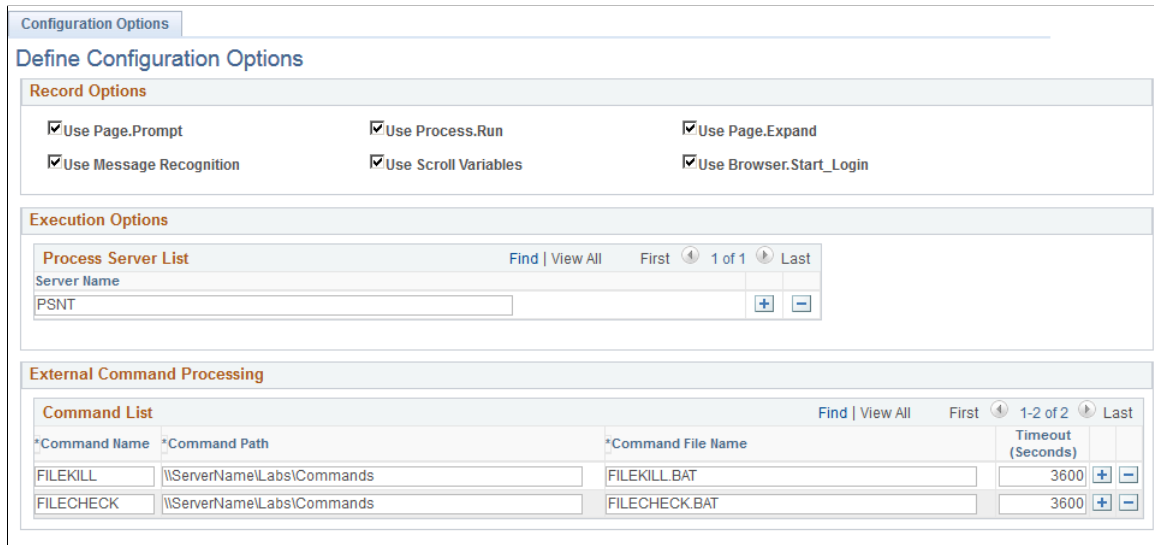
- Define record and execution options.
- Configure external command processing.

Navigation

PeopleTools, Lifecycle Tools, Test Framework, Define Configuration Options

Image: Define Configuration Options Page

This example illustrates the fields and controls on the Define Configuration Options Page.



Record Options

Use Page Prompt

Select to use Page Prompt and PromptOK steps during recording in place of menu navigation. The Use Page Prompt option is also available on the PTF Test Recorder toolbar. The option selected here is the default for all users in this environment. The option selected on the PTF Test Recorder toolbar overrides this selection for that recording session.

See [Page](#).

Use Message Recognition

Select to automatically create entries for the Message Recognition feature during recording. The Use Message Recognition option is also available on the PTF Test Recorder toolbar. The option selected here is the default for all users in this environment. The option selected on the PTF Test Recorder toolbar overrides this selection for that recording session.

Use Process Run

Select to use Process Run steps during recording in place of specific object actions. The Use Process Run option is also available on the PTF Recorder toolbar. The option selected here is the default for all users in this environment. The option selected on the PTF Test Recorder toolbar overrides this selection for that recording session. When this option is selected, PTF will recognize when a user accesses a run page

and will populate the Process.Run parameters with the process information. All actions in the run page or process monitor page will be ignored, because those actions will be handled by the Run.Process in execution.

Use Scroll Variables

Select to enable selection of scroll variables during recording. The Use Scroll Variables option is also available on the PTF Test Recorder toolbar. The option selected here is the default for all users in this environment. The option selected on the PTF Test Recorder toolbar overrides this selection for that recording session.

When this option is selected, a drop-down list box containing valid scroll variables appears in the PTF Test Recorder tool bar.

While recording, when the user selects any available variable from the Variable list, *in subsequent recording actions*, the PTF Recorder appends the variable to the name/ID/comment recognition string in the test.

Use Page Expand

Select to add a Page.Expand step type if the user expands a section of a page during recording. The Use Page Expand option is also available on the PTF Test Recorder toolbar. The option selected here is the default for all users in this environment. The option selected on the PTF Test Recorder toolbar overrides this selection for that recording session.

If this option is selected, when a user expands a page section, two steps are inserted:

1. A Page.Expand step, which is set to active.
2. The specific action used to expand the page, which is set to inactive.

When this option is not selected, when a user expands a page section, a step for only the specific user action is inserted.

Use Browser.Start_Login

Select this option to set the first step in a test to Browser.Start_Login. When this option is selected, the Ignore Login Steps option on the PTF Test Recorder tool bar is selected by default.

When this option is not selected, the first step in a test is set to Browser.Start, and the Ignore Login Steps option on the PTF Test Recorder tool bar is not selected, so any immediately subsequent recorded login steps are active.

The option selected here is the default for all users in this environment.

Execution Options

Process Server List

Add process server names to the list that can be selected in Execution Options.

See [Configuring Execution Options in PTF Client](#).

External Command Processing

Use this section to define the command line programs available to PTF users within any given PTF environment. When the step/action Command.Exec is defined in the test step, PTF will use the information supplied to run the command.

Note: Automation engineers may develop many executable utilities that assist in automating functional test cases. In order to limit the scope of command line programs available to PTF users within any given PTF environment, the command line program must be defined in the PeopleSoft database.

Command Name Enter the name to be used in PTF for the command line program. The command name will be all caps.

Command Path Enter the path where the command line program is located.

Note: the path to the defined external command and the executable file must be located on the application server hosting PIA for the PTF database.

Command File Name Enter the name of the command line file.

Timeout Enter the time in seconds before a command will time out.

See [Command](#).

Evaluating SSL Certification Requirements

By default, Peoplesoft Test Framework requires a secure connection to the application database (an HTTPS connection).

When you launch the Peoplesoft Test Framework client, if the database that you are attempting to connect to does not have a valid SSL certificate, an error message appears, indicating that the environment does not allow unsecured connections. You should check with your Peoplesoft Test Framework administrator to resolve the error.

Peoplesoft Test Framework administrators can activate the PTTST_CONFIG_NO_SSL web service if a connection to an unsecured database is allowed. Once the web service is active, when a user starts the Peoplesoft Test Framework client and it attempts to connect to an unsecured database, a warning message appears indicating that the connection is unsecure, but it provides an option to connect anyway.

To activate the PTTST_CONFIG_NO_SSL web service:

1. In the application database, access the Service Operations – Search page by selecting PeopleTools, Integration Broker, Integration Setup, Service Operations.
2. In the Service Operations field, enter *PTTST_CONFIG_NO_SSL* and click the Search button.
3. In the Service Operations search results, click the PTTST_CONFIG_NO_SSL link to access the Service Operations – General page.
4. Select the Regenerate Any-to-Local check box.

5. Select the Active check box to activate the service operation, then click the Save button.

Installing a PTF Client

A PTF client is an installation of the PTF executable software on an individual user's machine. It is the program that users run in order to create and execute automated tests. PTF test assets are not saved to the client machine. Rather, they are saved to an application database environment configured to exchange information with the PTF client. A PTF client does not need to be, and usually is not, installed on the same machine that hosts the PeopleSoft application environment.

To install a PTF client, you need to complete the following tasks:

1. Verify requirements.
2. Configure browser settings.
3. Install the PTF client software.
4. Create a connection to a PTF environment.
5. Select a PTF environment.
6. Configure local options.

Verifying Requirements

PTF client installation has the following requirements:

1. Microsoft Windows operating system.

Note: PTF requires a 64 Bit OS environment.

2. A supported Internet browser:

- Microsoft Internet Explorer 11 is required for recording tests and for identifying HTML objects using the Message tool, and can be used for test playback.
- Firefox, Chrome, and Microsoft Edge are supported, but *only* for test playback.

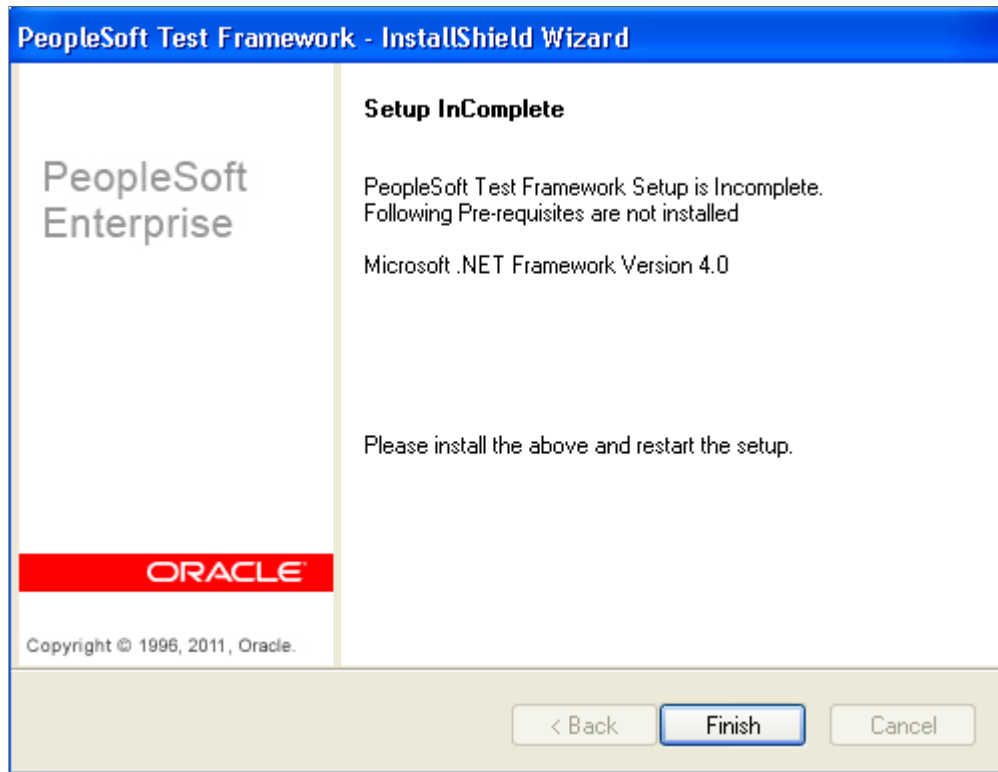
Note: For details about supported browser versions, please refer to the Certifications tab for your PeopleTools release on My Oracle Support (<https://support.oracle.com/>).

3. Microsoft .NET Framework v4.0.

If Microsoft .NET Framework version 4.0 is not present, the PTF Installer returns the following error during installation:

Image: Microsoft .NET Framework error message

This example illustrates the message received when Microsoft .NET Framework Version 4.0 is not present in the environment.



4. In order to install PTF, you will need read and write access to the PTF home directory (C:\Program Files\PeopleSoft\PeopleSoft Test Framework) by default.
5. PTF will need runtime access to the PTF data directory (C:\Documents and Settings \<User>\ApplicationData\PeopleSoft\PeopleSoft Test Framework) by default.

Note: When using a dual monitor system, PTF must be run in the primary display.

Configuring Browser Settings

Browser Security Settings: Test Application URL

You must configure the client browser security settings to accept the test application URL. If browser security settings are not properly configured you may encounter problems with PTF test playback.

Note: If you find few PeopleSoft application pages are not rendering correctly in Microsoft Internet Explorer then open the Compatibility View Settings dialog box. Remove the selection from Display intranet sites in Compatibility View check box and Use Microsoft compatibility lists check box . Select Tools, Compatibility View settings in the Internet Explorer browser to open the Compatibility View Settings dialog box.

To configure the browser security settings in Microsoft Internet Explorer:

1. In Microsoft Internet Explorer, select Tools, Internet Options.
2. In the Internet Options dialog box, access the Security tab.
3. Click the Local intranet zone.
4. Click the Sites button.
5. Click the Advanced button.
6. In the Add this website to the zone field, enter the domains for the test applications.
7. Add entries for both http and https.

For example:

```
http://*.<domain_name>
https://*.<domain_name>
```

Determine the domain name based on the URL for the test application. For example, if the URL is:

```
https://myserver.us.oracle.com:80/PTRN/signon.html
```

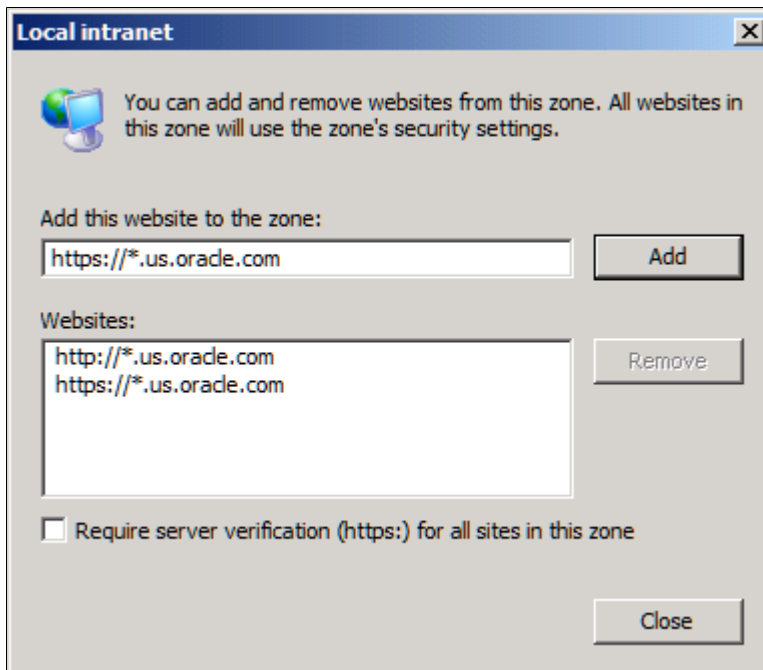
then the domain name is `us.oracle.com`

8. Click the Add button.
9. Click the Close button.
10. Click the OK button to close each open dialog box.

This example shows the Local intranet dialog box:

Image: Microsoft Internet Explorer Local intranet dialog box

This example illustrates the Microsoft Internet Explorer Local intranet dialog box



There are no specific browser security requirements for Chrome, Firefox or Edge.

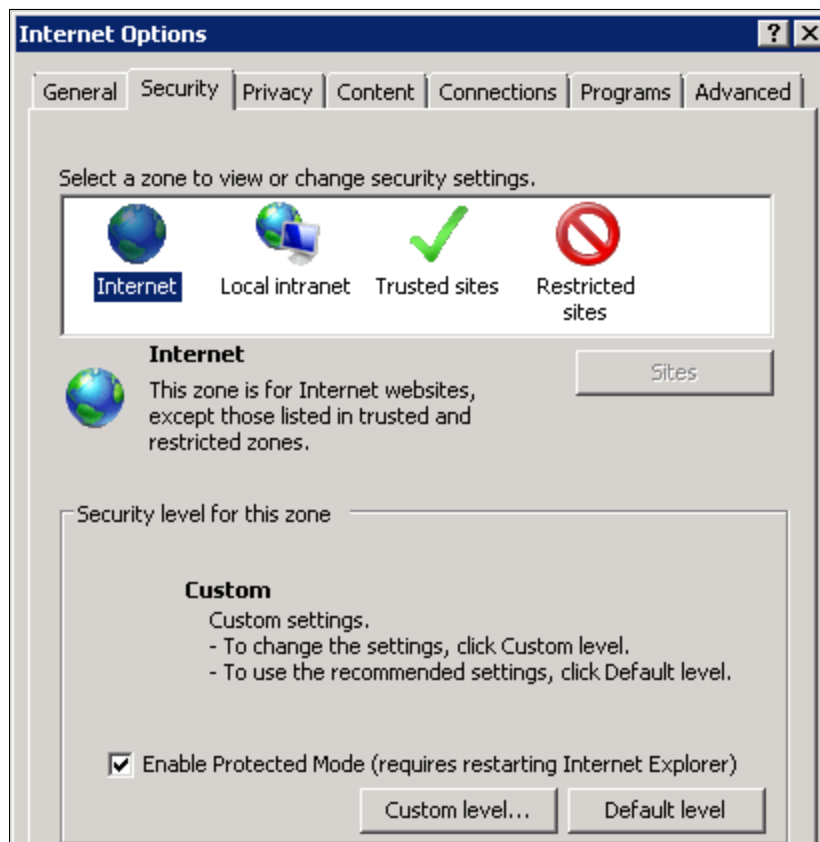
Browser Security Settings: Enable Protected Mode Option

For Microsoft Internet Explorer, the Enable Protected Mode setting for *each* zone must be equivalent – either enabled or disabled for all zones. In other words, if it is enabled for one zone, it must be enabled for all zones. Likewise, if it is disabled for one zone, it must be disabled for all zones.

In Microsoft Internet Explorer select Tools, Internet Options, Security to specify the security options for each zone.

Image: Enable Protected Mode on

This example illustrates setting Enable Protected Mode on.



There are no specific browser security requirements for Chrome, Firefox or Edge.

Zoom Settings

Browser zoom settings should be set to 100 percent. Check the lower right-hand corner of the browser and make sure it displays 100% for the zoom level.

Installing the PTF Client Software

Note: Beginning with PeopleSoft PeopleTools 8.55-08, the name of the installation executable has been changed from setup.exe to installTestFramework.exe.

When removing or installing the PTF Client, use `PS_HOME/setup/PsTestFramework/installTestFramework.exe`.

The following options are available for installing the PTF Client:

- Installing using an executable file.
- Installing (or uninstalling) from the command line (also referred to as a silent installation).

Installing Using an Executable File

To install the PTF client software using an executable file::

1. In Windows Explorer, navigate to the setup.exe executable.

If you are installing on a machine that has a PeopleTools 8.51 or later installation, setup.exe is located in the `<PS_HOME>\setup\PsTestFramework` directory.

If you are installing PTF client on another machine, the path will be `\\<machine_name>\<PS_HOME>\setup\PsTestFramework`. Your network administrator will need to make the directory accessible to users.

2. Run setup.exe.

The installation wizard appears.

3. Click the Next button.

4. You are prompted to select a folder where the wizard will install files. The default location is `C:\Program Files\PeopleSoft\PeopleSoft Test Framework`.

You can accept the default location or click the Browse button to select a different location.

5. Click the Next button.

The Ready to Install the Program page appears.

6. Click the Install button.

The InstallShield Wizard Complete page appears.

7. Click the Finish button to dismiss the install wizard.

Your PTF client software installation is complete.

8. To verify your installation, do any of the following:

- Locate the PTF shortcut on your desktop.
- Navigate to Start, All Programs, PeopleSoft Test Framework.
- In Windows Explorer, navigate to `C:\Program Files\PeopleSoft\PeopleSoft Test Framework` (or the installation directory you specified in Step 4).

Installing/Uninstalling from the Command Line

To install the PTF client software from the command line:

1. In Windows Explorer, navigate to the setup.exe file.

If you are installing on a machine that has a PeopleTools 8.51 or later installation, setup.exe is located in the <PS_HOME>\setup\PsTestFramework directory.

If you are installing PTF client on another machine, the path will be \\<machine_name>\<PS_HOME>\setup\PsTestFramework. Your network administrator will need to make the directory accessible to users.

2. Enter the following command from the command line:

```
setup.exe /s /f1"%PS_HOME%\setup\PsTestFramework\response-file.txt"
```

Note: Replace %PS_HOME% with the actual PS_HOME, or set the PS_HOME environment variable in the command window.

To uninstall from the PTF client software from the command line:, enter the following command from the command line:

```
setup.exe /s /f1"%PS_HOME%\setup\PsTestFramework\uninstall-silent-response.txt"
```

To uninstall an existing PTF instance *and* install a new instance of PTF enter the following command from the command line:

```
setup.bat
```

Creating a Connection to a PTF Environment

To create a connection to a PTF environment:

1. Run the PTF client.

Either double-click the PTF shortcut on your desktop or navigate to Start, All Programs, PeopleSoft Test Framework.

2. The PeopleSoft Test Framework - Signon dialog box appears. If you have not yet created a connection to a PTF environment, the environment signon dialog box is empty and the fields are disabled.
3. Click the New button.

Enter details for the following fields:

Database Name

Enter a descriptive name for this environment. You can use any name.

Server:Port

Enter the server name and port for the environment. Contact your Integration Broker administrator or system administrator for the correct values.

The format for the Server:Port field is:

```
<machine_name>:<https_port>
```

For example:

```
myserver.us.oracle.com:443
```

If the https port is the default 443 the port is optional.

You can also enter a complete https URL in this format:

```
https://<machine_name>:<https_port>/PSIGW/HttpListeningConne
```

For example:

```
https://myserver.us.oracle.com:443/PSIGW/HttpListeningConne
```

Use Proxy

Select this field if using a proxy server.

When you select this check box the Proxy Information link is enabled.

Proxy Information

Click this link to enter details for the proxy server.

Enter the following information for the proxy server:

- Server: Enter the server name
- Port: Enter the server port.
- User: Enter the user ID for the proxy server.

If you use network authentication, use the DOMAIN \USER format.

- Password: Enter the password.

Node ID

This field is required if more than one database is connected to the server. Enter the name of the PeopleSoft node with which the integration gateway is to communicate.

Contact your Integration Broker administrator or system administrator for the correct values.

See [Verifying Integration Broker Setup](#).

User

Enter a valid user ID for the PeopleSoft application that contains the environment. The user ID must have one of the PTF security roles assigned. Contact your security administrator to add the role if required.

If this user ID does not have PTF access you will receive a signon error:

See [Setting Up Security](#).

Password

Enter the password for this user.

4. Click the OK button.

PTF launches with a connection to the designated environment.

Image: Example of a Completed Environment Signon Dialog Box

This example illustrates a completed environment signon dialog box. In this example the Node ID field is left blank because the default gateway is used.

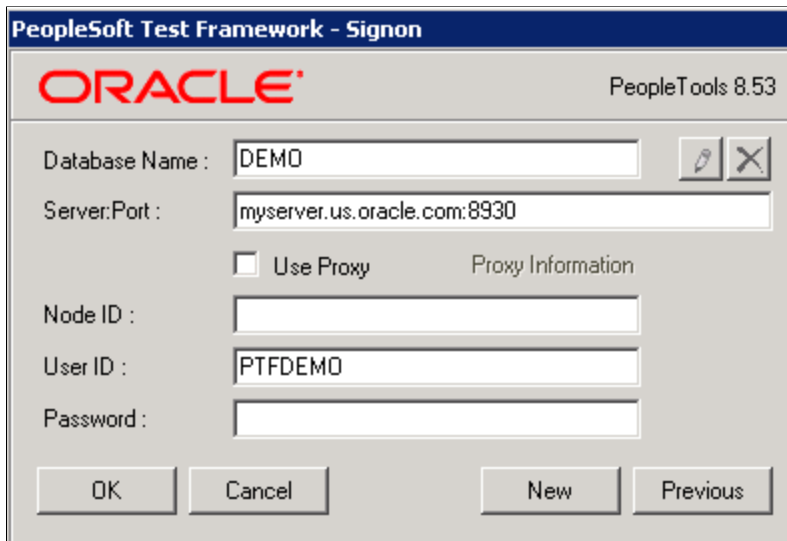
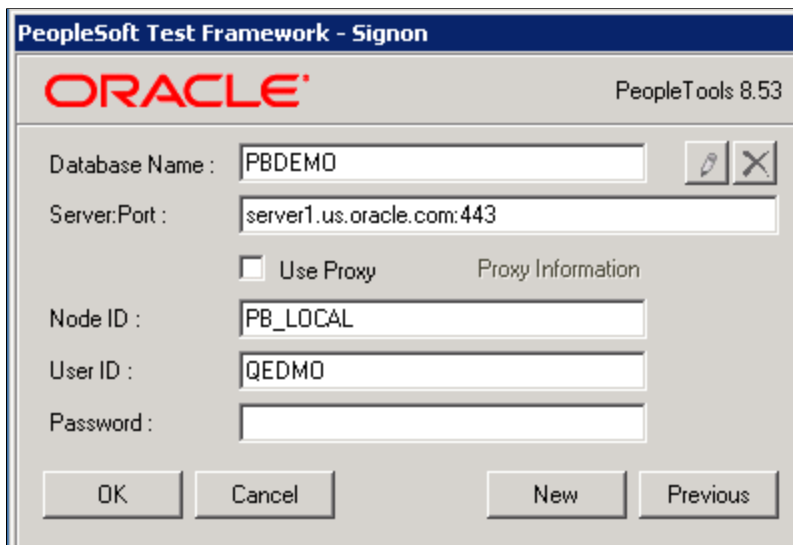


Image: Example of a Completed Environment Signon Dialog Box with Node ID Specified

This example illustrates a PeopleSoft Test Framework - Signon dialog box where the default gateway is not used. This requires that the Node ID be specified:



Note: Contact your Integration Broker administrator to determine the correct value to use for the Node ID field.

Troubleshooting Tips

This section describes some of the errors you might encounter when attempting to signon to PTF and suggests possible solutions.

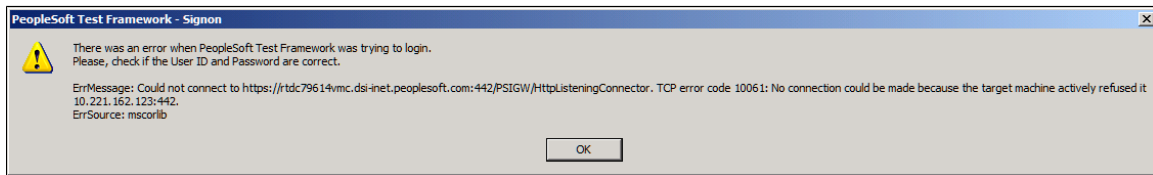
You will receive a signon error if PTF security has not been configured correctly. Possible causes and solutions for this error are:

- The user ID and password you entered are not valid for the PeopleSoft application corresponding to the entry in the Server:Port field.
- The user ID you entered in the User field in the Environment Login does not have PTF privileges. Add at least the PTF User role to the user profile.
- The user ID for the ANONYMOUS node does not have PTF privileges. Add at least the PTF User role to the user profile.

You will receive the following error message if you specify the wrong HTTPS port in the environment login URL:

Image: Wrong HTTPS Error Message

This example illustrates the error message received when the wrong HTTPS is entered.

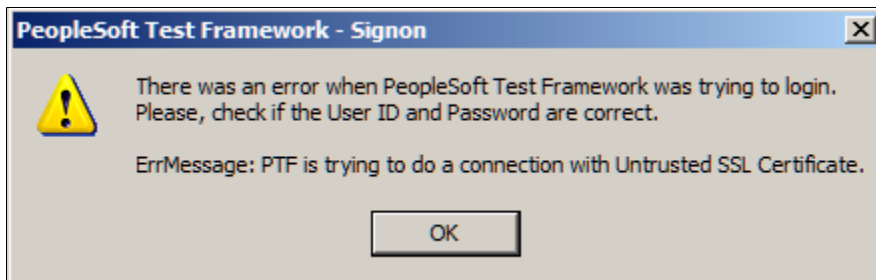


The default port is 443. If a different port was specified during installation, you will need to contact your system administrator to determine the correct port number.

You will receive the following error message if you try to logon to an environment that is not secured with an SSL Certificate. .

Image: Untrusted SSL Certificate Error Message

This example illustrates the error message received when trying to login to an unsecured environment.



PTF requires use of an HTTPS site for security purposes. Contact your PTF administrator to resolve this error.

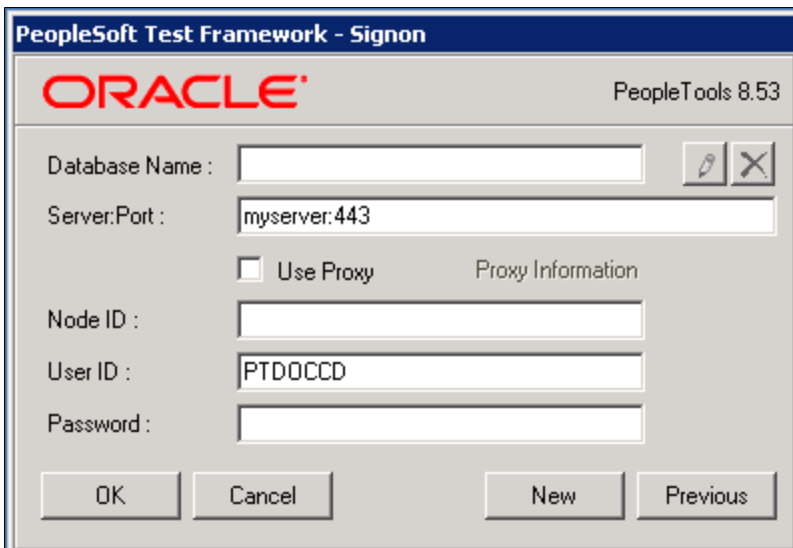
Selecting a PTF Environment

When you launch PTF again, the PeopleSoft Test Framework - Signon dialog box appears, with the last environment you used automatically selected.

You can enter the password and click the OK button to launch PTF using that environment, or you can click the New button to create another environment login.

Image: New PeopleSoft Test Framework - Signon Dialog Box

This example illustrates the dialog box for creating a new PTF environment login.

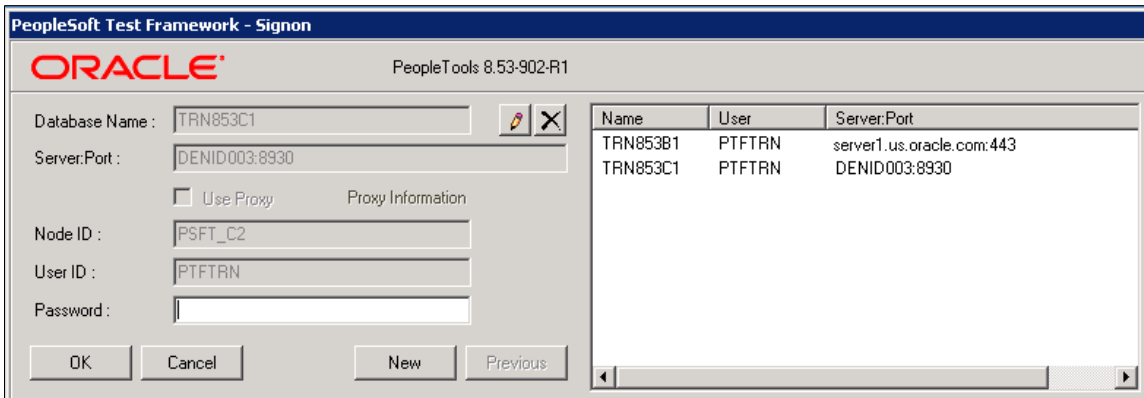


If you have created other environment signons, click the Previous button to select another environment signon.

Click the Edit button to edit the currently selected environment signon.

Image: Example of PeopleSoft Test Framework - Signon dialog box showing previously used test environments

This example illustrates the PeopleSoft Test Framework - Signon dialog box showing previously used test environments.



Environment signon settings are specific to the machine on which the PTF client is installed. The environment signon settings are stored in the environments.xml file in the PTF data directory (C:\Documents and Settings\\Application Data\PeopleSoft\PeopleSoft Test Framework) by default.

Note: The environment password is not stored in the environments.xml file.

Configuring Local Options

To configure local options, access the Local Options dialog box (from the PTF menu, select Local Options).

Local options are specific to the machine on which the PTF client is installed. The local options settings are stored in the localoptions.xml file in the PTF data directory (C:\Documents and Settings \<User>\Application Data\PeopleSoft\PeopleSoft Test Framework) by default.

Note: Changes made to the local options settings will take affect the *next* time you start the PTF client, or after a PTF test suite refresh.

Image: Local Options Dialog Box

This example illustrates the fields and controls on the Local Options dialog box. You can find definitions for the fields and controls later on this page.

The screenshot shows the 'Local Options' dialog box with the following settings:

- Process Run Options:**
 - Queued: Timeout (min.) Log Result
 - Posting: Timeout (min.) Log Result
 - Processing: Timeout (min.)
- Query Options:**
 - Output Folder
 - Output Format
- Test Options:**
 - Grid: Show Field Label Grid: Show Comment
 - Auto-Check Syntax
- Recorder Options:**
 - File.Download: prompt for path
- Kill Leftover Driver Options:**
 - On Exit Prompt vs. Kill

Buttons: OK, Cancel

Process Run Options

Queued: Timeout (min.)

Enter the time in minutes for a process to be queued before PTF logs a warning or a fail message.

Queued: Log Result	Specify whether a timeout causes PTF to log a warning or a fail message. If LogFail is selected and Stop on Error is set in the Debug menu, then execution will stop if a timeout occurs.
Posting: Timeout (min.)	Enter the time in minutes for a process to post before PTF logs a warning or a fail message.
Posting: Log Result	Specify whether a timeout causes PTF to log a warning or a fail message. If LogFail is selected and Stop on Error is set in the Debug menu, then execution will stop if a timeout occurs.
Processing: Timeout (min.)	Enter the time in minutes for a process to complete before PTF logs a warning or a fail message.

Query Options

Output Folder	Enter or browse to the path to use for query output.
Output Format	Select the file format to use for query output. Options are: <i>CSV</i> , <i>XLS</i> , or <i>XML</i> .

Test Options

Grid: Show Field Label	Select <i>Tooltip</i> to show field labels as tooltips (hover text). Select <i>Column</i> to show field labels in a column in the test window.
Grid: Show Comment	Select <i>Tooltip</i> to show step comments as tooltips (hover text).
Auto-Check Syntax	Select <i>Yes</i> to be prompted to check syntax every time you save a test.

Recorder Options

File.Download: prompt for path	Select <i>Yes</i> to be prompted to specify the path to use for file downloads when executing tests.
---------------------------------------	--

Kill Leftover Driver Options

These options specify whether to check for active web driver processes when you exit the PTF client, and how to manage them. Prior to version 8.55, PTF could execute tests only using Internet Explorer. With this release, multiple browsers are supported for test execution via use of a web driver, which is a web automation framework that enables test execution against different browsers. To assist in debugging, users have the option of leaving the test execution browser session open after test execution, using a toggle available on the PTF Debug menu. However, the web driver used for each open session consumes machine memory, until that browser session is closed. For that reason, these new options are available to provide a way to make sure the drivers and their associated browser sessions are properly managed, to prevent performance issues. There are also related options for managing the web drivers and test execution browser sessions in the PTF Debug menu and Tools menu.

On Exit	Determines if PTF should automatically check for leftover active driver sessions when terminating the client.
----------------	---

Options are:

- *Yes*: Select this option to include a check for existing active driver processes when you terminate PTF. The value selected in the Prompt vs. Kill field determines what action is taken.
- *No*: PTF will not check for leftover sessions.

You can check for active driver processes at any time by selecting Tools, Check/Kill Leftover Drivers from the PTF client menu.

Prompt vs. Kill

The value selected in this field controls how PTF manages existing active web driver sessions. Options are:

- *Prompt*: Select this option to view a dialog box that includes a count and list of the active driver processes, and a prompt asking if you want to kill (terminate) them. Choose Yes to terminate all active driver processes, No to leave them active, or Cancel to return to the Test Editor window.
- *Kill*: Select this option to automatically terminate all active driver processes. No prompt or preview of active driver processes appears when you select this option.

Configuring Execution Options in PTF Client

You use execution options to configure settings for the PeopleSoft applications that you test with PTF. Execution options are stored as part of the metadata for a PTF environment and are available to all users of that environment. Only a PTF administrator (a user with the PTF Administrator role) is able to insert, delete, or modify execution options. You can configure execution options either in the PTF client, or by using the Define Execution Options component in the PeopleSoft Internet Architecture.

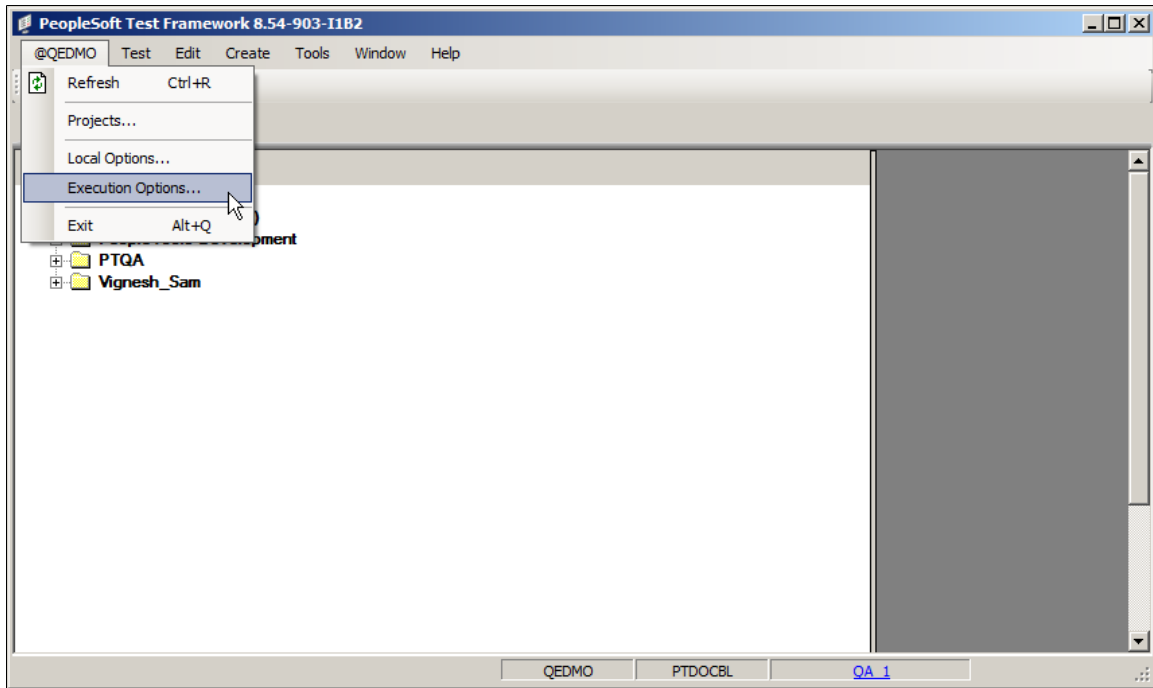
This section describes how to configure execution options in the PTF Client. For information about defining execution options in PIA, see [Configuring Execution Options in PeopleSoft Internet Architecture](#)

Note: Because test assets are PeopleTools-managed objects, we strongly recommend that you run tests only against the database on which they are stored. As part of the PTF maintenance process, PTF synchronizes test definitions with application metadata definitions. If tests are run against a different application database, you may encounter problems when an application is customized or upgraded. A PTF administrator can limit execution options to environments running against the same database where test assets are stored.

To establish execution options in the PTF client, access the Execution Options dialog (select *<PTF menu>*, Execution Options). The PTF menu is labeled with the name of the current PTF environment, preceded by an @ character, such as @QEDMO, for example. You can also access the Execution Options dialog box by clicking the execution options link in the lower right corner of the PTF application window. The execution options link is labeled with the name of the default execution option. In the following example, the execution options link is QA_1.

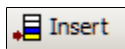
Image: Execution Options menu

This example shows how to access the Execution Options dialog in the PTF Client using the PTF menu.



The currently defined execution options appear on the left pane of the Execution Options dialog. The settings for the selected execution option appear in the right pane.

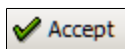
The following toolbar buttons are available:



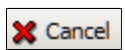
Click to add a new execution option.



Click to remove an execution option from the list.



Click to save changes and close the dialog box.



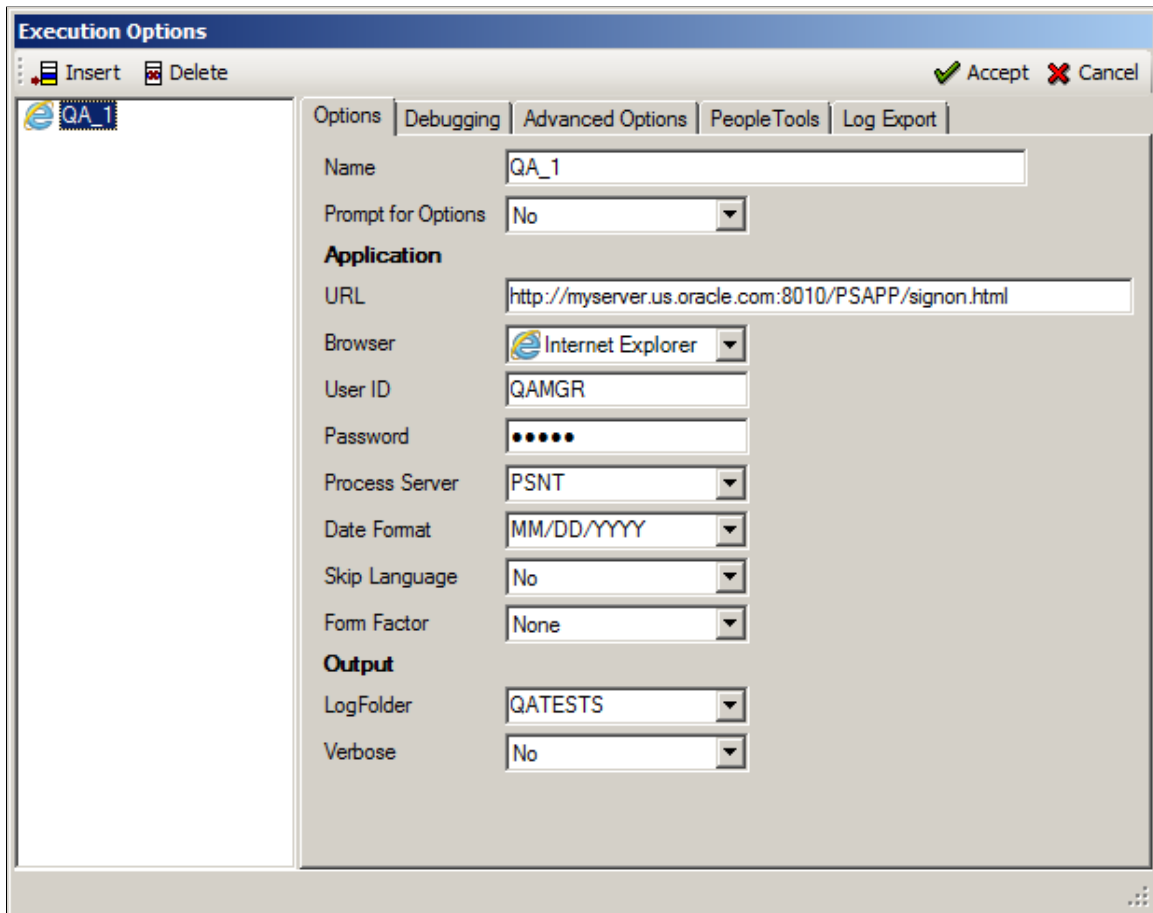
Click to close the dialog box without saving changes.

Options Tab

Use the Options tab to define the application settings and log output options to use during PTF test execution.

Image: Execution Options dialog – Options tab

This example illustrates the fields and controls on the Execution Options dialog – Options tab. You can find definitions for the fields and controls later on this page.



The following fields are on the Options tab:

- | | |
|---------------------------|--|
| Name | Enter a name for this execution option. |
| Prompt for Options | Specify whether the Execution Options dialog appears when a user executes a test. |
| URL | <p>Enter the URL of the login page for the PeopleSoft application. PeopleSoft Test Framework uses this URL for the Browser.Start_Login step type/action when executing tests and when you click the Home icon (to start the web client and go to the default URL) in the test recorder.</p> <p>If the URL that is specified is not a standard PeopleSoft login page, PeopleSoft Test Framework will try to determine the UserID and Password fields, and set their values accordingly. If that fails, PeopleSoft Test Framework will log a fatal error message. In that case you should use Browser.Start and the explicit steps required to log in, instead of using Browser.Start_Login.</p> |

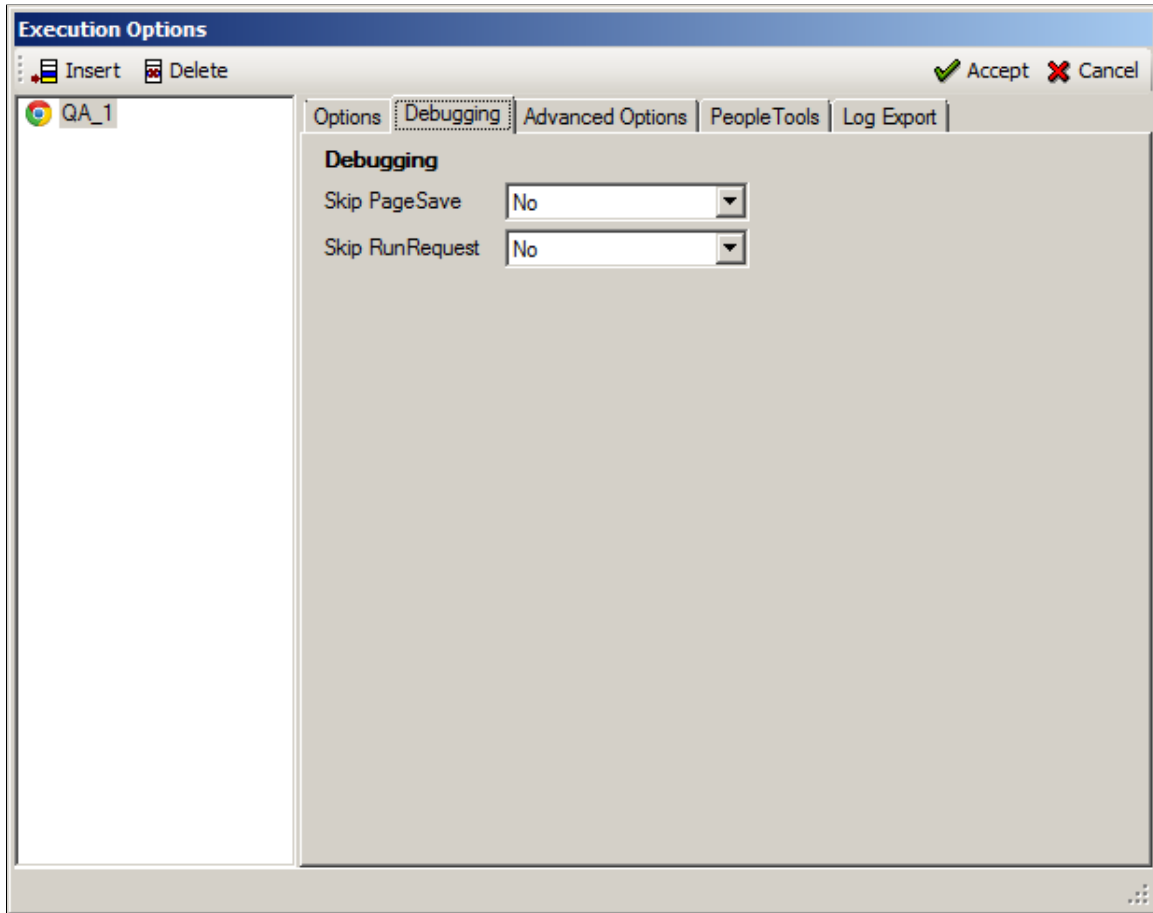
Browser	Specify the browser to use to launch the PeopleSoft application. Options are <i>Internet Explorer</i> , <i>Chrome</i> , <i>Edge</i> , or <i>Firefox</i> . If the specified browser is not present on the system, an error will appear and the test will abort.
User ID	Enter a valid user ID for the application database.
Password	Enter the login password for the user.
Process Server	Select a process server from the drop-down list. This list is populated by the Process Server List field in the Configuration Options page. See Defining PTF Configuration Options .
Date Format	Select a date format.
Skip Language	Select <i>Yes</i> to bypass the language selection on the PeopleSoft application login page.
Form Factor	Select the form factor size to use when launching the application. At test execution, the form size that you specify is selected on the login page, and a new browser instance will open using that size. The test will continue to execute in the new browser instance.
LogFolder	Select or enter the folder name to which test logs will be written. If the folder does not exist it will be created.
Verbose	Specify the log format. Select <i>Yes</i> to log a detail line for each step that is executed in the test. Select <i>No</i> to log only the test status (Pass or Fail) at the test level and to log a detail line for failed steps.

Debugging Tab

Use the Debugging tab to specify how PTF should manage page saves and run requests during test execution.

Image: Execution Options dialog – Debugging tab

This example illustrates the fields and controls on the Execution Options dialog - Debugging tab. You can find definitions for the fields and controls later on this page.



The following fields are on the Debugging tab:

Skip PageSave

Select *Yes* to prevent a test from executing a save. You would, for instance, select this option to avoid duplicate values in the application database if you plan to run a test repeatedly.

Skip RunRequest

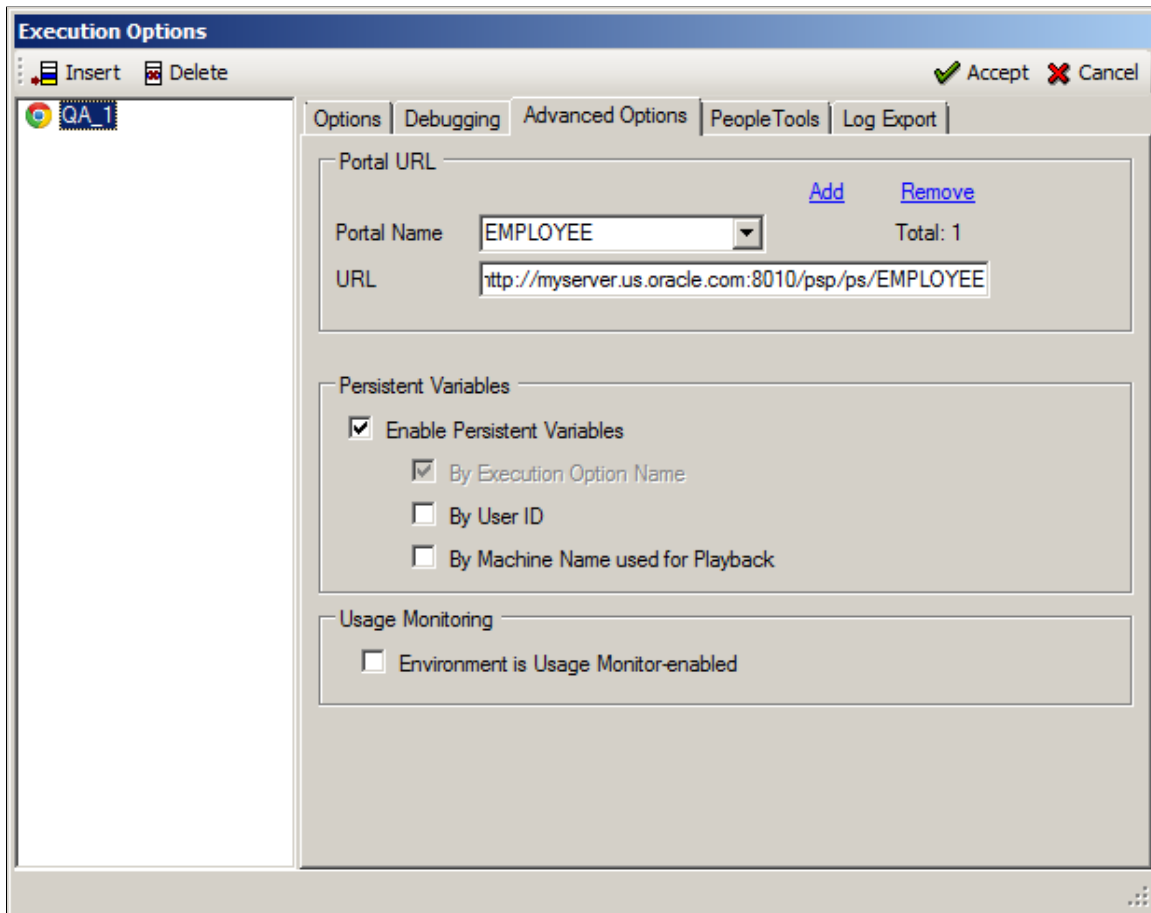
Select *Yes* to prevent the test from executing process requests.

Advanced Options Tab

Use the Advanced Options tab to specify the information required to connect to multiple portal URLs and to enable persistent variables.

Image: Execution Options dialog – Advanced Options tab

This example illustrates the fields and controls on the Execution Options dialog – Advanced Options tab. You can find definitions for the fields and controls later on this page.



The Portal URL is used to access the component when there is a step in the test to set the browser URL (Browser.Set_URL). See [Set_URL](#). To add a portal URL, click the Add link. To remove a portal URL click the Remove link.

The following fields are on the Advanced Options tab:

Add and Remove

Click to add or remove a Portal definition, then complete the Portal Name and URL fields.

Portal Name

Enter a valid portal name or select from a list of previously added portal names.

Note: The name is saved in upper case.

URL

Enter the portal URL.

The portal URL is entered in the following format:

```
http://webserver/psp/domain/portalname/node
```

For example: `http://myserver.us.oracle.com:8010/ps/ps/EMPLOYEE/QE_LOCAL/`

Note: The ending backslash / is optional

Enable Persistent Variables

Select to enable saving and using persistent variables. Selecting this option enables the other fields in this group.

See [Using Persistent Variables](#)

By Execution Option Name

This option is automatically selected when you select the Enable Persistent Variables check box. Persistent variables are stored in the database keyed by execution option name. Persistent variables can also be keyed by User ID, machine name or both.

By User ID

Select to store and retrieve persistent variables by PTF user ID.

By Machine Name Used for Playback

Select to store and retrieve persistent variables by machine name.

Environment is Usage Monitor Enabled

Select this option to enable managed object tracking with usage monitor during test execution. When this option is selected, PTF passes the test and testcase values as parameters to the usage monitor during test execution when it encounters the UsageMonitor step type.

When this option is not selected, if a test includes a Usage Monitor step type, then PeopleSoft Test Framework writes a warning to the log and skips the step. This enables you to run a test on environments with and without usage monitor enabled, without having to modify the test between executions.

Note: The PeopleSoft application must be properly configured for usage monitor if you select this option.

For more information about configuring usage monitor, see "Usage Monitor" (PeopleTools 8.55: Performance Monitor).

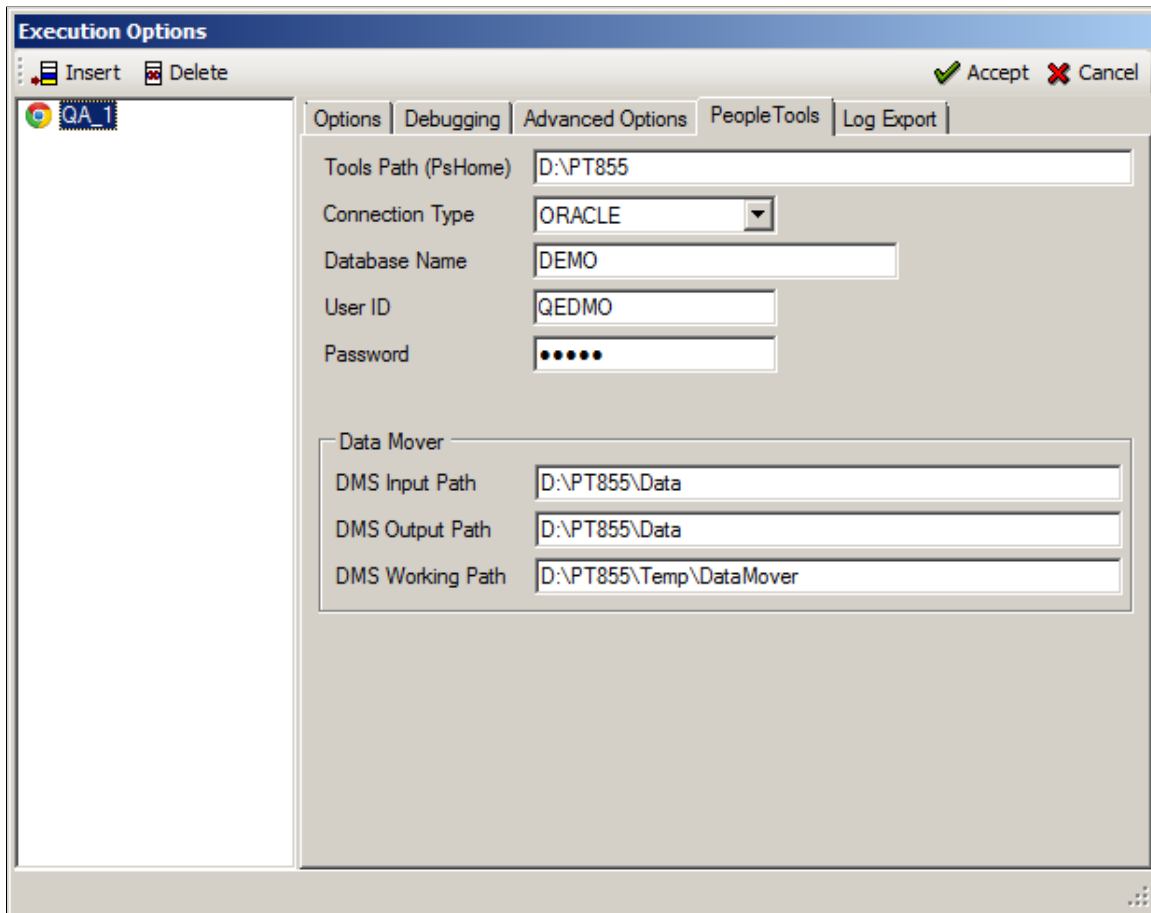
For more information about the Usage Monitor step type, see [UsageMonitor](#).

PeopleTools Tab

Use the PeopleTools tab to provide the information required to connect to DataMover.

Image: Execution Options dialog – PeopleTools tab

This example illustrates the fields and controls on the Execution Options dialog – PeopleTools tab. You can find definitions for the fields and controls later on this page.



The following fields are on the PeopleTools tab:

Tools Path (PsHome)	Enter the path to PS_HOME for this environment.
Connection Type	Select the connection type.
Database Name	Enter the name of the database for this environment.
User ID	Enter a valid database user ID.
Password	Enter the password for this user.
DMS Input Path	Enter the DataMover input path.
DMS Output Path	Enter the DataMover output path.
DMS Working Path	Enter the DataMover working path.

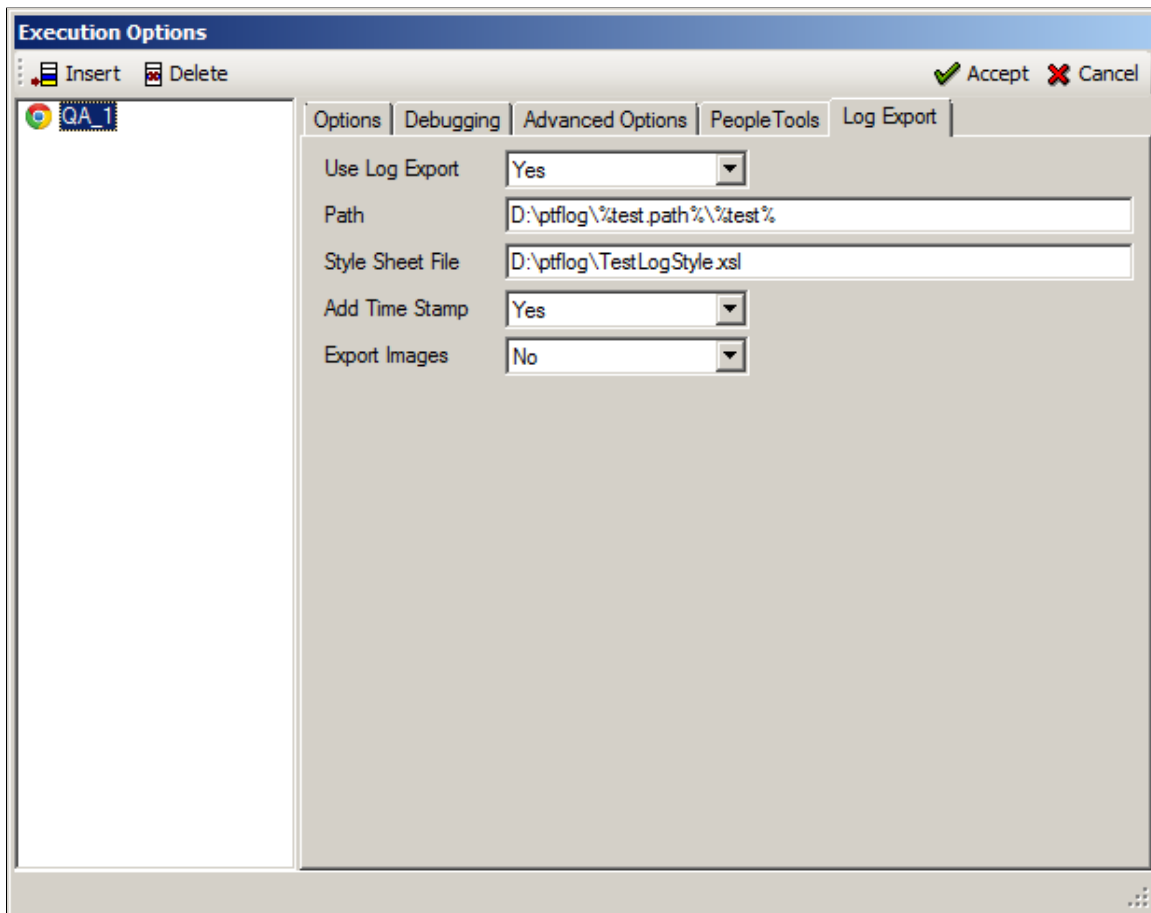
Log Export

The Log Export tab allows you to archive the result logs to a file system in XML + XSL format at the end of each test execution, which provides the following benefits:

- The logs are accessible from any browser.
- The PTF client is not required to verify test results.
- The logs are available even after the environment or database is brought down or upgraded.
- Since the log is in XML format, you can write customized utilities to parse or interpret the logs as needed.

Image: Execution Options dialog - Log Export tab

This example illustrates the fields and controls on the Execution Options dialog - Log Export tab. You can find definitions for the fields and controls later on this page.



Use Export Log

Select *Yes* to activate the export log functionality.

Path

Specify the shared drive to store the log files.

Note: Ensure that the shared drive path allows users full read/write access. The path can be customized using existing system variables. See [System Variables](#)

Style Sheet File

(Optional) Specify the path to the stylesheet. If specified, the exported XML log will be saved with the stylesheet.

You can create your own stylesheet to format the XML. If this field is left blank, the XML will not be saved with a stylesheet.

Note: Ensure that the shared drive path allows users full read/write access. The path can be customized using existing system variables. See [System Variables](#)

Add Time Stamp

If set to *Yes* the XML filename will be appended with the Time Stamp in the following format:

```
<TEST_NAME>-<LOG_ID>-T<YYYYMMDD>_<HHMMSS>
```

Example: FSCM_INS_VER-LOG5-T20121128_163649.XML

Export Images

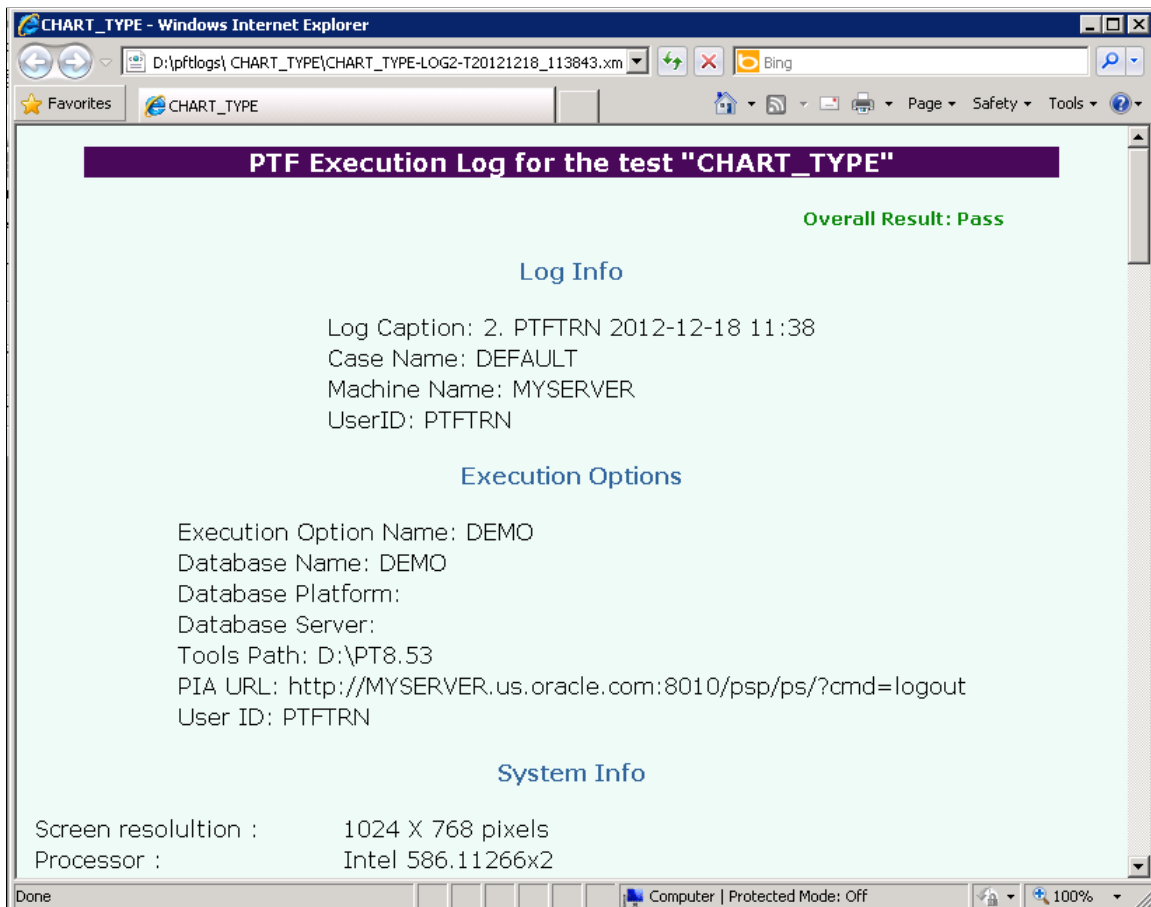
If set to *Yes* the Image will be saved in the same Log folder as the XML Log in the following format:

```
<TEST_NAME>-<LOG_ID>-T<OPTIONAL_TIME_STAMP>-<LINE_NUMBER_AS_SHOWN_IN_LOG>
```

Example: FSCM_INS_VER-LOG5-T20121128_163649-Line30.JPEG

Image: Sample export log displayed in the browser

This example illustrates a sample export log displayed in the browser.



Default Execution Option

When you click the Accept button in the Execution Options dialog box, PTF stores the name of the selected execution option and uses it, by default, in subsequent test recordings and executions. A link in the lower right corner of the PTF application window displays the name of the default execution option. You can click the link to open the Execution Options dialog box.

Overriding the Default Execution Option

Use an Execution step with a Set_Options action in a shell test to override the default execution option.

See [Execution](#).

Related Links

Use [Configuration and Execution Options](#)

Configuring Execution Options in PeopleSoft Internet Architecture

You use execution options to configure settings for the PeopleSoft applications that you test with PTF. Execution options are stored as part of the metadata for a PTF environment and are available to all users of that environment. Only a PTF administrator (a user with the PTF Administrator role) is able to insert, delete, or modify execution options. You can configure execution options either in the PTF client, or by using the Define Execution Options component in the PeopleSoft Internet Architecture.

This section describes how to define execution options using the Define Execution Options component in PIA. For information about defining execution options in the PTF client, see [Configuring Execution Options in PTF Client](#).

To configure execution options in PIA, complete the following tasks:

- Specify execution options.
- Configure debugging options.
- Define advanced options.
- Specify PeopleTools options.
- Establish Export Log options.

Specifying Execution Options

Use the Define Execution Options (PSPTTSTEXEOPTIONS) page to define PTF options for executing tests.

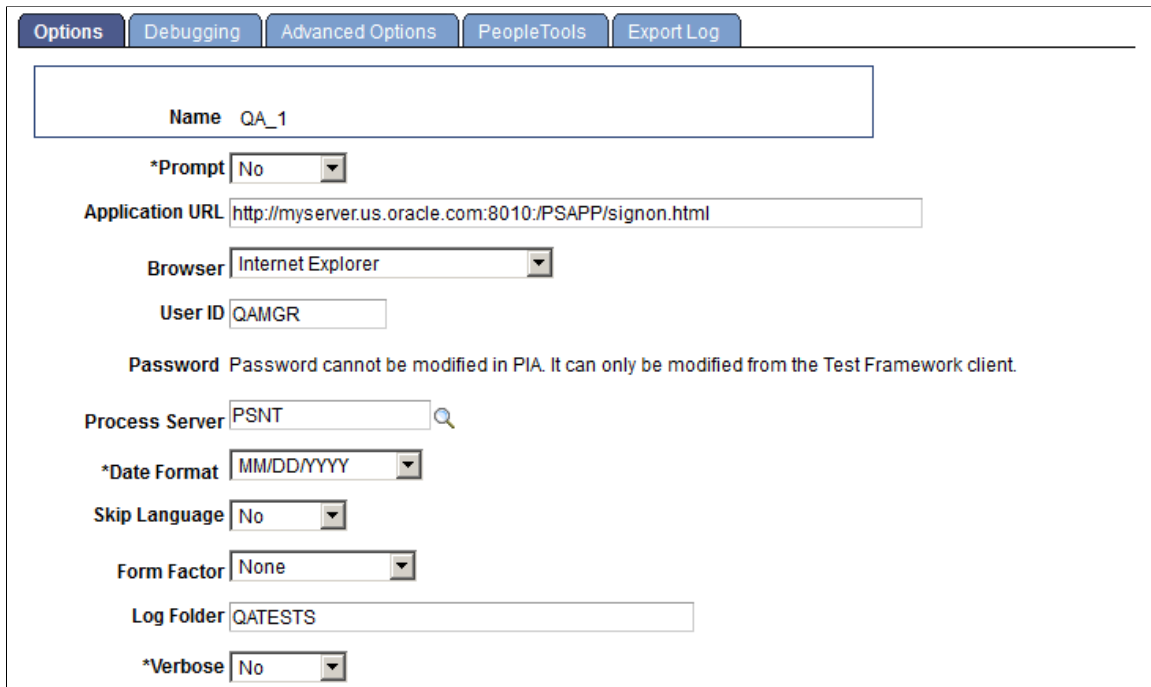
Navigation

PeopleTools, Lifecycle Tools, Test Framework, Define Execution Options

This example shows the Execution Options component - Options page in PIA:

Image: Define Execution Options Page

This example illustrates the fields and controls on the Define Execution Options page. You can find definitions for the fields and controls later on this page.



The screenshot shows the 'Define Execution Options' page with the following fields and values:

- Name:** QA_1
- *Prompt:** No
- Application URL:** http://myserver.us.oracle.com:8010:/PSAPP/signon.html
- Browser:** Internet Explorer
- User ID:** QAMGR
- Password:** Password cannot be modified in PIA. It can only be modified from the Test Framework client.
- Process Server:** PSNT
- *Date Format:** MM/DD/YYYY
- Skip Language:** No
- Form Factor:** None
- Log Folder:** QATESTS
- *Verbose:** No

The fields on the Define Execution Options page are the same as the fields on the Options tab in the PTF client, with the exception that you cannot modify passwords on the PeopleSoft application component.

Prompt

Specify whether the Execution Options dialog appears when a user executes a test.

Application URL

Enter the URL of the login page for the PeopleSoft application. PeopleSoft Test Framework uses this URL for the Browser. Start_Login step type/action when executing tests and when you click the Home icon (to start the web client and go to the default URL) in the test recorder.

Browser

Specify the browser to use to launch the PeopleSoft application. Options are *Internet Explorer*, *Chrome*, *Edge*, or *Firefox*. If the specified browser is not present on the system, an error will appear and the test will abort.

User ID

Enter a valid user ID for the application database.

Password

This field is unavailable for entry. The password must be specified using the Execution Options dialog in the PTF client. For more information, see [Configuring Execution Options in PTF Client](#).

Process Server

Select a process server from the drop-down list. This list is populated by the Process Server List field in the Configuration Options page.

Date Format	Select a date format.
Skip Language	Select <i>Yes</i> to bypass the language selection on the PeopleSoft application login page.
Form Factor	Select the form factor size to use when launching the application.
Log Folder	Select or enter the folder name to which test logs will be written. If the folder does not exist it will be created.
Verbose	Specify the log format. Select <i>Yes</i> to log a detail line for each step that is executed in the test. Select <i>No</i> to log only the test status (Pass or Fail) at the test level and to log a detail line for failed steps.

See [Configuring Execution Options in PTF Client](#).

Configuring Debugging Options

Use the Define Execution Options – Debugging (PSPTTSTDEBOPTIONS) page to define PTF configuration options.

Navigation

PeopleTools, Lifecycle Tools, Test Framework, Define Execution Options

Select the Debugging Options tab.

Image: Define Execution Options-Debugging page

This example illustrates the fields and controls on the Define Execution Options – Debugging page. You can find definitions for the fields and controls later on this page.

The screenshot shows the 'Define Execution Options-Debugging' page. At the top, there are five tabs: 'Options', 'Debugging' (which is selected), 'Advanced Options', 'PeopleTools', and 'Export Log'. Below the tabs, there is a large text input field containing 'Name QA_1'. Underneath this field are two dropdown menus. The first is labeled '*Skip Save' and is set to 'No'. The second is labeled '*Skip Run Request' and is also set to 'No'.

Skip Save

Select *Yes* to prevent a test from executing a save. You would, for instance, select this option to avoid duplicate values in the application database if you plan to run a test repeatedly.

Skip Run Request

Select *Yes* to prevent the test from executing process requests.

Defining Advanced Options

Use the Define Execution Options - Advanced Options page (PSPTTSTADVOPTIONS) to define multiple portal URLs and to enable persistent variables.

PTF uses the portal URL to access the component when there is a step in the test to set the browser URL (Browser.Set_URL). See [Set_URL](#). To add a portal URL, click the Add icon. To remove a portal URL click the Delete icon.

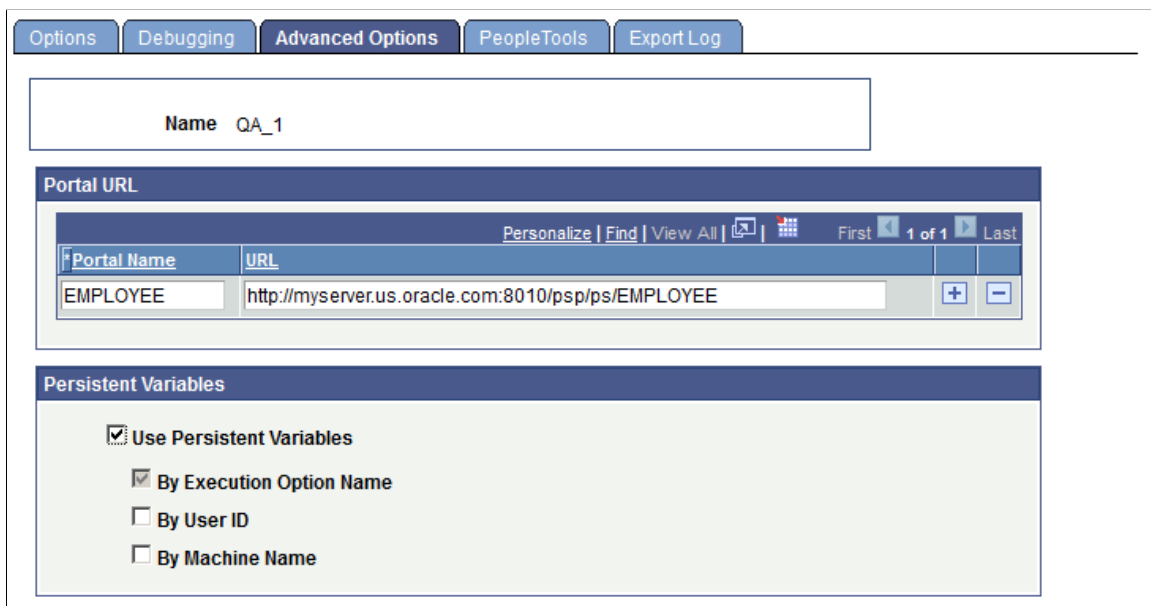
Navigation

PeopleTools, Lifecycle Tools, Test Framework, Define Execution options

Select the Advanced Options tab.

Image: Define Execution Options - Advanced Options page

This example illustrates the fields and controls on the Define Execution Options - Advanced Options page. You can find definitions for the fields and controls later on this page.



The screenshot shows the 'Define Execution Options - Advanced Options' page. At the top, there are navigation tabs: 'Options', 'Debugging', 'Advanced Options', 'PeopleTools', and 'Export Log'. The 'Advanced Options' tab is active. Below the tabs, there is a text input field labeled 'Name' containing the text 'QA_1'. Below this is a section titled 'Portal URL' which contains a table with the following data:

Portal Name	URL
EMPLOYEE	http://myserver.us.oracle.com:8010/ps/ps/EMPLOYEE

Below the table are '+', '-', and 'X' icons for adding, deleting, and refreshing rows. Below the 'Portal URL' section is a 'Persistent Variables' section with the following options:

- Use Persistent Variables
 - By Execution Option Name
 - By User ID
 - By Machine Name

Portal Name

Enter a portal name.

Note: The name is saved in upper case.

URL

Enter the portal URL.

The portal URL is entered in the following format:

```
http://webserver/ps/domain/portalname/node
```

For example: `http://myserver.us.oracle.com:8010/ps/ps/EMPLOYEE/QE_LOCAL/`

Note: The ending backslash / is optional

Use Persistent Variables

Select to enable saving and using persistent variables. Selecting this option enables the other fields in this group.

By Execution Option Name

This option is automatically selected when you select the Use Persistent Variables check box. Persistent variables are stored in the database keyed by execution option name. Persistent variables can also be keyed by User ID, machine name or both.

By User ID

Select to store and retrieve persistent variables by PTF user ID.

By Machine Name

Select to store and retrieve persistent variables by machine name.

Specifying PeopleTools Options

Use the PeopleTools page (PSPTTSTTOOLOPTIONS) to supply the information required to connect to DataMover.

Navigation

PeopleTools, Lifecycle Tools, Test Framework, Define Execution options

Select the PeopleTools tab.

This example shows the Define Execution Options - PeopleTools page:

Image: Define Execution Options - PeopleTools page

This example illustrates the fields and controls on the Define Execution Options - PeopleTools page. You can find definitions for the fields and controls later on this page.

The fields on the PeopleTools page are the same as the fields on the PeopleTools tab in the PTF client, with the exception that you cannot modify passwords on the PeopleSoft application component.

- Tools Path** Enter the path to PS_HOME for this environment.
- Connection Type** Select the connection type.
- Database Name** Enter the name of the database for this environment.
- User ID** Enter a valid database user ID.
- Password** This field is unavailable for entry. The password must be specified using the PTF Client.
- DMS Input Path** Enter the DataMover input path.
- DMS Output Path** Enter the DataMover output path.
- DMS Working Path** Enter the DataMover working path.

Establish Export Log Options

Use the Define Execution Options - Export Log page (PSPTTSTLOGOPTIONS) to automatically archive the result logs to a file system in XML + XSL format at the completion of each test.

This option provides the following benefits:

- The logs are accessible from any browser.

- The PTF client is not required to verify test results.
- The logs are available even after the environment or database is brought down or upgraded.
- Since the log is in XML format, you can write customized utilities to parse or interpret the logs as needed.

Navigation

PeopleTools, Lifecycle Tools, Test Framework, Define Execution options

Select the Export Log tab.

Image: Define Execution Options - Export Log page

This example illustrates the fields and controls on the Define Execution Options - Export Log page. You can find definitions for the fields and controls later on this page.

Use Log Export

Select *Yes* to activate the export log functionality.

Path

Specify the shared drive to store the log files.

Note: Ensure that the shared drive path allows users full read/write access. The path can be customized using existing system variables. See [System Variables](#)

Style Sheet File

(Optional) Specify the path to the stylesheet. If specified, the exported XML log will be saved with the stylesheet.

You can create your own stylesheet to format the XML. If this field is left blank, the XML will not be saved with a stylesheet.

Add Time Stamp

If set to *Yes* the XML filename will be appended with the Time Stamp in the following format:

```
<TEST_NAME>-<LOG_ID>-T<YYYYMMDD>_
<HHMMSS>
```

Example: FSCM_INS_VER-LOG5-T20121128_163649.XML

Export Images

If set to *Yes* the Image will be saved in the same Log folder as the XML Log in the following format:

<TEST_NAME>-<LOG_ID>-T<OPTIONAL_TIME_Stamp>-<LINE_NUMBER_AS_SHOWN_IN_LOG>

Example: FSCM_INS_VER-LOG5-T20121128_163649-Line30.JPEG

Chapter 3

Using PeopleSoft Test Framework

Using PTF Explorer

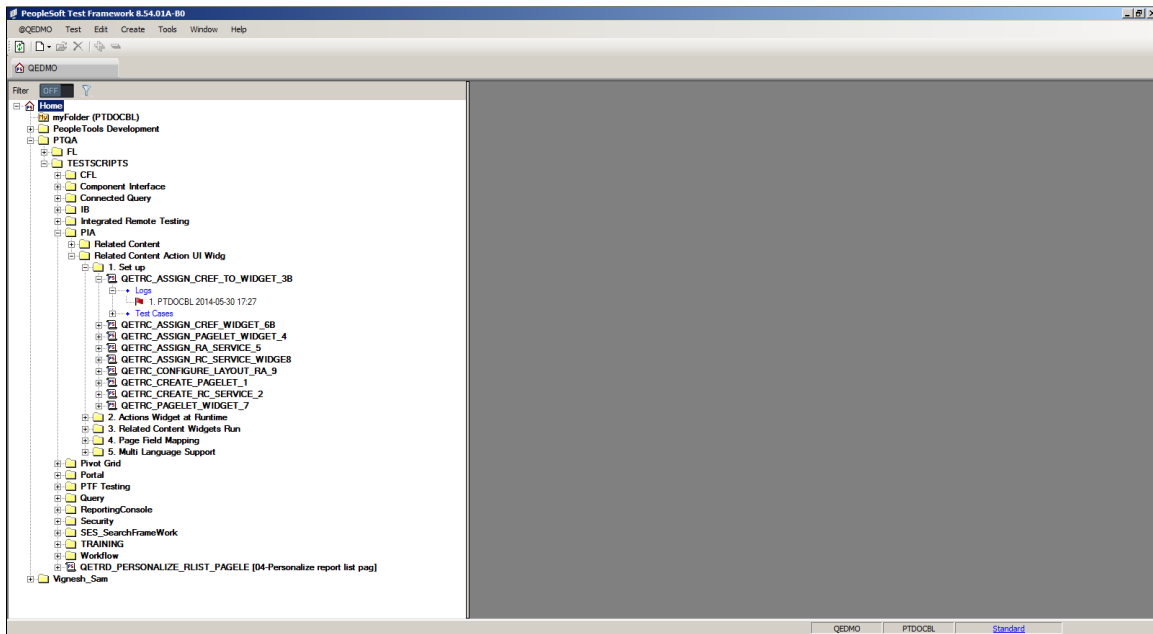
This section provides an overview of PTF Explorer and describes how to define and apply filters.

Understanding PTF Explorer

PTF Explorer gives you access to the PTF test assets (tests, test cases, libraries, and logs) stored within an application database. Assets appear in a tree structure with collapsible folders for organizing test assets. After PTF has been installed and configured, PTF Explorer is the first pane that appears when you start the client. It is labeled with the name of the PTF environment, QEDMO in this example:

Image: Example of the PTF Explorer User Interface

This example illustrates the PTF Explorer user interface.



You use PTF Explorer to:

- Create tests and folders.
- Delete tests and folders.
- Copy and move tests.
- Run a test.
- Rename tests and test cases.

- Navigate to and open test assets.
- Associate tests with Application Designer projects.

Using myFolder

The PTF Explorer tree contains a folder called myFolder. You can use myFolder to store tests that you do not want to share with other users. Users with the PTF User role can create, edit, and delete tests *only* in myFolder.

PTF Explorer Menus

This section describes the menus that appear when PTF Explorer has focus. Note that many menu commands are specific to the currently selected item.

This table describes the PTF Explorer PTF menu commands:

Note: The PTF Explorer PTF Menu name corresponds to the name of the current PTF environment preceded with an @ character. In the previous example, the PTF Explorer menu name is @QEDMO.

<i>PTF Menu Command (@<PTF_Environment>)</i>	<i>Usage</i>
Refresh	Refreshes the current view.
Projects	Opens the Projects dialog box.
Local Options	Opens the Local Options dialog box.
Execution Options	Opens the Execution Options dialog box.

This table describes the PTF Explorer Test menu commands:

<i>Test Menu Command</i>	<i>Usage</i>
Open	Opens the selected test or test case.
Delete	Deletes the selected test.
Rename	Renames the selected test, test case, or folder.
Refresh Selection	Refreshes the view for the selected test.
Expand Selection	Expands all the branches in the selected node.
Collapse Selection	Collapses all the branches in the selected node.

The PTF Explorer Edit menu contains standard Microsoft edit commands, such as Cut, Copy, and Paste, and the following menu command.

Edit Menu Command	Usage
Copy Link to Clipboard	Copies the link for the selected test, test case, or log to the clipboard. You can use this information in conjunction with the Quick Open command.

Use the PTF Explorer Create menu to create folders and tests.

This table describes the Create menu commands:

Create Menu Command	Usage
Folder	Creates a new folder within the selected folder.
Test	Creates a new test in the selected folder.
Shell Test	Creates a shell test in the selected folder.

This table describes the PTF Explorer Tools menu commands:

Tools Menu Command	Usage
Message	Opens the Message tool, which enables you to monitor test execution. The Message tool displays details about the current step, including name, object type, and value.
Log Manager	Opens the Log Manager tool, which enables you to delete logs from tests.
Mass Update	Opens the Mass Update tool, which enables you to modify steps from multiple tests based on specific criteria.
Tests PT Upgrade	Opens the Tests PT Upgrade dialog box, which enables you to upgrade tests from a previous version of PeopleTools.
Check/Kill Leftover Drivers	Closes all open test execution windows to free up memory used by web drivers.

This table describes the PTF Explorer Window menu commands:

Window Menu Command	Usage
Quick Open	<p>Opens a test, test case or a text execution log with data that was copied to the clipboard using the Copy Link to Clipboard command.</p> <p>Using the Copy Link to Clipboard command and the Quick Open command together enables users to easily share tests without having to navigate to the test in PTF Explorer. You can select a test and select Edit, Copy Link to Clipboard. Then, you paste that data into a text message and send it to another user, who can then copy and paste the data into the Quick Open dialog box and open the test.</p> <p>You can also use this menu command to search for and open multiple tests, test cases, or text execution logs.</p>
Close All	Closes all windows.

Multi-Select Functionality

Based on the item or items selected in the tree view, when you right-click a list of valid actions displays. The tree view supports multiple selections. The following actions are supported with multi-select:

- Open tests and logs.
- Cut and paste folders and tests.
- Delete tests and folders.
- Expand and collapse tests and folders.

The valid actions are based on the PTF role associated with the logged on user.

Defining and Applying Filters

You can create and apply filters to the PTF Explorer tree, to view subsets of test assets based on specific criteria. This enables you to view only tests with certain characteristics, or that belong to certain categories, such as: tests with similar names; tests belonging to the same parent folder; tests last updated by the same user; tests resulting from the same Test Maintenance Report run. When the filter is applied, if a folder does not contain any tests that meet the filter criteria, then that folder is hidden.

Note: The tree view does not refresh automatically with every edit or action on the Explorer Tree. If you create a test which does not satisfy the active filter requirements, the folder for that test will not automatically be hidden from the tree; the folder remains visible until you update the filter or refresh the tree view.

Filter Action Icons

Use the following filter action icons, which appear above the PTF Explorer tree, to define, activate, or clear a filter:



Click to open the Tree Filters dialog, where you can define filter criteria.



Indicates that there is no active filter. This icon functions as a toggle switch; click to activate the filter.



Indicates that a filter is currently active. This icon functions as a toggle switch; click to clear the filter.

Filter Definition

To define a filter, click the Set Filter icon to open the Tree Filters dialog box, and specify the filter criteria.

Image: Tree Filter Dialog Box

This example illustrates the fields and controls in the Tree Filter dialog box.

The filter criteria fields are grouped by these filter categories:

- Test filters: Includes these test properties: Test Name, Test Description, Updated By, Update Date/Time, Version.
- Step filters: Includes these step properties: Recognition, Component, Page, Record, Field.
- Maintenance report filters: Includes tests flagged by the test maintenance report by these properties: UserID, Project Name, Seq (sequence).
- Coverage report filters: Includes tests flagged by the test coverage report by these properties: UserID, Project Name.
- Others filters: Includes test by their associated Application Designer Project Name.

Filters specifying a comparison against a text value use a *starts with* condition (when no wildcards are used) or a *Like* condition (when wildcards are used). For example, the following filter field entries would return the values listed:

Filter Criteria	Returns	Does Not Return
DAY	DAY, DAYS, DAY_ONE	MONDAY
DAY%	DAY, DAYS, DAY_ONE	MONDAY
%DAY	DAY, MONDAY	DAYS, DAY_ONE
%DAY%	DAY, DAYS, DAY_ONE, MONDAY	

The supported standard wildcard characters are:

Wildcard	Search Action
% (percent symbol)	Match one or more characters.
_ (underscore)	Match any single character.
\ (backslash)	Escape character; do not treat the next character as a wildcard.

Filters specifying a comparison against a Date/Time, Version, or Seq (sequence) use a *greater than or equal to* condition.

When multiple filters are specified, the filter clauses are all conjoined with ANDs (rather than ORs).

Using the Test Editor

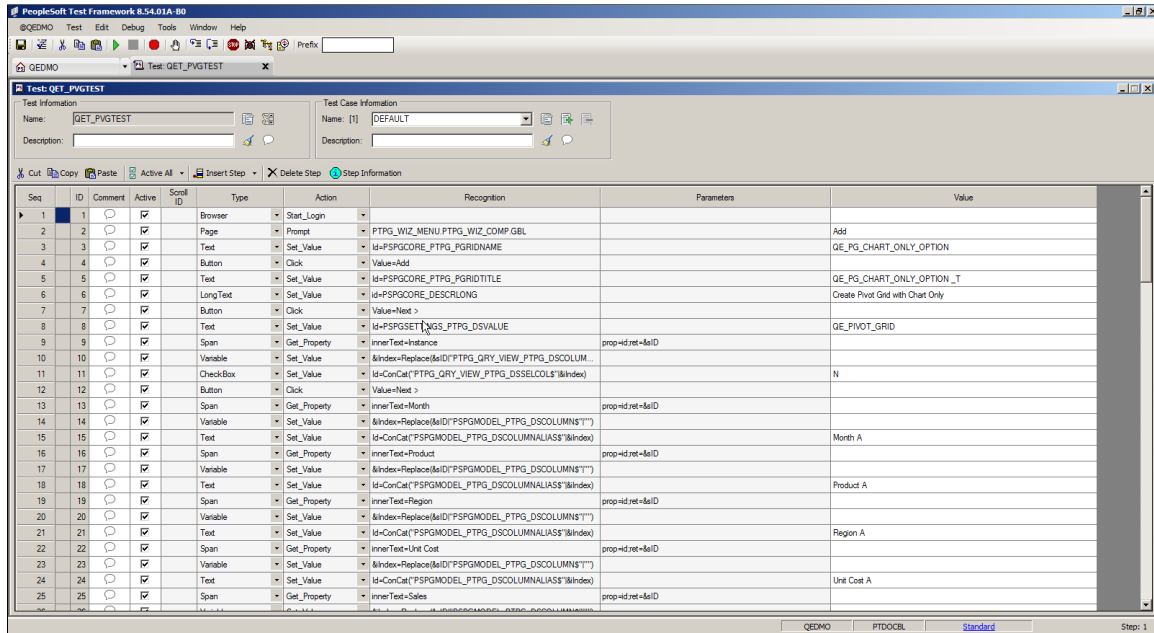
When you create or open a test, test case, or shell test, it opens in the Test Editor. The Test Editor enables you to:

- Record and edit test steps.
- Add, copy, and delete test steps.
- Create and edit test cases.
- View both test and test case in a single view.
- Debug tests.

This example shows the PTF Test Editor:

Image: PTF Test Editor

This example illustrates the PTF Test Editor. You can find definitions for the menus and controls later on in this section.



Test Editor Menus

The following menus appear when the Test Editor has focus. Note that many menu commands are specific to the currently selected step.

This table describes the Test Editor PTF menu commands:

<i>PTF Menu Commands</i>	<i>Usage</i>
Projects	Opens the Projects dialog box.
Local Options	Open the Local Options dialog.
Execution Options	Open the Execution Options dialog.

This table describes the Test Editor Test menu commands:

<i>Test Menu Commands</i>	<i>Usage</i>
Save	Saves the current test.
Save As	Creates a copy of the current test with a new name.
Test Case Save As	Creates a test case as a copy of the current test case.
Export	Export test cases and test case values to a character-delimited text file, such as comma-separated values (csv) file.

Test Menu Commands	Usage
Import	Import test cases and test case values from a character-delimited text file, such as comma-separated values (csv) file.
Copy Link to Clipboard	Copies a link to the test to the clipboard. You can send this information to another user, who can use the Window, Quick Open feature to open the test without having to navigate to it in PTF Explorer.
Validate Test	Validate the test against the application metadata to identify changes in the metadata since the test was last modified. As you modify a test based on the results of a test maintenance report, you can validate the test by running a test maintenance report. The report is generated for a single test, using analyses of compare reports generated in Step 2 of the Test Maintenance Wizard. See Creating Test Maintenance Reports .
Check Syntax	Validates the parameter syntax and values. See PTF Test Language
Run	Runs the current test.
Run from Step	Runs the current test from the specified line number.
Pause	Pauses execution of the test.
End	Stops execution of the test.
Open Test Recorder	Launches the Test Recorder.

The Test Editor Edit menu contains standard Microsoft edit commands, such as Cut, Copy, Paste, and Delete, and the following PTF commands:

This table lists the additional PTF commands:

Edit Menu Command	Usage
Find	Finds occurrences of a specific text string in the current test. Select the List all Matching Steps check box to create a list of matches.
Again	Searches for the Find string again.

This table describes Test Editor Debug menu commands:

Debug Menu Command	Usage
Step Into	Executes the current step of the test and advances to the next step.

<i>Debug Menu Command</i>	<i>Usage</i>
Step Over	Executes the current step of the test and advances to the next step, unless the next step calls another test. Steps over called tests.
Toggle Break	Sets a break point at the selected step or removes an existing breakpoint.
Clear All Breaks	Removes all break points.
Stop on Error	Stops execution if the test encounters an error.
Disable Screen Shots	By default, the recorder creates a screen shot with each error. Select this option to save space in the log by not creating screen shots.
Highlight Errors	Highlights errors in the log in yellow.
Overwrite Log	If a log window is open after the most recent execution of a test, you can choose whether to append to the open log or overwrite the open log. Select this option to overwrite the open execution log.
Leave Browser Open	Keeps the PTF-launched browser window open after executing a test, to assist with debugging. Be aware that if many tests are executed with this option enabled, performance may degrade as the webdrivers used by PTF consume memory. Use the Kill Leftover Drivers option if performance is impacted.
Kill Leftover Drivers	Closes open Test Framework browser windows, to release the memory that is used by their associated webdrivers. When the Leave Browser Open option is enabled, if performance issues arise, you can use Kill Leftover Drivers at any time to free up memory.

This table discusses the Test Editor Window menu commands:

<i>Window Menu Command</i>	<i>Usage</i>
Quick Open	Using information from the Copy Link to Clipboard command, quickly opens a test without having to navigate to the log in PTF Explorer. You can also use this menu command to search for and open multiple tests, test cases, or text execution logs.
Close All	Closes all windows.

Test Editor Toolbar

In addition to buttons for the standard Microsoft cut, copy, and paste commands, the test editor toolbar provides the following functions:



Check Syntax



Run test.



End test.



Show Test Recorder.



Debug break.



Step into.



Step over.



Stop on error.



Disable screen shots.



Highlight errors in the execution log.



Overwrite the open execution log.

Prefix

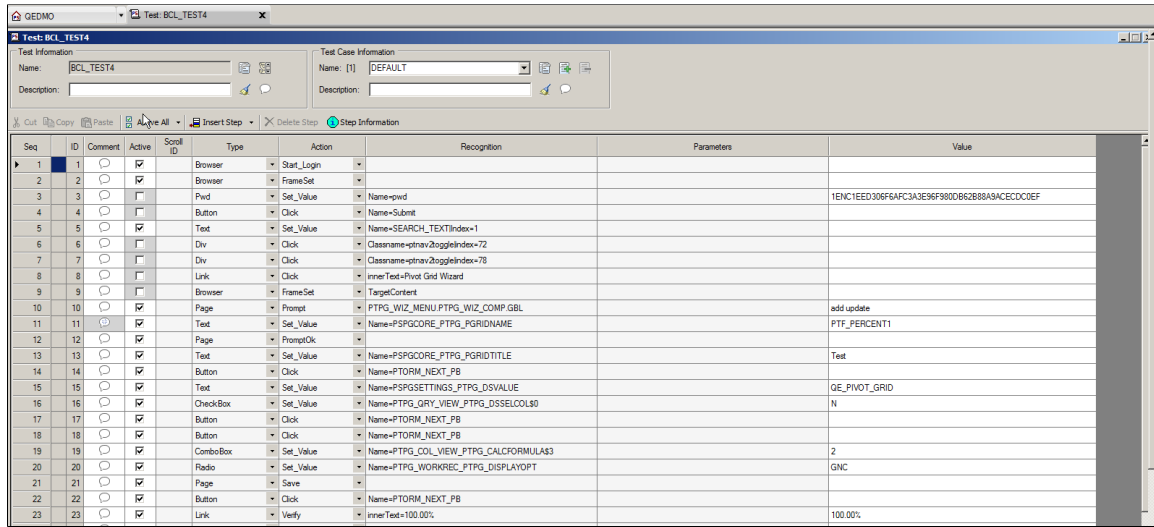
Specify text that will be added to text fields when the test is executed. The prefix text is substituted for the **#PREFIX#** reserved word in the Value field. Using a prefix can help prevent an error caused by a duplicate entry when the page is saved.

Test Window

You can have multiple tests open in PTF. Each test has its own test window. This example shows a Test Editor test window:

Image: Example of a Test Editor Test Window

This example illustrates the fields and controls on the Test Editor test window. You can find definitions for the fields and controls later on this page.



Test Window Fields

The following fields appear in the test window:

Test Information Fields

- Name** Displays the test name. This field is display-only.
- Description** Brief description of the test. You can enter a detailed description in the Comments.

Test Information Functions



Test Save As.



Click to access the Message Recognition dialog box.

Use the Message Recognition dialog box to define how PTF will respond to messages that are encountered during execution of the test. The Message Recognition icon contains a check mark when the message recognition is enabled for the test.



Open the Test Properties dialog. In the Language field, select the language for the PeopleSoft application. The default is English.

Changing the value in the Language field only affects the language the test selects at sign in. It does not enable the test to execute against a different language. PTF tests should be executed against the same language in which they were recorded.

Select the Variable Action to use with this test. The variable action determines whether or not to use persistent variables. By default the action is set to None and persistent variables are not used. See [Using Persistent Variables](#)

Select the Library Test checkbox to make the test a library test.

See [Using Library Tests](#).



Open the Test Comments dialog. You can enter a long description of the test using Rich Text and add images. Click the take Snapshot icon in the Comment dialogue to insert a screenshot.

Test Case Information Fields

Name

Displays the name of the current test case. You can select a different test case using the drop-down list. When you create a test, the system automatically associates it with a test case named DEFAULT.

Description

Brief description of the test case. You can enter a detailed description in the comments.

Test Case Information Functions



Test Case Save As.



Create a new test case.



Delete the selected test case.



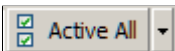
Open the Test Case Properties dialog.



Open the Test Case Comments dialog. You can enter a long description of the test case using Rich Text and images.

Test Window Toolbar

In addition to buttons for the standard Microsoft cut, copy, and paste commands, the test window toolbar provides the following functions:



Selects the Active check box for all steps. Click the drop-down arrow for additional step activation options, including:

- Active Selected Steps.

Selects the Active check box for only selected steps.

- Inactive Selected Steps.

Deselects the Active check box for only selected steps.

- Inactive All.

Deselects the Active check box for all steps.



Inserts a new, blank step below the current step. Click the drop-down arrow to insert the new step above the current step.



Deletes the current step.



Displays the Step Information dialog box showing the PeopleTools metadata associated with the object referenced in the step.

Test Step Fields

A PTF test consists of a series of steps. Each step in a test is composed of the following fields:

Seq (sequence)

A system-generated sequence number. Test steps execute according to Seq order. When you move, add, or delete a step, Seq is refreshed.

ID

A system-generated unique identifier for each step in a test. This value does not change when you move, add, or delete a step.

Test maintenance reports use the ID value.

Comment

When you click on the comment icon in the comment field a rich text editor opens that enables you to add detailed comments for each step. The comment field is gray when it contains a comment.

Active

Deselect this field to inactivate a step. PTF will skip inactive steps when the test runs. PTF syntax check is also skipped on inactive steps. Each step is active by default. This field is grayed for inactive steps.

Scroll ID

This field is only required for scroll handling.

Type

The step type. Often, the step type corresponds to the type of application object the step is to take an action on or to validate, such as Text, Check box, Browser, and so on. Other step types perform certain functions, such as executing a test or query, setting a variable, or performing conditional processing.

Action	The action the test is to take . Each step type has a set of associated actions. The two most common actions used with a Text step type, for example, are <i>Set</i> and <i>Verify</i> .
Recognition	The means that PTF uses to identify the HTML object within the application. Often, this is the HTML ID property.
Field Label	<p>Contains the label text of the page control referenced in the Recognition field.</p> <p>The value for the Show Field Label option in the Local Options dialog box determines whether this column appears:</p> <ul style="list-style-type: none"> • If Column is selected, the Field Label column appears. • If Tooltip is selected, the column does not appear; instead, the field label appears in a tooltip when you position the mouse cursor over the Value field. <p>For more information about setting local options, see Configuring Local Options.</p>
Parameters	<p>The conditions that apply to this specific step, if applicable.</p> <hr/> <p>Note: This field was introduced in release 8.54; in previous releases, parameters were specified as part of the Recognition field. When you load and save a test from release 8.53 or lower, PTF automatically moves any parameters from the Recognition field to the Parameters field.</p> <hr/>
Value	<p>In a typical recorded step, this is the value the tester entered for an object.</p> <p>In a step recorded in Verify mode, this would be the value that was present in the object when it was checked.</p> <p>Value is part of the test case, not the test itself.</p>
Related Links	<p>PTF Test Language</p> <p>Creating Tests</p>

Using the PTF Test Recorder

You use the PTF Test Recorder to record the steps in a test. When you record a test, PTF monitors each action you perform in the target application and creates a corresponding step in the test. As you are recording, you can also add additional steps, such as Verify, Log, and Conditional, that do not directly correspond to actions performed in the target application.

This section discusses:

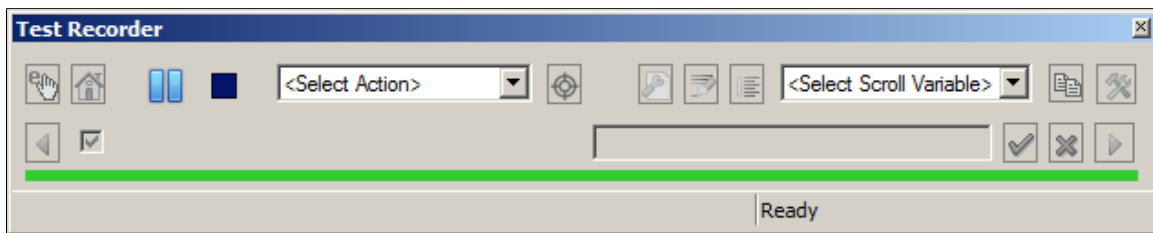
- The Test Recorder toolbar.
- Recording action tools.
- Step insert tools.
- Step modification tools.
- Step utility tools.

Test Recorder Toolbar

To access the Test Recorder toolbar, select Test, Open Test Recorder or click the Show Test Recorder icon.

Image: Test Recorder Toolbar

This example illustrates the tools and controls on the Test Recorder toolbar. You can find definitions for the tools and controls later on in this section.



Note: To move the Test Recorder toolbar, click in the title bar to the left of the close icon and drag to the new location.

The Test Recorder toolbar elements can be categorized into these functional areas:

- Recording action tools.
- Step insert tools.
- Step modification tools.
- Recorder utility tools.
- Tool hints bar.
- Status bar.

Recording Action Tools

Use these tools to control recording actions: start the web client; select the browser; and start, stop, or pause recording.

This list describes the recording action tools that are available in the Recorder Toolbar.



Hook a browser. Drag and drop this icon onto an active PeopleSoft browser session to hook the recorder to that browser. PTF will only hook a browser that was originally launched

by PTF. You can drag the hook icon to any portion of the PIA application. No page objects will be highlighted.



Launch a PeopleSoft application in a browser window using the URL of the default execution option and hook the recorder to that browser.

See [Configuring Execution Options in PeopleSoft Internet Architecture](#) and [Configuring Execution Options in PTF Client](#).



Begin recording or resume recording. When you begin recording, the recorder adds or inserts steps following the current step. When you resume recording after pausing you are prompted to choose from the following options:

- Insert new steps at the current step (do not delete subsequent steps in the test)
- Insert new steps at the current step (delete subsequent steps in the test)
- Insert new rows at the end of the test



Stop recording and write recorded steps to the test.



Pause recording.

Step Insert Tools

Use the items in the <Select Action> list box to insert action steps while recording. These items are grouped into these categories:

- Drag and drop actions.
- Step insert actions.
- Usage monitor actions.

<Select Action> - Drag and Drop Actions

These recording action tools capture page element recognition data using a method called “drag and drop”, which refers to using the mouse to click an item and drag it then drop it to another location. To use these options while recording:

1. Select a drag and drop option from the <Select Action> list.

This activates the Target icon, making it available to use.

2. Click and drag the Target icon to the desired page element.

Notice that the mouse cursor changes to a bold question mark.

- Highlight the page element, then release the mouse button to drop the icon over it.

As you hover over a page element, the page element becomes highlighted. The element that you intend to reference must be highlighted when you drop the icon over it. If the field is not highlighted, PTF will not recognize it.

Drag and drop actions are not applicable to all HTML objects.

This list describes the drag and drop action options that are available in the Recorder Toolbar <Select Action> list:

ComboBox.ValueExists

Adds a ComboBox.ValueExists step. Drag and drop over a combobox element. You are prompted to select the value, and the variable to store the return value, and to select an expected value. Valid values are True, False, or Ignore.

Div.ScrollIt

Adds a Div.ScrollIt step. Drag and drop over a horizontal/vertical scrollbar element.

Field Exists

Adds an Exists step. Drag and drop over a page element. You are prompted to enter a variable name or select an existing variable and to select an expected value. Valid values are True, False, or Ignore.

See [Exists](#).

Get Label

Adds a Field.GetLabel step. You are prompted for the return variable that will contain the label.

See [GetLabel](#)

Get Property

Adds a GetProperty step. Drag and drop over a page element. You are prompted with a list of properties for the selected page element. Enter a variable name or select an existing variable name for one or more properties. The recorder adds a GetProperty step for each property you identify.

See [Get_Property](#).

Get Style

Adds a GetStyle step. Drag and drop over a page element. You are prompted with a list of style definitions for the selected page element. Enter a variable name or select an existing variable name for one or more style definitions. The recorder adds a GetStyle step for each style you identify.

HTMLCell Get Index

Returns the index value for an HTML table cell. Drag and drop over the HTML table cell. You are prompted for the return variable that will contain the HTML index.

See [CellGetIndex](#)

HTMLCell Get Value

Returns the value for an HTML table cell. Drag and drop over the HTML table cell. You are prompted for the return variable that will contain the HTML cell value.

See [CellGetValue](#)

Set Scroll Key

Adds a Scroll.Key_Set step. Drag and drop over a key field in a scroll. Enter a Scroll ID in the Step Modification area and click the Confirm icon.

See [Incorporating Scroll Handling](#).

Show Popup

Opens field pop-ups or organization chart menu items.

See [MouseOver](#).

Verify

Adds a Verify step. Drag and drop over a page element. The step is automatically populated with the object ID and value of the field.

See [Verify](#).

<Select Action> - Step Insert Actions

These recording action tools insert test steps that provide logic constructs, manage scroll actions, and add variables and logs.

This list describes the step insert action options that are available in the Recorder Toolbar <Select Action> list:

Add Log

Adds a Log step.

You are prompted to select a Log action and enter text for the Message and Details.

Add Variable

Adds a Variable.Set_Value step. You are prompted for Variable Name and Value. User-defined variables and PTF test variables appear in the <Select Variable> drop down list.

See [Variable](#), [System Variables](#).

End-If

Adds a Conditional.End_If step.

If-Then

Adds a Conditional If_Then construct. You are prompted for an expression, such as &Exists=True. The Recorder inserts a Conditional.If_Then step. Record additional steps, then select *End-If* to add a Conditional.End_If step.

See [Conditional](#).

Scroll Action

Adds a Scroll.Action step. You are prompted to enter a Scroll ID, return variable, and action. Click the Confirm icon.

<Select Action> - Usage Monitor Actions

These recording action tools insert test steps that control usage monitor actions.

This list describes the usage monitor action options that are available in the Recorder Toolbar <Select Action> list:

Start	Adds a UsageMonitor.Start step.
Stop	Adds a UsageMonitor.Stop step.

Step Modification Tools

As you are recording, you can modify steps recorded during the current session without stopping the recording and exiting the Test Recorder.

This list describes the step modification tools that are available on the Test Recorder toolbar:



Move to the previous step in the recording. When you move to the previous step or the next step, recording is paused. You must click the Start Recording icon to continue recording.



Move to the next step in the recording. When you move to the previous step or the next step, recording is paused. You must click the Start Recording icon to continue recording.



Specify whether the step is active or inactive.



Drag and drop to capture page element information. This icon is used in conjunction with the drag and drop items in the <Select Action> list.

<Select Scroll Variable>

Select a variable to insert.

This list is used in conjunction with items in the <Select Action> list that utilize variables.



Accept step modifications. If you move to another step or continue recording without accepting modifications, the modifications are lost.



Cancel modifications.



Modify the step.



Insert a #PREFIX# keyword at the beginning of the text in the Value field.



Edit test step comments.

Recorder Utility Tools

This list describes the recorder utility tools that are available on the Test Recorder toolbar:



Copy the recording to the clipboard.



Configure recording settings.

- Use Page Prompt

If you select the Use Page Prompt check box, the Test Recorder will replace menu navigation steps with a Page.Prompt step and Page.PromptOK step. The Test Recorder records menu navigation steps, but they are set to inactive.

Note: When recording a search page with facets, PTF does not automatically insert Page.Prompt constructs because all user actions with facets must be recorded.

When using Page Prompt mode, explicitly enter the values for the key fields. Do not perform a partial search and pick from the list. If you must click a dropdown list, search, or take any other action on a search page, do not use Page Prompt mode, use explicit menu navigation.

Note: Page Prompt uses advanced search mode on search pages. In certain cases, objects that are available in basic search mode are not available in advanced search mode. If you record in Page Prompt mode on a search page that uses basic search mode, you may encounter 'Object not found' errors. If that occurs, you can either delete the extra steps or inactivate the Page.Prompt steps and activate the navigation steps.

See [Page](#).

- Message Recognition

Select the Message Recognition check box to automatically configure message recognition for any messages, such as error, warning, or information messages, that the application generates during recording. When the Test Recorder adds a new message it also sets Use Message Recognition to True.

See [Handling Application Messages](#).

- Run Control Recognition

If you select the Run Control Recognition check box, the Test Recorder will replace all recorded actions on a PsRun Control (Process Scheduler page) with the Process.Run Step.

Note: Similar to the Page Prompt with facet, this option is only applicable to a conventional Process Scheduler page.

See [Run](#)

- Ignore Login Steps

If you select the Ignore Login Steps check box, the Test Recorder will ignore all actions on the login page.

- File Download Prompt

Select the File Download Prompt check box to prompt for the file path when recording a step that does a file download.

See [Download](#)

- Use Scroll Variables

If you select the Use Scroll Variables check box, when recording activity in scrolls and grids, the Test Recorder will replace the row number suffix $\$n$ by the selected PTF variable. For example if the field is FIELDNAME \$0, and the scroll variable is &scr1, PTF will record FIELDNAME&scr1.

See [Scroll](#)

- Use Page Expand

If you select the Use Page Expand check box, when you click an image to expand a page section or scroll area, instead of recording the image click, PTF will record the page.expand action.

See [Page](#)

Using the Log Viewer

Whenever you run a test, PTF creates an execution log. The log is located in PTF Explorer under the test name, in the log folder specified in Execution Options.

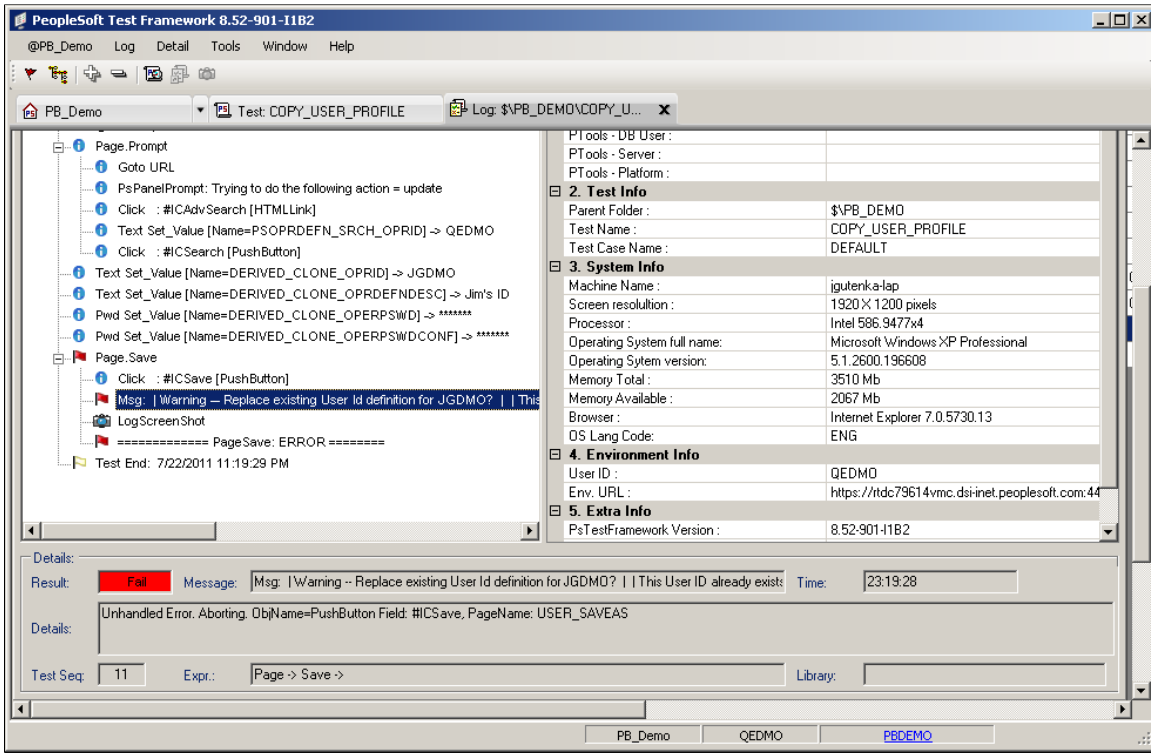
After you run a test, PTF automatically displays the log in the Log Viewer.

You can also view a log by opening it from PTF Explorer.

This example shows the Log Viewer:

Image: Log Viewer

This example illustrates the fields and controls on the Log Viewer. You can find definitions for the fields and controls later on this page.



The Log Viewer has three panes:

- The left pane displays the log details.
Typically, the log will contain one high-level entry for each step in the test.
- The right pane displays the execution options that were used for the test.
- The bottom pane displays details about the selected entry in the log.

Log Viewer Menus

The following menus appear when the Log Viewer has focus. Note that many menu commands are specific to the currently selected item.

This table describes the Log menu commands:

Log Menu Command	Usage
Find	Find text within a log.
Expand All	Expands all the selections in the log.
Collapse All	Collapses all the selections in the log.

Log Menu Command	Usage
Change Log View	Toggles between a view that shows log results as colored flags and a view that shows log results as shaded labels.
Highlight Errors	Highlights log entries with errors in yellow.
Copy Link to Clipboard	Copies a link to the log to the clipboard. You can send this information to another user, who can use the Window, Quick Open feature to open the log without having to navigate to it in PTF Explorer.
Export	Exports a log to an XML file.

This table describes the Detail menu commands:

Detail Menu Command	Usage
Go to Test Step	Accesses the test and selects the step that corresponds to the selected log entry.
Open Link	Opens the application URL in the browser or opens an external file, such as a DataMover log.
Open Screenshot	Opens the screen shot that corresponds to the selected log entry.

This table describes the Window menu commands:

Window Menu Command	Usage
Quick Open	Using information from the Copy Link to Clipboard command, quickly opens a log without having to navigate to the log in PTF Explorer.
Close All	Closes all open windows.

Related Links

[Reviewing Test Logs](#)

[Interpreting Logs](#)

Chapter 4

Creating Tests and Test Cases

Creating Tests

This section discusses how to create and manage tests.

Creating a New Folder

Each test, along with the related test cases and logs, is stored in a folder in the PTF Explorer tree structure.

To create a new folder:

1. In PTF Explorer, highlight the folder in which you want to create the new folder or highlight Home to create the folder at the root level.
2. Select Create, Folder.
3. Enter a new folder name.
4. Click OK.

Note: Folder names must not contain the following characters:
space & ? / \ * < > ' "

Creating a New Test

To create a new test:

1. Highlight a folder in PTF Explorer.
2. Select Create, Test.

The Test Editor launches with a new test.

3. (Optional) In the Test Descr field, enter a description for the test case.
4. (Optional) Click the Test Properties icon to enter a long description of the test case.

It is a good practice to provide a good description for a test, such as a description of the product, feature, or business process being tested. Test names are limited in length and provide little information.

5. (Optional) Click the Test Comment to add comments for the test.
6. Select Test, Save or click the Save button in the toolbar.
7. Enter a name for the new test.

8. Click OK.

Naming Tests

These rules apply to test names:

- Test names and test case names are limited to 30 characters in length.
- Test names and test case names can consist of letters, numbers, and underscores, but they must begin with a letter.
- PeopleSoft Test Framework (PTF) converts test names and test case names to uppercase.
- Two tests in the same database instance must not have the same name, even if they reside in different folders.

Because PTF data resides in the application database, conflicts may occur if PTF users or administrators attempt to copy tests or test cases from one database to another database containing tests or test cases with the same name.

Copying a Test

To copy a test:

1. Highlight a test in the PTF Explorer.
2. Select Edit, Copy or press Ctrl + C to copy the test.
3. Highlight the folder where the copy of the test is to be located.
4. Select Edit, Paste or press Ctrl + V to paste the test.
5. Enter a name for the new test.

When you copy and paste a test the test cases are copied as well, but not the logs. When you cut and paste a test the logs are moved also.

Renaming a Test

A PTF Administrator or Editor can rename a test, however the test name must be unique to the database instance. A PTF User can only rename tests within the myFolder folder. Renaming a test or test case will update the test or test case name in all tests that make a call to the renamed test or test case.

To rename a test:

1. Highlight a test in the PTF Explorer.
2. Right-click and select Rename or select Edit, Rename from the menu.
3. Enter the new name and click OK.

The test is renamed.

Renaming a Folder

A PTF Administrator or Editor can rename a folder, however the folder name must be unique within that level of the PTF Explorer tree. Before you can rename a folder, you must close all of the tests that are associated with that folder.

To rename a folder:

1. In PTF Explorer, highlight the folder you want to rename.
2. Right-click and select Rename or select Test, Rename from the menu.
3. Enter the new folder name and click OK.

Note: Folder names must not contain the following characters:
space & ? / \ * < > ' "

Recording Tests

When you record a test, PTF monitors each action you perform in the target application and creates a corresponding step in the test.

PTF Recorder populates these fields for each step:

- Seq
- ID
- Active
- Type
- Action
- Recognition
- Parameters
- Value

This is an example of test steps:

Image: Example of Test Steps

This example illustrates the fields and controls associated with the test steps. You can find definitions for the fields and controls later on this page.

Seq	ID	Comment	Active	Scroll ID	Type	Action	Recognition	Parameters	Value
1	1		<input checked="" type="checkbox"/>		Browser	Start_Login			
2	28		<input checked="" type="checkbox"/>		Browser	FrameSet			
3	29		<input type="checkbox"/>		Text	Set_Value	Name=userid		PTDCCBL
4	30		<input type="checkbox"/>		Pwd	Set_Value	Name=pwd		1ENC29CDD008E830562C4010DE2C487016EC80509EEC
5	31		<input type="checkbox"/>		Button	Click	Name=Submit		
6	32		<input type="checkbox"/>		Link	Click	id=phnavbca_PORTAL_ROOT_OBJECT		
7	33		<input type="checkbox"/>		Link	Click	Name=hdra_PT_PEOPLETOOLSIndex=0		
8	34		<input type="checkbox"/>		Link	Click	id=hdra_PT_SECURITY		
9	35		<input type="checkbox"/>		Link	Click	id=hdra_PT_PERMISSIONS_POLES		
10	36		<input type="checkbox"/>		Link	Click	id=hdra_PT_USER_PROFILES		
11	37		<input type="checkbox"/>		Link	Click	innerText=User ProfilesIndex=1		
12	38		<input type="checkbox"/>		Browser	FrameSet	TargetContent		
13	39		<input checked="" type="checkbox"/>		Page	Prompt	MAINTAIN_SECURITY.USERMAINT.GBL		update
14	40		<input checked="" type="checkbox"/>		Page	PromptOk			
15	41		<input checked="" type="checkbox"/>		Page	Prompt	MAINTAIN_SECURITY.USERMAINT.GBL		update
16	42		<input checked="" type="checkbox"/>	2	Scroll	Action		ret=Ivariable1	ina
17	43		<input checked="" type="checkbox"/>		Variable	Set_Value	A Test		Test 1
18	2		<input checked="" type="checkbox"/>		Browser	FrameSet			
19	3		<input type="checkbox"/>		Pwd	Set_Value	Name=pwd		1ENC1EED306FAFC3A3E9F980D862888A9ACECC0EF
20	4		<input type="checkbox"/>		Button	Click	Name=Submit		
21	5		<input checked="" type="checkbox"/>		Text	Set_Value	Name=SEARCH_TEXTIndex=1		
22	6		<input type="checkbox"/>		Div	Click	Classname=ptnav_ZoggleIndex=72		
23	7		<input type="checkbox"/>		Div	Click	Classname=ptnav_ZoggleIndex=78		
24	8		<input type="checkbox"/>		Link	Click	innerText=Pivot Grid Wizard		
25	9		<input type="checkbox"/>		Browser	FrameSet	TargetContent		

Seq and ID are system-generated fields.

Recognition, Parameters, and Value are not used with some actions, such as Browser.Start or Page.Save.

The Comment field is populated by the test developer to document the test.

See [Using the Test Editor](#).

Test Action Tools

In addition to simply recording your interaction with the target application, the test recorder enables you to add steps to perform the following functions:

- Verify the value in a page control.
- Check for the existence of a page element.
- Get a property of a page element.
- Populate a variable.
- Insert an entry in the execution log.
- Insert a conditional construct.
- Reference scrolls.
- Reference HTML table cell contents.
- Add or modify step comments.

Related Links

[Using the PTF Test Recorder](#)

[PTF Test Language](#)

Recording a Test

To record a test:

1. Create a new test or open an existing test.
2. If you are recording within an existing test, highlight a test step. The system inserts the steps in the new recording following the highlighted step. If you are recording a new test, recording typically begins at Step 2. By default, Step 1 is `Browser.Start`, or `Browser.Start_Login`, depending on your configuration options.
3. With a test open, select **Test, Open Test Recorder** or click the **Show Test Recorder** button in the toolbar.
4. The PTF Test Recorder toolbar appears.

See [Using the PTF Test Recorder](#).

5. *Hook* a browser, that is, associate a PeopleSoft application browser with the test.

To hook a browser, you can either:

- Click the **Start Web Client** icon on the recorder toolbar. This starts the web browser with the default test application URL from the current execution option..
- Select a browser window with the test application open and hook the application by dragging the **Select Browser** icon to the browser window.

Note: PTF will only hook a browser that was initiated by PTF, either by using the **Start Web Client** icon or by running a test.

6. Click the **Start Recording** icon in the Test Recorder toolbar to begin recording.
7. Perform the test steps in the PeopleSoft application.
8. As needed, insert additional steps using the Test Recorder test action tools.
9. Modify steps or add step comments using the Test Recorder step modification tools.
10. Click the **Stop Recording** icon in the Test Recorder toolbar to end the recording and write the steps to the test.
11. Save the test.

Related Links

[Using the PTF Test Recorder](#)

[Defining PTF Configuration Options](#)

Opening Tests

This section discusses how to open tests.

Opening Tests with PTF Explorer

You can open a single test or multiple tests with PTF Explorer.

Opening a Single Test

To open a single test with PTF Explorer:

1. Expand folders to navigate to the desired test.
2. Highlight the test.
3. Use one of the following methods to open the test:
 - Right-click and select Open.
 - Select Test, Open from the PTF Explorer menu.
4. The test opens, and the test editor is the active window.

Opening Multiple Tests

To open multiple tests with PTF Explorer:

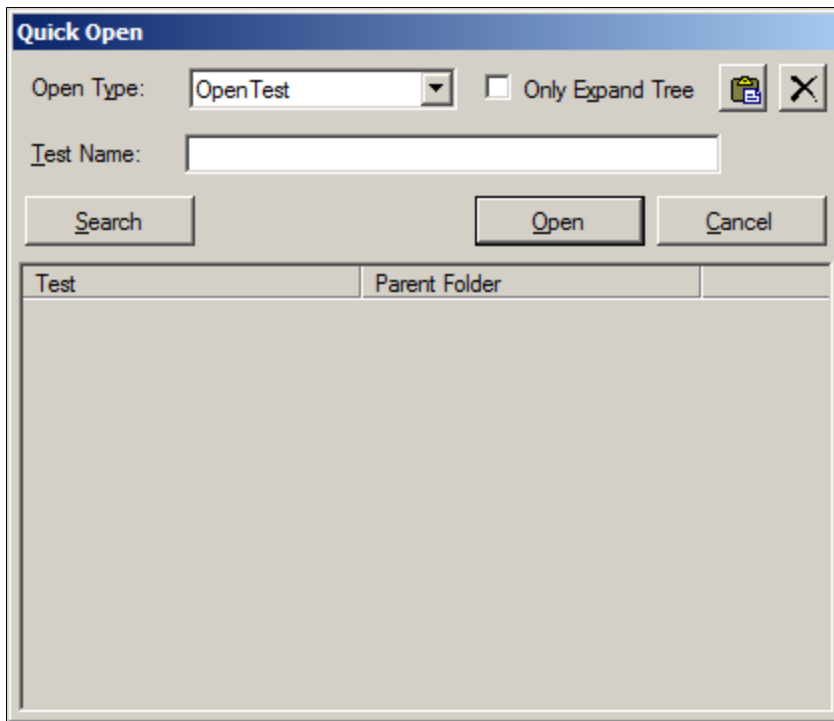
1. Expand the tree folders to view the desired tests.
2. Select each test that you want to open (use Ctrl+Click to multi-select).
3. Use one of the following methods to open all of the selected tests:
 - Right-click and select Open.
 - Select Test, Open from the PTF Explorer menu.
4. All of the tests open.

Opening Tests Assets with the Quick Open Dialog Box

Use the Quick Open dialog box to search for and open one or more test assets. To access the Quick Open dialog box, select Window, Quick Open.

Image: Quick Open Dialog Box

This image illustrates the fields and controls on the Quick Open dialog box.



The following fields and controls are available on the Quick Open dialog box.

Open Type	Select the type of test asset to open. Options are: <ul style="list-style-type: none"> • <i>Open Test</i> • <i>Open Case</i> • <i>Open Log</i>
Only Expand Tree	Select this check box to only expand the tree folder the asset is associated with, not load the test asset.
Test Name	Enter the test name.
Case Name	This field is available only when Open Type is set to <i>Open Case</i> . Enter the test case name.
Log Folder	This field is available only when Open Type is set to <i>Open Log</i> . Enter the name of the log folder.
Log Name	This field is available only when Open Type is set to <i>Open Log</i> . Enter the name of the log.

Search

Click to search for tests that *contain* the value entered in Test Name. Partial matches are supported for Test Name when using the Search button.

Matching tests appear in the grid. To open one or more tests in the grid, select the check box adjacent to the test name, then click Open.

Open

Click to open the test asset.

An exact match is required if you do not use Search before you click Open.

Working with Test Cases

Often, you will want to run the same test multiple times using different values in the Value column. Test cases enable you to associate different sets of data with a test. You can view and edit both the test and test case in a single unified window.

Creating a New Test Case

To create a new test case:

1. With a test open, click the New button to the right of the Test Case field.
2. Enter a name for the new test case.
3. (Optional) In the Test Case Descr field, enter a description for the test case.
4. (Optional) Click the Test Case Properties icon to enter a long description of the test case.

It is a good practice to provide a good description for a test case. Test case names are limited in length and provide the user with little information.

5. (Optional) Click the Test Case Comment icon to add a comment for the test case.
6. The new test case provides a set of new, empty Value fields for all the steps of the test.

Enter a new value for each step that requires a value.

7. Save the test case.

Image: New test case with blank values

This example illustrates a new test case with blank values.

Seq	ID	Comment	Active	Scroll ID	Type	Action	Recognition	Value
1	1		<input checked="" type="checkbox"/>		Browser	Start_Login		
2	2		<input type="checkbox"/>		Link	Click	innerText=Sign in to PeopleSoft	
3	3		<input type="checkbox"/>		Text	Set_Value	Name=userid	
4	4		<input type="checkbox"/>		Pwd	Set_Value	Name=pwd	
5	5		<input type="checkbox"/>		Button	Click	Name=Submit	
6	6		<input type="checkbox"/>		Link	Click	id=fldra_PT_PEOPLETOOLS	
7	7		<input type="checkbox"/>		Link	Click	id=fldra_PT_SECURITY	
8	8		<input checked="" type="checkbox"/>		Browser	FrameSet	TargetContent	
9	9		<input type="checkbox"/>		Link	Click	Name=defaultinnerText=Copy User Profiles	
10	10		<input checked="" type="checkbox"/>		Page	Prompt	MAINTAIN_SECURITY.USER_SAVEAS.GBL	
11	11		<input checked="" type="checkbox"/>		Text	Set_Value	Name=PSOPRDEFN_SRCH_OPRID	
12	12		<input checked="" type="checkbox"/>		Page	PromptOk		
13	13		<input checked="" type="checkbox"/>		Text	Set_Value	Name=DERIVED_CLONE_OPRID	
14	14		<input checked="" type="checkbox"/>		Text	Set_Value	Name=DERIVED_CLONE_OPRDEFNDESC	
15	15		<input checked="" type="checkbox"/>		Pwd	Set_Value	Name=DERIVED_CLONE_OPERPSWD	
16	16		<input checked="" type="checkbox"/>		Pwd	Set_Value	Name=DERIVED_CLONE_OPERPSWDCONF	
17	17		<input checked="" type="checkbox"/>		Page	Save		
*			<input checked="" type="checkbox"/>					

Creating a Test Case With Values

For convenience, you may want to create a test case with the Value column populated with the values from the original test or from another test case. Then, you can edit the values in the new test case.

Note: The test case associated with the original test is named DEFAULT. Every test has one test case named DEFAULT.

You can select an existing test case from the Test Case drop-down list box.

PTF displays the number of test cases associated with the test, including the DEFAULT test case, next to the Test Case field label.

To create a test case with values:

1. With a test case open, select Test, Test Case Save As.
2. Enter a name for the new test case.
3. (Optional) Enter a short description and a long description for the test case.
4. Highlight each value you want to change and enter a new value.
5. Save.

Managing Values when Pasting Test Steps

When you cut/copy and paste test steps, a dialog box appears providing options for how to manage test case values. You can select one of the following options.

- Paste only the steps, leaving the test case values blank.
- Paste the current test case values from the source to the target.
- Paste the test case values only for matching test case names.

Exporting and Importing Test Cases

You can create or modify a large number of test cases by exporting test cases to a file, editing the file, and then importing the file back into the test.

You can export a text file using the following delimiter characters:

- Comma
- Semicolon
- Vertical pipe
- Tab

Exporting Test Cases

To export test cases:

1. With a test open, select Test, Export.
2. Specify a location for the file.
3. Select a separator character.

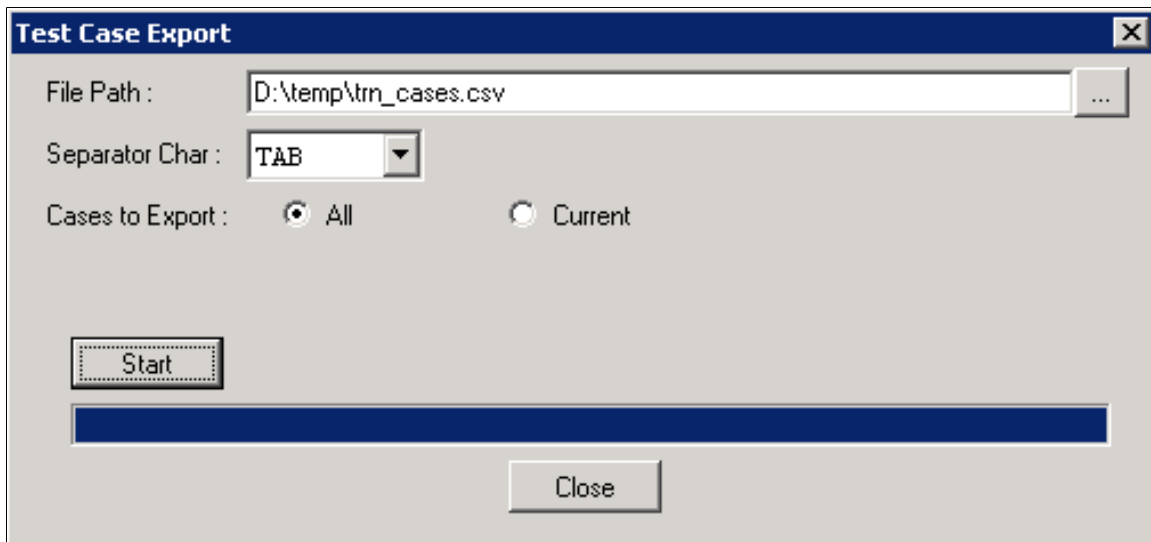
Note: The default separator character is a comma, which is also the default separator used by spreadsheet programs. If any of the values in your test case contain commas, the value will be split into separate fields. If you have commas in your test case values, choose a different separator.

4. Specify whether to export one or all test cases.
5. Click Start.

The bar along the bottom of the dialog box denotes the progress of the export.

Image: Example of export using a tab delimiter for all test cases

This example illustrates export using a tab delimiter for all test cases.



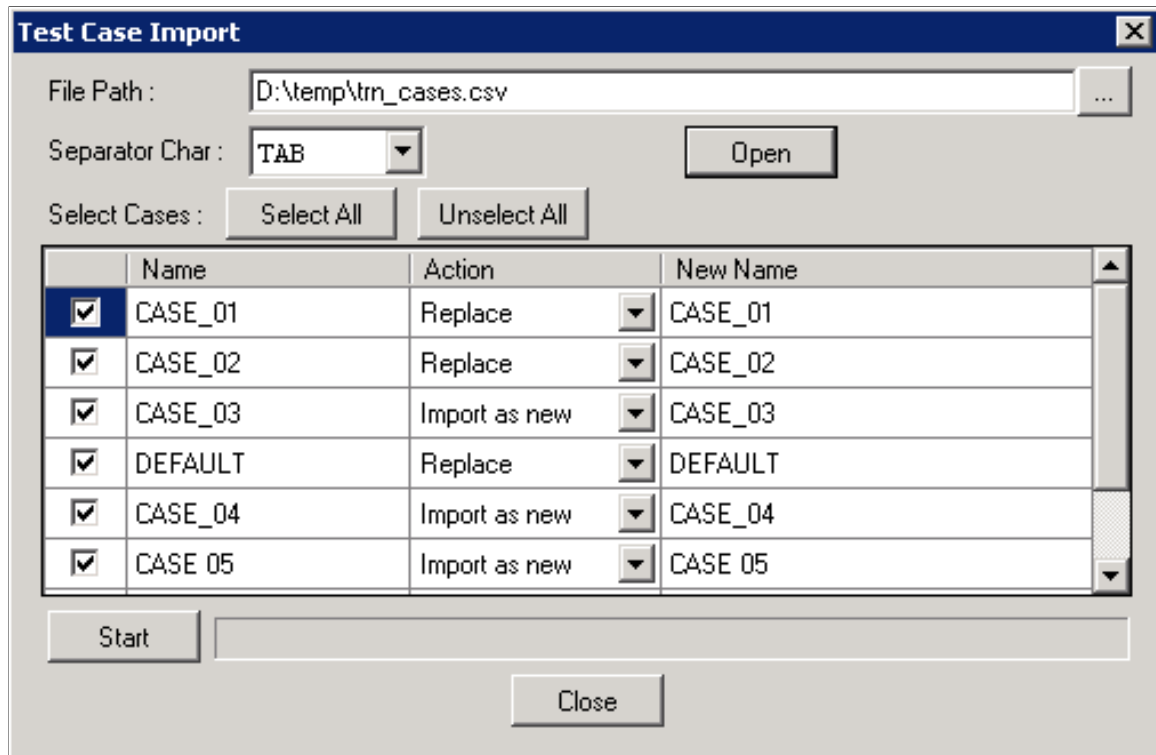
Importing Test Cases

To import test cases.

1. In PTF, with a test open, select Test, Import.
2. Enter the file path or browse to the file.
3. Specify the separator character.
4. Click Open to display a list of test cases in the file.

Image: Test Case Import dialog box

This example illustrates the fields and controls on the Test Case Import dialog box. You can find definitions for the fields and controls later on this page.



If the test case already exists in the test, the action is set to Replace. If the test case does not exist in the test, the action is set to Import as new. If you change the action to Import as new for an existing case, you must also enter a new test case name in New Name field.

5. Select the test cases to import.
6. Click Start.

Executing Tests

This section describes how to execute tests and test cases.

Executing a Test

To execute a test:

1. With a test open in PTF, select Test, Run.

Alternatively, you can press F5 or click the Run button.

2. If Yes is specified in the Prompt for Options field for the default execution option, then the Execution Options dialog box appears.

Select an execution option from the list and click Accept.

PTF opens the PeopleSoft application specified in Execution Options and executes the steps in the test.

3. After the test executes, PTF opens the test log in the Log Viewer.

Note: When PTF executes a test it disables Num Lock and Caps Lock on your keyboard and restores them when the test completes. If the execution terminates abnormally, the test does not complete, or hangs up, and PTF does not restore Num Lock and Caps Lock. Select Test, End or click the End icon in the toolbar to end the test and restore Num Lock and Caps Lock. Also, you should not press the Caps Lock or Num Lock keys during test execution; if you do, subsequent test steps may fail. Also, you should not press the Caps Lock or Num Lock keys, during test execution; if you do, subsequent test steps may fail.

Related Links

[Reviewing Test Logs](#)

Executing a Test Case

To execute a test case:

1. With a test open in PTF, select a test case from the Test Case drop-down list.

If you do not select a test case, the system uses the DEFAULT test case.

Alternatively, you can open a test case from PTF Explorer.

2. Select Test, Run.
3. Review the log in the Log Viewer.

Related Links

[Reviewing Test Logs](#)

Executing a Test from a Specific Step

You can start execution from a specific step using the Run from Step option in the Test menu.

Important! You are responsible for completing any required preceding steps when executing a test using the Run from Step command, such as Browser.Frameset, assigning variables, and hooking the browser, for example.

To execute a test from a specific step:

1. With a test open in PTF, select Test, Run from Step.

Alternatively, you can press F6.

2. In the Run from Step dialog box, enter or select the step sequence number from which to begin test execution, and click OK.

If the specified sequence number does not exist, test execution begins at either the next highest sequence number, or the last test step sequence, whichever is the lower value.

If the specified sequence number is an inactive step, then test execution begins with the next highest sequence number.

3. After the test executes, PTF opens the test log in the Log Viewer.

The log file will include information about which step the test was started from.

Executing a Test from the Command Line

You can execute a test from the command line using the **PsTestFW** command.

Enter a *text file* argument which contains all the parameters in the command line. The text file is deleted immediately after PTF client reads it. You can write the text file manually or it is generated from the Change Assistant during runtime.

The parameters in the file must follow the below guidelines:

- Split the parameter and its value with =.
- Enter each parameter and its value in a new line.
- Enter each parameter and its value in a single line.

For example, your text file will include the following statements.

```
-CS=<server>:<port>
-CUA=TRUE
-CO=<userID>
-CP=<password>
-TST=<TestNameFile>
-TC=DEFAULT
-PFX=
-EXO=<ExecutionOptions>
-LOG=<logfile name>.log
```

Syntax

Use the following syntax to login using a *text file* argument.

```
'<Path>\PsTestFw.exe' 'C:\temp\ptf_cmd_param.txt'
```

Use this syntax to execute a test using the existing environment connection:

```
PsTestFw -CD=ConnectionName -CP=ConnectionPassword -TST=TestName
-TC=TestCaseName [-TL=Line X(integer)] [-PFX=Prefix] -EXO=ExecutionOption [-LOG=Log=>
FileName]
```

Use this syntax to execute a test using specified environment connection parameters:

```
PsTestFw -CS=Server -CNO=NodeName -PS=ProxyServer -PU=ProxyUser -PP=ProxyPassword
-CO=UserName -CP=ConnectionPassword -TST=TestName -TC=TestCaseName [-TL=Line X(int=>
eger)]
[-PFX=Prefix] -EXO=ExecutionOption [-LOG=LogFileName]
```

Parameters

-CD= Specify the name of the environment signon to use for connection. This is the Database Name you would select in the

PeopleSoft Test Framework Signon dialog box when signing on to PTF.

The environment signon settings are stored in the environments.xml file in the PTF data directory (C:\Documents and Settings\\Application Data\PeopleSoft\PeopleSoft Test Framework by default). If the environment connection data is not set in the environments.xml file, then you can explicitly specify the connection parameters. See the following table for a description of connection parameters.

See [Creating a Connection to a PTF Environment](#).

-CP=	Specify the user password.
-TST=	Specify the test name.
-TC=	Specify the test case name.
-TL=	(Optional) Specify the line number from which to start the test.
-PFX=	(Optional) Specify the prefix. See #PREFIX# .
-EXO=	Specify the execution option to be used in the execution.
<hr/>	
Note: The name of the execution option must not contain the following characters: space & ? / \ * < > ' "	
<hr/>	
-LOG=	(Optional) Specify the name for the log. The default is unattended.log.

If you do not use the -CD= parameter to specify the connection data, use the parameters in the following table:

-CS=	Specify the server:port to connect to. This is the Server:Port value you would enter in the PeopleSoft Test Framework Signon dialog box when signing on to PTF.
-CNO=	Specify the node name.
-CO=	Specify the user name.
-PS=	(Optional) Specify the ProxyServer:Port.
-PU=	(Optional) Specify the proxy user. If you use network authentication, use the DOMAIN\USER format.
-PP=	(Optional) Specify the proxy password.

Example

The following example uses the `-CD=` parameter to set connection parameters:

```
PsTestFw -CD=QE851 -CP=VP1 -TST=TEST_CMD_LINE -TC=DEFAULT -PFX=Prefix
-EXO=QE851_No_Folder -LOG=my_run_log
```

The following example explicitly sets connection parameters:

```
PsTestFw -CS=rtdc79637vmc:8643 -CNO=PT_LOCAL -PS=ProxyServer:2345
-PU=mydomain\username -PP=pwd123 -CO=VP1 -CP=VP1 -TST=TEST_CMD_LINE
-TC=DEFAULT -PFX=Prefix -EXO=QE851_No_Folder -LOG=my_run_log
```

Log File

The execution will generate an output log file in the PTF data directory (C:\Documents and Settings\\Application Data\PeopleSoft\PeopleSoft Test Framework by default).

If the log file exists it will be overwritten.

This is an example of a log file:

```
<execution>
  <Started>2014-06-18 04:41:46</Started>
  <Param>
    <Database>L921PDVL</Database>
    <TestName>DR_DUMMY_SHELL</TestName>
    <TestCase>DEFAULT</TestCase>
    <ExecOpt>L921PDVL</ExecOpt>
  </Param>
  <Status>Failed</Status>
  <Test>
    <Exec1>
      <Name>DR_DUMMY_LOOP</Name>
      <Case>DEFAULT</Case>
      <LogLine15>case is not DEFAULT2</LogLine15>
      <LogLine16>Execution stopped by Execution.Stop_On_Error: 6/18/2014 4:41:54 PM=>
    </LogLine16>
    <Status>Failed</Status>
  </Exec1>
  <Exec2>
    <Name>DR_DUMMY_LOOP</Name>
    <Case>DEFAULT2</Case>
    <Status>Passed</Status>
  </Exec2>
  <Exec3>
    <Name>DR_DUMMY_LOOP</Name>
    <Case>DEFAULT3</Case>
    <LogLine44>case is not DEFAULT2</LogLine44>
    <LogLine45>Execution stopped by Execution.Stop_On_Error: 6/18/2014 4:41:55 PM=>
  </LogLine45>
  <Status>Failed</Status>
</Exec3>
<LogFolder>TEST</LogFolder>
<LogName>4. PS 2014-06-18 16:41</LogName>
<Message>Execution stopped by Execution.Stop_On_Error: 6/18/2014 4:41:55 PM</Message>
</Test>
</execution>
```

Return Code Option For PTF Command Line Execution

The `iRetCode` command returns the status of test execution from the command line.

The syntax for this command is:

```
iRetCode= PsTestFw -CS=<Environment_server_port> -CNO=<optional_Environment_Node_ID>  
>  
-CO=<Environment_User_ID> -CP=<Environment_User_Password> -TST=<ShellTest_Name>  
-TC=<Test Case Name> -EXO=<Execution_options> -LOG=C:\Temp\LOG1.xml
```

Return values are:

-1	Failed
-2	Not Completed-FatalError
0	Passed
-3	Starting
-10	Else

Reviewing Test Logs

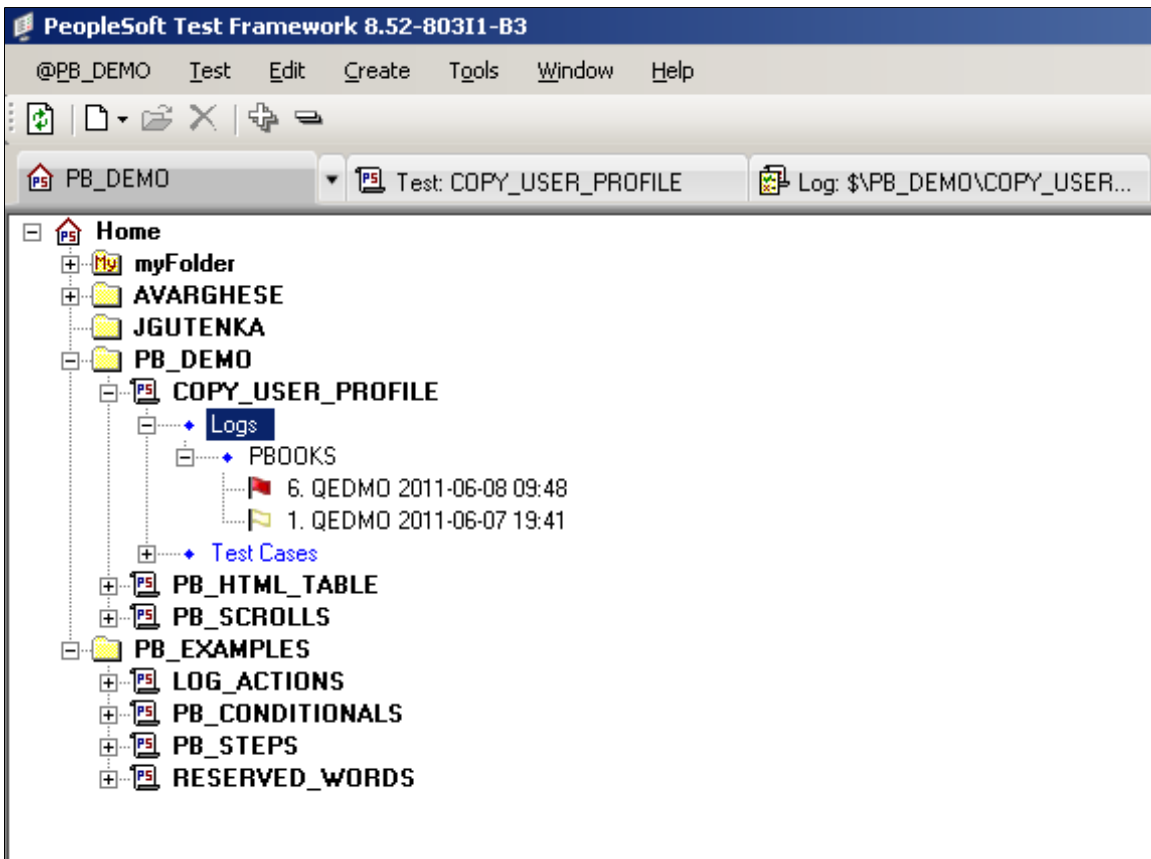
Whenever you run a test, PTF creates an execution log entry. The log is located in PTF Explorer under the test name, in the log folder specified in Execution Options.

After you run a test, PTF automatically displays the log in the Log Viewer.

You can also view a log by opening it from PTF Explorer.

Image: Example of PTF Explorer Showing Log Entries in the Logs Folder

This example illustrates log entries in the Logs folder of PTF Explorer.

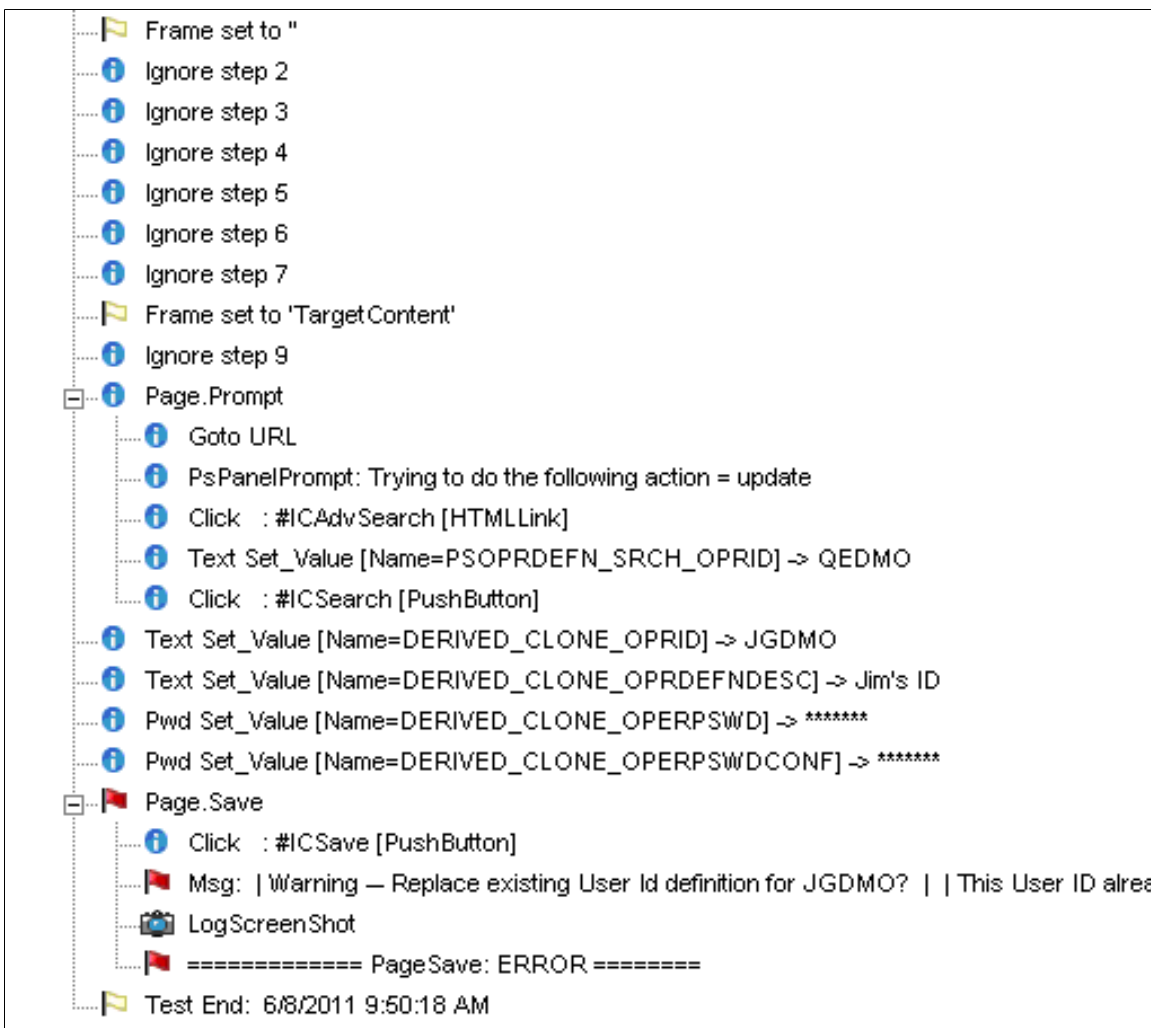


To open a log from PTF Explorer, do one of the following:

- Highlight a log entry and select Test, Open.
- Double-click a log entry.
- Right-click a log entry and select Open.

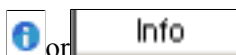
Image: Example of Log Entries in a Test Log

This example illustrates log entries in a test log. You can find icon definitions later on this page.



Typically, the Log Viewer includes one high-level entry for each step in the test. Entries for complex steps appear in collapsible sections that can be expanded to view the additional details.

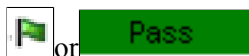
An icon or shaded label appears next to each log entry, indicating the success state of the associated step:



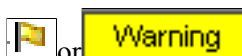
Information message only.



Information message only.



Step was successful.



Step was successful, but with a warning.



Step failed.



The test encountered a condition that it was not configured to handle, or the test encountered an error while PTF was configured to stop on all errors.

Note: Highlight a log entry and select Detail, Go to Test Step or right-click and select Go to Test Step to open the test with the corresponding step selected.

Related Links

[Interpreting Logs](#)

Exporting Test Logs to XML

PTF provides the ability to export test logs to XML. Once the logs have been exported to XML, you can compare the results in a text editor, such as NotePad or EditPlus, or view them in most web browsers. You can export all fields, or limit the export to specific fields.

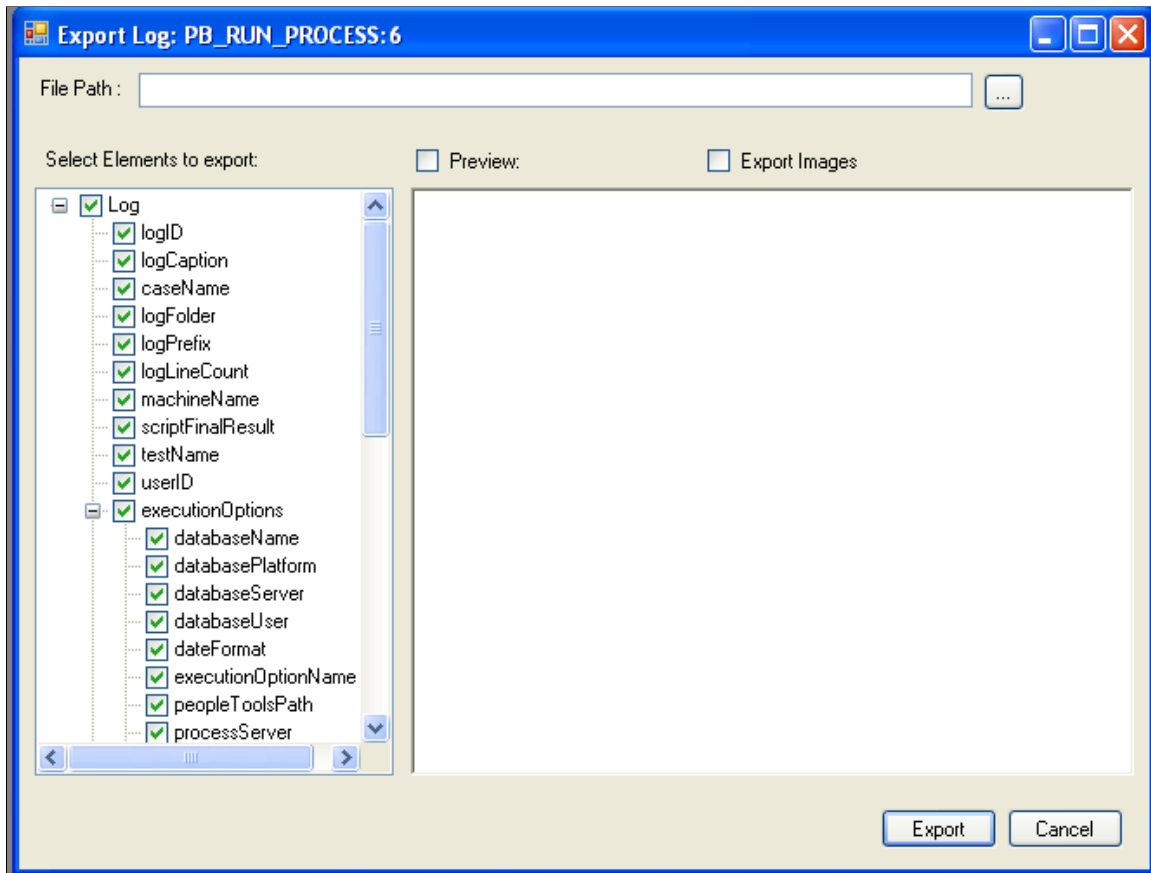
To export a test log to XML:

1. In PTF Explorer, open the logs folder.
2. Right-click on a specific log.
3. Select Export.

The Export Log dialog box is displayed:

Image: Export Log Dialog Box

This example illustrates the fields and controls on the Export Log Dialog Box. You can find definitions for the fields and controls later on this page.



On the Export Log dialog, the user can see the structure(schema) of the Log XML, include/exclude elements to export and set the order of the elements, also the user is able to preview the exported result.

By default, all the timestamp (lastUpdate element in the schema tree) fields are excluded, as well as extra information such as ExecutionOptions, LogParentLineID, and so forth.

File Path	Enter the file path. The file name is automatically generated based on the test name and log number. For example PB_RUN_PROCESS-LOG-6.xml.
Select Elements to Export Column	By default all items are selected, click the check box to deselect specific items.
Preview	Select this check box to preview the xml.
Export Images	Select this check box to include images in the exported file.

You can also set an option to automatically generate XML log files every time that tests are executed. For more information, see [Configuring Execution Options in PTF Client](#).

Organizing Tests In PTF Explorer

This section discusses how to use PTF Explorer to organize tests.

Cutting and Pasting Multiple Tests or Folders

PTF Administrators and PTF Editors can cut and paste tests and folders. A user with only the PTF User role is not able to cut folders that reside outside of the myFolder folder.

To cut and paste (move) multiple tests or folders:

1. In the PTF Explorer, select multiple tests and folders to move to a new location on PTF explorer tree.
2. Right-click and select Cut, or select Edit, Cut from the menu.
3. Select the desired target folder location on PTF explorer tree.
4. Right-click and select Paste, or select Edit, Paste from the menu.

All contents are moved to the target folder location.

Deleting Multiple Tests or Folders

PTF Administrators and PTF Editors can delete tests and folders. A user with only the PTF User role is not able to delete folders that reside outside of the myFolder folder.

To delete multiple tests or folders:

1. In the PTF Explorer, select multiple tests and folders to delete.
2. Right-click and select Delete, or select Edit, Delete from the menu.
3. Click OK to confirm the delete.

The folders or tests are deleted.

Expanding or Collapsing Tests and Folders

To expand or collapse multiple tests or folders:

1. In the PTF Explorer, select multiple tests and folders to expand or collapse.
2. Right-click and select Expand Selection or Collapse Selection,

PTF expands or collapses all selected tests and folders at the level of selection in the tree.

Chapter 5

Developing and Debugging Tests

Using the Message Tool

As you modify a test, you will often need to use the Message tool to capture details about a browser object. You can then use these details to modify a test step.

You can use the Message tool to display and collect information about fields in the application. The Message tool also monitors test execution. The Message tool displays types, actions, IDs, and values for each step of a PTF test as the test executes.

Note: The Message tool works only with Internet Explorer.

Select Tools, Message to open the Message tool.

Click and drag the Object Properties icon and hover over a browser object to view details about that object in the Message window.

Image: Message Tool ID Values

This example illustrates Message tool ID values.



To copy and paste recognition information from the application browser to the PTF Test Editor, drag and drop the Object Properties icon onto a browser object. Object recognition details appear in the Message tool. Double-click the name in the Message tool to copy it to the clipboard. You can then paste the information into the Recognition field of a test step. To automatically copy each selection to the clipboard, select **MENU**, Auto Copy to Clipboard.

You can also use the Message tool to monitor test execution. The Message tool displays types, actions, IDs, and values for each step of a PeopleSoft Test Framework (PTF) test as the test executes.

The Message tool includes the following items:

MENU

Click to access the Message tool menu. Menu items include:

- Clear: Select to clear the message display content area.
- Auto Copy to Clipboard: Select to automatically copy object recognition details to the clipboard. This is a toggle that you can turn on or off. When this option is turned on, a check appears next to the menu item to indicate that it is enabled.
- Auto Collapse: Select to have the message window automatically collapse when not active. This is a toggle that you can turn on or off. When this option is turned on,

a check appears next to the menu item to indicate that it is enabled.

- **Show HTML Browser:** Select to open the Object Properties dialog, where you can view the properties of browser objects as you hover over them with the Object Properties icon.
- **Save Windows Settings:** Saves message tool window settings, including size and position, to the C:\Documents and Settings\\AppData\Roaming\PeopleSoft\PeopleSoft Test Framework\PstTstMsgConfig.xml file.
- **Close:** Closes the Message tool menu.
- **Exit:** Closes the Message tool.



Object Properties Icon

Click then drag and hover over a browser object to view details about that object in the message area.

Message Area

The message area is the yellow rectangular section of the Message tool, which displays the following object recognition information:

- When you use the Object Properties icon to hover over a browser object, it displays details for that browser object.
- During test execution, it displays information about each step of a PTF test as the test executes.

Moving and Resizing the Message Tool Window

Use the following options to move and adjust the width of the Message tool:

- To move the Message tool, select **MENU**, then drag and drop the window to a new location.
- To change the width of the Message tool, position the pointer over the left border of the Message tool window until the cursor changes to a double arrow, then click and drag to adjust the width.

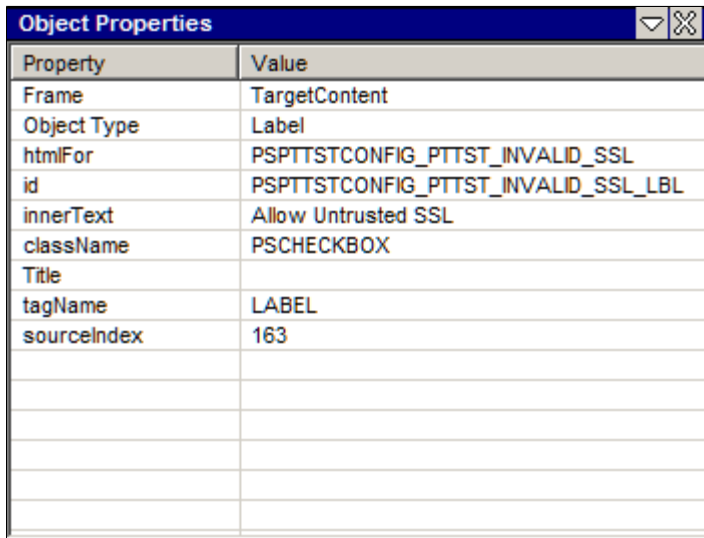
To save the Message tool window's location and width, select **MENU**, Save Windows Settings.

Viewing Browser Object Properties

To view additional properties about a browser object, access the Message tool and select **MENU**, Show HTML Browser.

Image: Example of the HTML Browser

This example illustrates the HTML Browser Object Properties window.



Property	Value
Frame	TargetContent
Object Type	Label
htmlFor	PSPTTSTCONFIG_PTTST_INVALID_SSL
id	PSPTTSTCONFIG_PTTST_INVALID_SSL_LBL
innerText	Allow Untrusted SSL
className	PSCHECKBOX
Title	
tagName	LABEL
sourceIndex	163

The Object Properties window displays properties and values of HTML objects as you hover over them using the Object Properties icon. Double-click a row in the Object Properties window to copy the text to the clipboard.

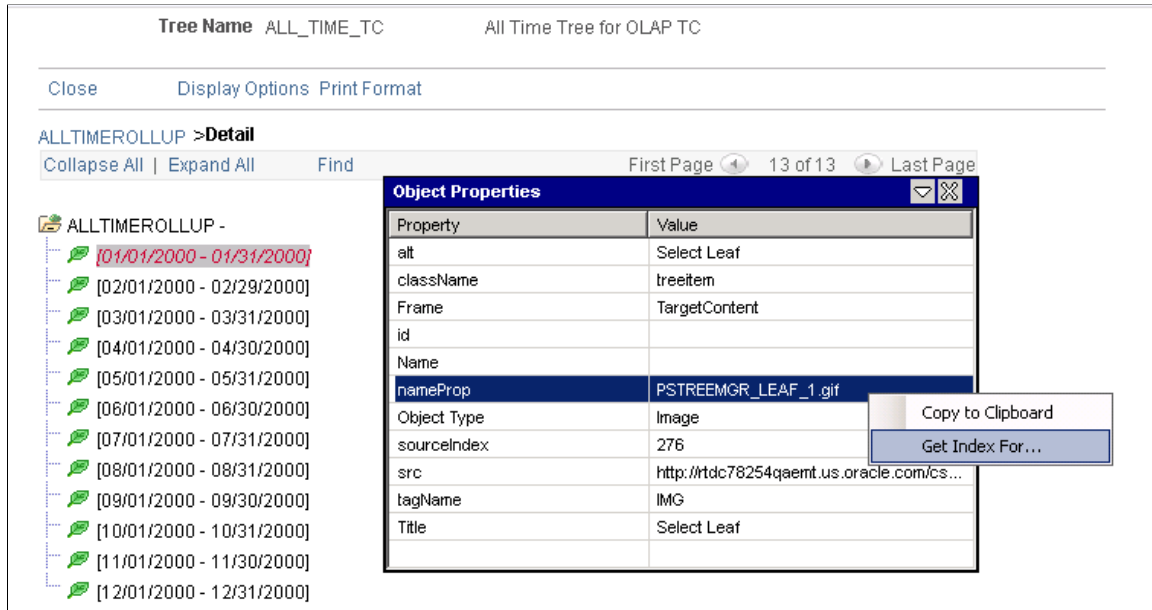
Capturing Index for Non-unique Objects using the HTML Browser

For certain objects that do not have enough of a unique identifier property, you can use the HTML Browser to capture the index during recording. An example of this would be leaves on a tree structure. To capture the index:

1. In PTF Client, select Tools, Message to open the Message tool.
2. Start your recording.
3. Open the tree to the desired location.
4. Click the Object Properties icon in the Message Toolbar and drag it to the leaf image on the tree.
5. Access the Message tool and select Menu > HTML Browser > Show.
6. Right-click on the nameProp in the Object Properties window and select Get Index for.
7. The index will be added to the object Properties.

Image: Object properties Dialog Box

This example illustrates the Object properties Dialog Box displaying the properties for a leaf on a tree. To get the index, you will select Get Index For...



Using Reserved Words

This section discusses how to use reserved words.

Reserved words enable you to access data available from the PTF program when a test is executed.

Reserved words are useful when data is not known before the test is executed. For example, suppose you have the following manual test instruction:

12. Enter the current date into the Voucher Creation Date field.

If, when you record the step, you enter the current date, then you will put specific, or static, data into the test, similar to the following example, which shows the data created by a test recorded on June 30, 2010:

Image: Example of a Test Step with Static Data

This example illustrates a test step with static data for the date.



However, the test instruction calls for the *current date*, which may be different each time you run the test. Data that can change is called *dynamic* data. To make the test data in PTF dynamic, replace the recorded data with the reserved word, **#TODAY**, which represents the date at the moment of test execution, as shown in this example:

Image: Example of Test Step Using the #TODAY Reserved Word

This example illustrates a test step using the **#TODAY** reserved word.



Other reserved words enable you to define specific actions in the Value field of a step.

For example, suppose you are required to test two very similar test scenarios:

1. Create a new pension calculation using the following parameters.
2. Open the pension calculation created earlier and verify that the parameters entered into the application are the same as those specified in the previous scenario.

You can meet this requirement by using a single test step with two test cases. The first test case might be named CREATE and the second named VERIFY.

In the CREATE test case, a step that sets the Calculation Description field might look like this:

Image: Example of a Step that Sets a Value

This example illustrates a step that sets a value.

Text	Set_Value	PA_CALCULATION_DESCR	Sample pension calc
------	-----------	----------------------	---------------------

The VERIFY test case uses the reserved word `#CHECK#` in the Value field. Using the same step, the `#CHECK#` reserved word causes the Set_Value action to behave like a Verify action, which satisfies the second test scenario. Rather than setting the value of the object, the same step now verifies it, as shown in this example:

Image: Example of a Step that Verifies a Value

This example illustrates a step that verifies a value.

<input checked="" type="checkbox"/>	Text	Set_Value	PA_CALCULATION_DESCR	#CHECK#Sample pension calc
-------------------------------------	------	-----------	----------------------	----------------------------

Related Links

[Reserved Words](#)

Using Variables

This section discusses how to use variables.

Variables enable you to store a value in one step and access that data in a subsequent step. Variables are useful when your test requires a value that will not be known until the test is executed.

Variables are always prefixed by an ampersand (&) – both when you set their value and when they represent a value.

Store a value for a variable either by placing the `ret=&varname` parameter in the Parameters field in a step that supports return values, or by using a Variable.Set_Value step.

You can refer to the values stored in a variable in two ways:

- Use the variable in the Recognition field of a Conditional.If_Then step or in the Parameter field for any step that takes a parameter.
- Use the variable in the Value field of any step that sets or verifies the value of an object on the page.

Important! Variables are not automatically initialized. If the variable is not assigned a value in either the recognition or value column, the value will be the variable name. For example, if the variable `&var1` was never initialized, it will return `&var1`.

For example, suppose you have the following test instructions for a test of the Maintain Proposal component:

1. From the Maintain Proposal page, make a note of the new proposal ID.
2. Click on the version ID link and verify that the same proposal ID appears on the Resource Estimate page.

The application generates a proposal ID when the test is executed, so it is not known ahead of time.

The following example shows one way to automate these steps using a variable. The first step gets the value of the Proposal ID field from the application and stores it to the `&propID` variable. The second step clicks the version ID link, which brings up the Resource Estimate page. The third step verifies the value of the Proposal ID field on the Resource Estimate page against the value saved in the `&propID` variable:

Image: Example of Using a Variable

This example illustrates using a variable to verify the value of a property on a page.

Test	Get_Property	ID=GM_PROP_ID	prop=Value;ret=&PropID	
Link	Click	innerText=V101		
Span	Verify	ID=GM_PROP_ID		&PropID

The length of PTF fields (Value, Recognition, Parameters) is limited to 254 characters. To construct a variable string that is longer than 254 characters, you will need to wrap the string into multiple variables and concatenate them for usage in verification or set value steps.

Image: Example Using Multiple Variables for a String Over 254 Characters

This example illustrates the steps to create smaller variables and then concatenate the string for use in a LongText.Verify step.

Variable	Set_Value	&var1	First 254 character string
Variable	Set_Value	&var2	The remaining characters string
Variable	Set_Value	&var3	concat(&var1&var2)
LongText	Verify	Name=PSU_INSTR_EXP_DESCRLONG	&var3

For additional examples of variable usage, see the “Test Language Reference” chapter, especially the Conditional.If_Then and Variable.Set_Value steps.

Wrapping Variables in Quotes

Variable assignments, comparisons, and functions may use special characters (such as the equals sign, parentheses, and so forth) to manage text string operations. Quotes are necessary in these situations to help PTF distinguish strings that contain these characters from the actual operators themselves.

Quotes are only necessary in the following situations:

- Around text being assigned to a variable in the Recognition field.
- Around text being provided as a parameter of a function (in either the Parameters or Value field).
- Around text being used in a Conditional step in the Recognition field.

Assigning Variables in the Recognition Field

Use the following guidelines when assigning variables in the Recognition field:

- Use quotes around all text strings.
- Use double quotes (""") to return one quotes character. When a final text string result is expected to include leading and trailing quotes characters, the parameters supplied should be lead and trailed by three quotes characters, for example: """"Hello"""".

Note: PTF will return an error (unrecognized or illegal variable format) if the user fails to wrap a text string being assigned to a variable in quotes

This table displays examples of setting the variable &MyVar in the Recognition field and the expected result:

Recognition	Expect Result
&MyVar=&PriorVar	Contents of the &PriorVar variable
&MyVar="&PriorVar"	&PriorVar
&MyVar="Hello"	Hello
&MyVar=""""Hello"""	"Hello"
&MyVar=Hello	error message

Assigning Variables in the Value field

It is a good practice to assign variable values in the Value field (rather than the Recognition field) for the following reasons:

- It simplifies variable assignments.
- It reinforces habits that allow the user to take fuller advantage of dynamic PTF functionality such as Test Case data and reserved words (which are only recognized in the Value field).

Use the following guidelines when assigning variables in the Value field:

- Quotes are not necessary to identify a text string in the Value field for the purpose of a variable assignment step.
- The CONCAT() function should be used to return a text string containing special characters such as ampersands or pound symbols that could be interpreted as a variable or reserved word.
- Double quotes are generally not needed, because text entered into the Value field is interpreted literally.

This table displays examples of setting the variable &MyVar in the value field and the expected result:

Recognition	Value	Expected Result
&MyVar	&PriorVar	Contents of &PriorVar variable
&MyVar	CONCAT("&PriorVar")	&PriorVar
&MyVar	Hello	Hello
&MyVar	"Hello"	"Hello"
&MyVar	""Hello""	""Hello""

Related Links

[Parameters](#)

[Variable](#)

[Conditional](#)

Using Text Strings as Parameters in Functions

The following rules apply when text strings are used as parameters in functions:

- Text strings supplied as parameters for a function must always be wrapped in quotes, regardless of whether they appear in the Recognition field or the Value field.
- The CONCAT() function should be used to return a text string containing special characters such as ampersands that could be interpreted as a variable reference.
- Double quotes (""") should be used to return one quotes character. Consequently, when a final text string result is expected to include leading and trailing quotes characters, the parameter supplied should be lead and trailed by three quotes characters, for example, """"Hello"""".
- To include a line break within a text string, use <NL>. For example:

```
prompt=Newline Message Prompt!!<NL>Line1<NL>Line2<NL>Line3
```

Creates the following text string:

```
Newline Message Prompt!! Line1 Line2 Line3
```

```
Newline Message Prompt!!
Line1
Line2
Line3
```

Note: PTF will return an error (unrecognized / illegal variable format) if the user fails to wrap a text string being supplied as a parameter for a function in quotes.

This table shows examples of the text string used in the recognition field assuming the variable &MyVar is set to "Hello" (&MyVar="Hello") and the expected results:

Recognition	Expected Result
CONCAT("&MyVar = " &MyVar)	&MyVar = Hello
SUBSTR("Hello" 1 5)	Hello
SUBSTR("""Hello """" 1 7)	"Hello"
INSTR("Hello" "e")	2
INSTR("""Hello"""" "e")	3
INSTR("""Hello"""" ""e""")	0
INSTR(Hello e)	PTF returns error message

Using Persistent Variables

If you need to store variables between tests, use persistent variables. PTF stores persistent variables to the database so that subsequent test executions can use them.

Persistent variables are stored in the database keyed by execution option name. Persistent variables can also be keyed by User ID, machine name, or both.

There are two key elements required to use persistent variables:

1. Set the variable action in the test.
2. Set persistent variable options.

Setting Variable Option in the Test

To set the variable option in the test, use the Test Property icon.

1. Create a new test or open an existing test.
2. Click on the Test Properties icon.



The test properties icon is located in the Test information group box.

3. Set the Variable Action field.
 - None (default). The test will not use persistent variables. All variables set in the test are only global to the current test execution.
 - Read. The test can only read persistent variables from existing persistent variables in the PTF test database.
 - Write.

The test can only write persistent variables to the PTF test database.

- Read & Write. The test can read and write persistent variables from and to the PTF test database.

Setting Persistent Variable Options

The persistent variable options are available on the Advanced Options page in Execution Options. You can set the Advanced Options in the PTF client or PIA.

See [Configuring Execution Options in PTF Client](#) or [Defining Advanced Options](#)

Managing Persistent Variables

Use the Manage Persistent Variables page (PSPTTSTPERVAR) to modify or delete persistent variables.

Navigation

PeopleTools, Lifecycle Tools, Test Framework, Manage Persistent Variables

Image: Manage Persistent Variables page

This example illustrates the fields and controls on the Manage Persistent Variables page.

Manage Persistent Variables				
Name	User ID	Machine Name	Name	Value
<input type="checkbox"/> T1B85301	PTFTRN		&ROLE	ADS Designer
<input type="checkbox"/> T1B85301	PTFTRN		&USERID	PTINT

This page will list all the persistent variables stored in the database. The variable is stored when the test writes the variable to the database.

The variable is always stored with the name of the execution option. The User ID and Machine Name are dependent on the selection you made for the persistent variable on the Advanced Options page of the Execution Options.

You can delete persistent variables or modify the variable value.

Example of a Test that uses Persistent Variables

This example shows a test where the variable action was set to Read & Write in the Test Properties. The variable &userid exists in the persistent variable table.

Image: Example of using persistent variables in a test

This example illustrates reading a persistent variable (&userid) and writing a persistent variable (&role).

	Browser	Start_Login			
	Page	Prompt	MAINTAIN_SECURITY.USERMAINT.GBL		update
	Log	Message	Step below will read persistent variable from persistent variable table		
	Text	Set_Value	Name=PSOPRDEFN_SRCH_OPRID		&userid
	Page	PromptOk			
	Page	Go_To	Roles		
	Log	Message	Step below will write persistent variable to persistent variable table		
	Text	Get_Property	Name=PSROLEUSER_VW_ROLENAME\$0	ret=&role;prop=Value	

Using Conditional Logic

This section discusses how to use conditional logic.

Some test scenarios call for conditional logic—special handling based on information gathered from the application during the test. Conditional logic uses a Conditional.If_Then step with a Conditional.End_If step. The If_Then step evaluates a statement. If the expression evaluates to True, the system executes the lines between the If_Then step and the End_If step or the Else step, if it exists. If the expression evaluates to False, the system jumps to the Else step, if it exists, or to the End_If step if there is no Else, and continues execution.

What appears to be simple test instruction, such as the following, may require conditional logic to automate successfully:

- In the Modify a Person page, click the Brazil flag if necessary to expand the Brazil region of the page.

When you record the click on the Brazil flag, PTF creates the following step, which looks deceptively simple:

Image: Example of Step that Requires a Conditional Construct

This example illustrates an image field that requires conditional logic. The conditional logic is described later on this page.

Image	Click	Name=DERIVED_IC_GBL_BRA\$img		
-------	-------	------------------------------	--	--

The problem is that pages like the Modify a Person page typically use the same image to collapse and expand a section. The page may remember which regions are expanded and which are collapsed, so that the next time you run the test, the Brazil section might be expanded when you enter the page, in which case the Image.Click action in the previous example would collapse the Brazil section and potentially cause the test to fail.

The solution is to click the flag image only if the section is collapsed, which requires putting the click action within a conditional If-Then construct.

For example, suppose that you use the Message tool to determine that when the region is already collapsed, the alt property of the flag image is equal to “Expand section Brazil.” Alternatively, when the region is already expanded, the alt property of the same image is equal to “Collapse section Brazil.” You would construct your test such that the click would only occur if the alt property is equal to “Expand section Brazil.” You could do that with the following steps:

Image: Example of Using Conditional Logic

This example illustrates the steps necessary for the conditional logic.

Image	Get_Property	Name=DERIVED_IC_GBL_BRA\$mg	prop=alt,ret=&flagstate	
Conditional	if_Then	&flagstate="Expand section Brazil"		
Image	Click	Name=DERIVED_IC_GBL_BRA\$mg		
Conditional	End_f			

Note: The same rules apply for text strings used in the recognition field as described in the Using Text Strings as Parameters in Functions section.

See [Using Text Strings as Parameters in Functions](#)

Related Links

[Conditional](#)

Handling Application Messages

This section discusses how to handle application messages.

Use the Message Recognition feature to specify how PTF will respond to messages, such as warning messages or error messages, issued by the application being tested.

For example, suppose you have the following manual test instructions:

12. Clear the Calculate all Plans checkbox. If you get the following warning, click OK:
 Warning - Remember all plans must be selected to ensure an accurate 415 limit calculation (48,17)

When you record the test, the step that triggers the message and the step that clicks OK might look like this:

Image: Example of steps that trigger and dismiss a warning

This example illustrates a step that triggers the message and the step to dismiss a warning.

CheckBox	Set_Value	Name=PA_CALCULATION_CALC_ALL_PLANS_cd\$0		N
Button	Click	Name=#COK		

However, some test cases belonging to this test might not deselect the Calculate all Plans check box in the first step. In these cases, the first step would not trigger the warning message and PTF would fail to find the OK button in the second step.

You could use conditional logic to evaluate whether the test case deselects the check box. Alternatively, you could use the Error Handling feature to indicate that PTF should click OK whenever that specific message appears in the application.

To create a message definition, access the Message Recognition dialog box.

1. With a test open, click the Message Recognition icon.
2. Enter the text of the message, or a portion of the text, in the Message field.

The following example uses the message catalog number rather than the full message text.

You can define message definitions at either the test or test case level. Message definitions defined at the test case level take precedence over message definitions defined at the test level.

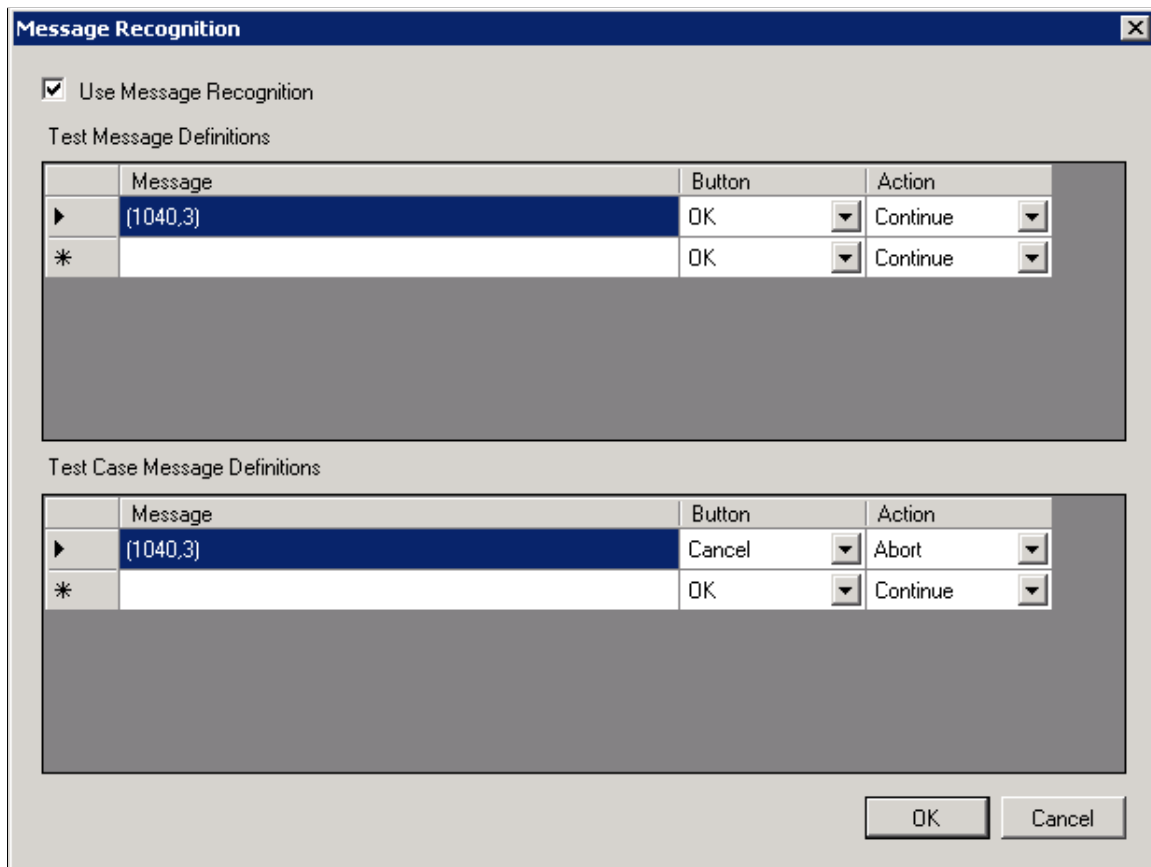
3. Select which button the step should click.
4. Specify what action PTF should take after the button is clicked.
5. To delete a message, select the message line and press the Delete key.

This example shows the Message Recognition dialog box. In this example when the application message (1040,3) error occurs, PTF will click the *Cancel* button and abort the test execution.

Note: Test case message definitions take precedence over test message definitions.

Image: Message Recognition dialog box

This example illustrates the fields and controls on the Message Recognition dialog box. You can find definitions for the fields and controls later on this page.



This table lists the names and definitions of the elements on the Message Recognition dialog box:

Message Enter the message, or portion of the message, displayed by the error. For example, you might use the Message Catalog numbers.

Button Enter the button that PTF should click when it encounters the message.

Valid values are:

- *OK*
- *Abort*
- *Retry*
- *Ignore*
- *Yes*
- *No*
- *Cancel*

Action

Select the action that PTF will take.

Valid actions are:

- *Continue*: Log an Info message and continue processing.
- *Abort*: Log a Fail message and stop test execution.

Interpreting Logs

This section discusses how to interpret logs.

This table lists and describes common log messages:

Status	Message	Description
Fail	Access is denied. Please check browser settings.	PTF is not able to read information from your browser. Check browser settings to make sure your browser is configured properly. See Configuring Browser Settings .

Status	Message	Description
Fail	Object not found in the page, or access is denied to the Frame.	<p>If all of your steps that involve setting values, verifying values, or getting properties return a failure with this message in the log, then it is likely PTF is having trouble interacting with your browser in general.</p> <p>Check browser settings to make sure your browser is configured properly.</p> <p>See Configuring Browser Settings.</p> <p>If this log message appears sporadically throughout your log, it could mean that objects within your application have been removed or renamed.</p> <ul style="list-style-type: none"> • Check the application or the screen shots in the log associated with the failed step to see if the object still appears on the page. • If the object appears where expected on the page, use the Message tool to compare the names and IDs of the objects on the page with those in the Recognition field of the step. <p>See Using the Message Tool.</p> <ul style="list-style-type: none"> • If object names or IDs have changed since you recorded your tests, consult with your PTF administrator about the possible need to run maintenance on your tests. <p>See Interpreting Test Maintenance Reports</p>

Incorporating Scroll Handling

This section discusses how to incorporate scroll handling in PTF tests.

Data on a PeopleSoft component is organized hierarchically using rowsets, or scrolls, and rows.

A scroll can be implemented as a scroll area or a grid. In scroll areas, the fields appear on the page in a freeform manner. In grids, fields appear as columns similar to those on a spreadsheet. Individual rows of data within a scroll or grid are uniquely identified by a set of one or more fields, or keys.

PTF references a field on a scroll by the field name and the row number. The PTF scroll handling feature enables a test to identify a row number at test execution time based on the keys for that row.

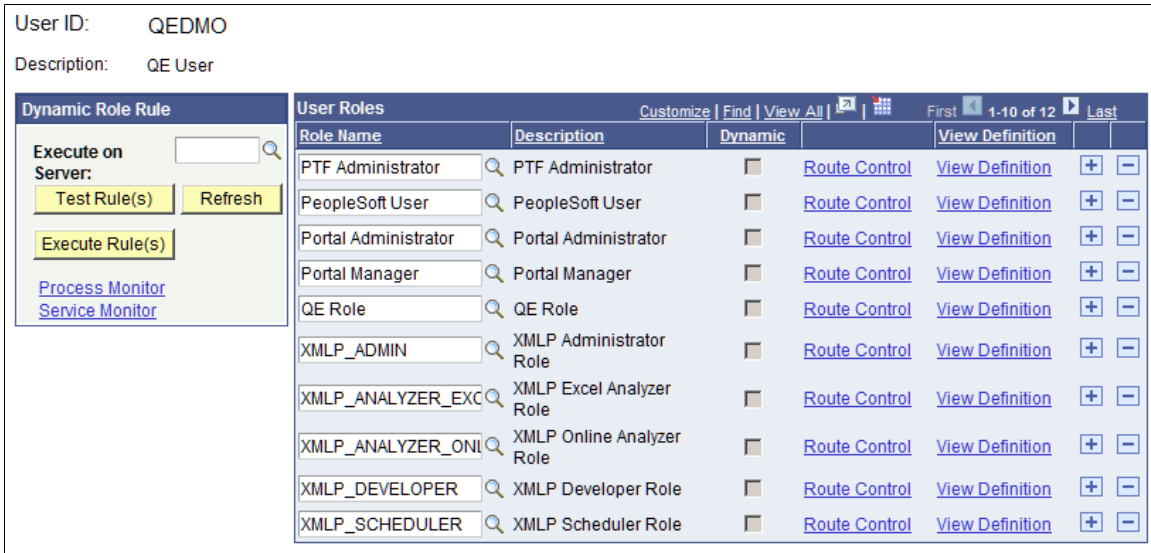
For example, suppose you have a test requirement that says:

```
12. Verify that the QEDMO user profile has the PTF Administrator role.
```

Here is an example of the Roles page for the QEDMO user profile:

Image: Example of the User Profile - Roles page

This example illustrates the fields and controls on the Example of the User Profile - Roles page.



When you record the test, PTF generates a step similar to the following example:

Image: Test step to verify a field on a scroll area

This example illustrates the test step to verify a field on a scroll area. You can find additional information on this step later on this page.

Scroll ID	Type	Action	Recognition	Value
	Text	Verify	Name=PSROLEUSER_VW_ROLENAME\$0	PTF Administrator

The step uses two elements in the Name= parameter in the Recognition column to reference the Role Name field: the name of the field (PSROLEUSER_VW_ROLENAME) and its row position index (\$0). A row position index is composed of a dollar sign (\$) and an integer. The integer count starts at zero, so indexes for a scroll containing 11 rows are \$0 through \$10.

This test will work as recorded until something changes the grid position of the row that contains PTF Administrator. For instance, if another row is inserted before PTF Administrator, the PTF Administrator row position index changes to \$1, and the test fails. The same problem occurs if the grid is sorted differently, or if a test case tests a different user profile, such as QEMGR.

Using a Dynamic Position Index

You can use the Scroll.Key_Set action and a Scroll.Action step to locate a row by key and generate a dynamic position index variable. Then you can use the dynamic position index variable to reference a row or a field reliably and repeatedly because the variable is regenerated each time the test is run.

For example, instead of looking at the first row, PTF looks for the row where the key equals “PTF Administrator”.

If the value exists in the scroll, the test finds it, takes the specified action, and returns the position index.

If it does not find the value in the first displayed set of rows, PTF clicks the Show Next Rows icon on the scroll and continues searching until it has found the key value or searched all rows on all pages of the scroll.

Follow these steps to use a dynamic position index variable:

1. Create Key_Set steps.
2. Create an Action step.
3. Use the index variable for other steps.
4. Specify the Scroll ID.

Note: You can accomplish all of the steps in this section during recording using features of the PTF Recorder Tool Bar. The manual process is provided to illustrate how information is shared between the different scroll-handling step types and how it can be manipulated manually after recording, if necessary. See [Using the PTF Test Recorder](#).

Creating Key_Set Steps

Create one Key_Set step for each field in the scroll key. If the key consists of three fields, create three Key_Set steps and specify key values for each of the three fields.

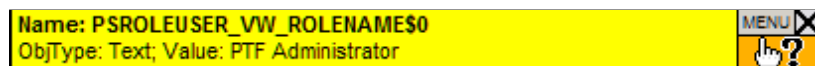
You can insert Scroll steps during recording or you can add them afterward. The following process explains how to insert and modify key steps manually. Even if you record Scroll steps, you may need to modify them using some of these concepts.

Key_Set requires two parameters in the Recognition column; `Type=` and `Name=`. You can get these values by recording a step with the PTF recorder and then manually modifying the recorded step or by copying the information from the application using the Message tool and pasting into a step.

Specify the field value in the Value column. You can use the PTF Recorder or Message tool to get the field value as well.

Image: Example of the Message tool

This example illustrates the Message tool with recognition data for the Role Name field.



Here is an example of a test step that references a row on a scroll. This step checks for the existence of *PTF Administrator* in the Role Name field:

Image: Example of a step that verifies a field on a row

This example illustrates a test step that references a row on a scroll.

Scroll ID	Type	Action	Recognition	Value
	Text	Verify	Name=PSROLEUSER_VW_ROLENAME\$0	PTF Administrator

This is one way to convert the step in the example to a Scroll.Key_Set step:

1. Change the Type to *Scroll*.
2. Change the Action to *Key_Set*.

3. Add *Type=Text*; to the Parameter column.
4. Leave the Name parameter in the Recognition column, but remove the row position index (\$0).

This signals PTF that the actual row number for the key is not yet known.

Image: Example of a Scroll.Key_Set step

This example illustrates a Scroll.Key_Set step that sets the key to PTF Administrator.

Scroll ID	Type	Action	Recognition	Parameters	Value
1	Scroll	Key_Set	Name=PSROLEUSER_VW_ROLENAME	type=Text	PTF Administrator

This step defines the key value, but PTF does not take an explicit action on the page based on the key until it executes an Action step.

Creating an Action Step

Create an Action step, based on what action you want to take on the row, such as update, insert, select, and so on. All of the available actions are detailed in the PTF Language Reference.

In this example, you want to select a row, so enter *sel* in the Value field.

The Action step attempts to locate a row defined by Key_Set. If a row is found, it returns the index of the row. Use the *ret=* parameter of the Action step to populate an index variable with the row index value.

Image: Example of an Action step

This example illustrates an action step.

Scroll ID	Type	Action	Recognition	Parameters	Value
1	Scroll	Action		ret=&Scroll1	sel

Using the Index Variable

Now that you have the row position stored in the index variable, you can use that value to reference other fields on that row.

For instance, suppose you want to verify the Dynamic checkbox. You can use the PTF Recorder or the Message tool to get its name and position and create a step similar this one:

Image: Example of step using positional reference

This example illustrates a step with a positional reference.

CheckBox	Verify	Name=PSROLEUSER_VW_DYNAMIC_SW\$0		N
----------	--------	----------------------------------	--	---

This step has the problem that the positional reference is static, so it won't work if the position changes. To fix that, replace the position index with the index variable.

This example shows a step that uses an index variable following the steps that locate the key and set the index variable:

Image: Example of a step using an index variable

This example illustrates a step using an index variable.

Scroll ID	Type	Action	Recognition	Parameters	Value
1	Scroll	Key_Set	Name=PSROLEUSER_VW_ROLENAME	type=Text	PTF Administrator
1	Scroll	Action		ret=&Scroll1	sel
	CheckBox	Verify	Name=PSROLEUSER_VW_DYNAMIC_SW&Scroll1		N

Now whenever the test is run, the index variable is updated dynamically by the Key_Set and Action steps and the positional reference is accurate.

Specifying the Scroll ID

Assign a Scroll ID to group Scroll actions for each scroll. Use a different scroll ID and scroll variable for each different scroll area. You can assign any integer you like, as long as it is unique.

Scroll ID is a required field for Scroll actions.

If you are taking multiple actions in the same scroll, using the same scroll ID and scroll variable improves performance over using a new scroll ID.

In this example, the test defines two Action steps that both act on the same scroll. The first action verifies that the user profile does not contain the PTF Administrator role. The second action verifies that the user profile does contain the PTF User role. You would assign the same Scroll ID number to all four of the Scroll steps because they all act on the same scroll.

Image: Example of assigning Scroll IDs to Scroll steps

This example illustrates assigning scroll IDs to scroll steps.

Scroll ID	Type	Action	Recognition	Parameters	Value
	Page	Go_To	Roles		
1	Scroll	Key_Set	Name=PSROLEUSER_VW_ROLENAME	type=Text	PTF Administrator
1	Scroll	Action		ret=&Scroll1	not
	Log	Message		Scroll1 index variable = &Scroll1	
1	Scroll	Key_Set	Name=PSROLEUSER_VW_ROLENAME	type=Text	PTF User
1	Scroll	Action		ret=&Scroll1	sel

Related Links

[Scroll](#)

Calling Tests

This section provides an overview of calling tests and discusses how to use library and shell tests.

- Use library tests.
- Use shell tests.
- Use parameters with library tests.
- Share test assets.

Understanding Calling Tests

If a test uses a sequence of steps repeatedly, you may want to isolate the repetitive sequence of steps and move them to another, smaller test. Doing so enables you to call the steps repeatedly and also make them available to other tests that use the same sequence of steps.

Moving shared test steps to a distinct test (a called, or child, test) provides these benefits:

- Reduces the amount of recording or development you need to do.
- Reduces the amount of debugging you need to do.
- Reduces the effects of development changes that require manual updates to existing tests.

You can use the Test.Exec action to call any other test, but you may be able to manage and identify the relationships between calling and called tests more easily on the PTF Explorer tree if you use library tests and shell tests.

Using Library Tests

A library test cannot be executed by itself. It must be called by another test.

To make a test a library test, select the Library Test check box in the Test Editor.

Complete these steps to create a library test from an existing test:

1. Open an existing test.
2. Identify repetitive steps within the test, copy them from the existing test, and paste them into a new test.
3. Select the Library Test check box on the new test.
4. Save the library test.
5. Remove the repetitive steps from the original test and replace them with a step that uses a Test.Exec action to call the new test.
6. Save the original test.

Using Parameters with Library Tests

Parameters enable you to pass dynamic values from a calling test to a library test.

To use parameters:

1. Define the parameters in the library test.
2. Assign values to the parameters in the calling test.
3. Use the parameters as you would text strings in the library test.

To define parameters in a library test:

1. Select the Library Test check box in the Test Properties dialog box.

2. Enter the names of parameters in the Parameter List.

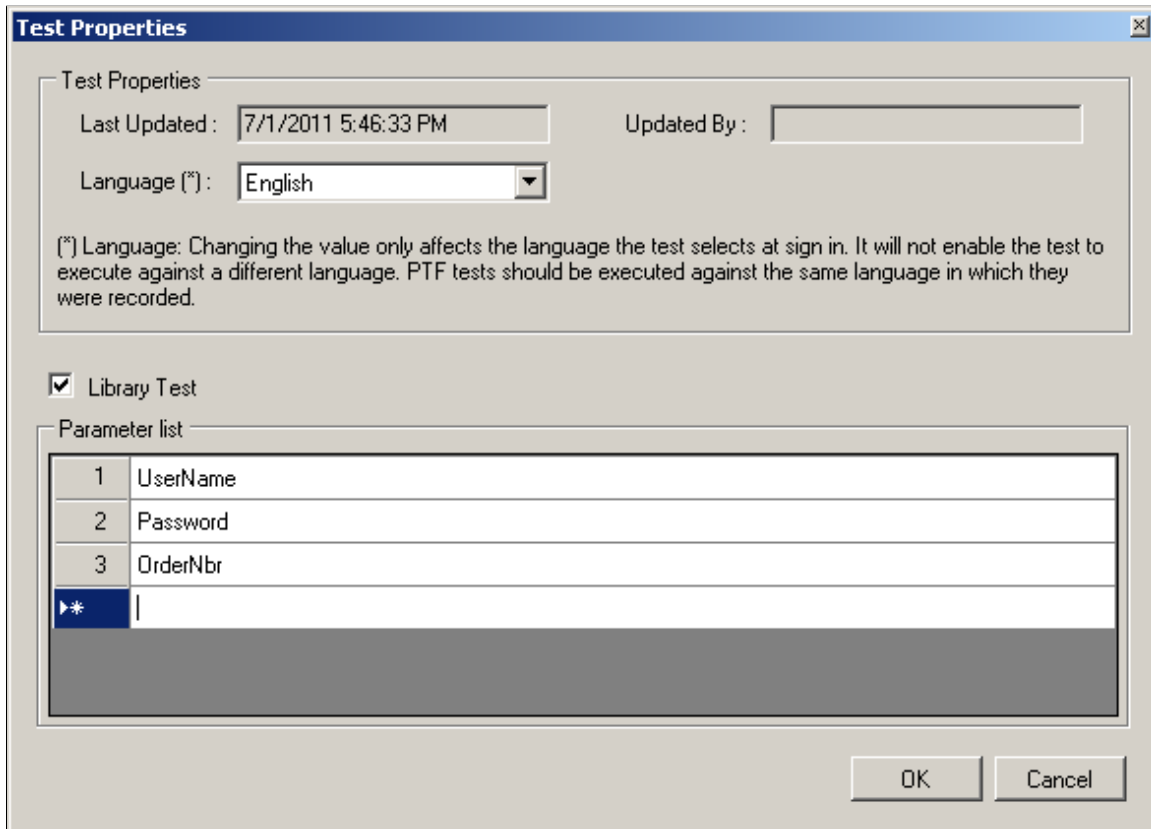
Parameter names can be up to 254 characters, and can contain only letters, numbers, and underscores.

3. Click OK to dismiss the dialog box.

The following example shows a parameter list:

Image: Test Properties dialog box with a parameter list

This example illustrates the fields and controls on the Test Properties dialog box with a parameter list.



In the calling test, place parameter assignments in Test.Exec steps in the Parameters field, separated by semicolons, with no spaces. You can place multiple parameter assignments, separated by semicolons. The last semicolon is optional.

Note: The semicolons must not be followed by spaces.

The following example shows a Test.Exec step that calls a library test and passes three parameters.

Image: Test.Exec step with parameters

This example illustrates a Test.Exec step with parameters.

Type	Action	Recognition	Parameters	Value
Test	Exec	PB_PARMS_LIB	UserName=QEDMO.Password=QEDMO.OrderNbr=1001	DEFAULT

In the library test, place a %param.parameter_name% construct wherever you want to use a parameter. The parameter is replaced at runtime by the text assigned in the calling test.

Image: Library test steps with parameters

This example illustrates library test steps with parameters.

Type	Action	Recognition	Parameters	Value
Text	Set_Value	Name=userid		%param.UserName%
Pwd	Set_Value	Name=password		%param.Password%
Variable	Set_Value	#OrderNbr=%param.OrderNbr%		

Note: Parameters in library tests need to be assigned a value in the calling test, because variables are not automatically initialized. The parameters without a variable initialized use default PeopleSoft values on the page.

If the user makes a call to the library without passing a value for the parameter, the parameter value will be the parameter name. For example, if the parameter is *UserName* and the value is not set, the value would be %param.UserName%, not blank.

Using Shell Tests

To create a shell test, while in PTF Explorer select Create, Shell Test.

A shell test is a type of test that is meant to be used primarily to call other tests. For this reason, a shell test only supports these actions:

- Execution actions - modify the behavior of tests during execution. Execution actions include Skip_PageSave, Skip_RunRequest, Skip_Login, StopOnError, and Set_Options.
- Test.Exec - calling other tests. Test.Exec enables you to call multiple test cases with a test and, using the #Ignore reserved word, skip the test call for certain test cases.
- DataMover.Exec - calling data mover scripts.
- Query.Exec - running queries.
- Log.Message and Log.Screenshot.
- Variable.Set_Value - manipulating variables.

Organizing component tests into shells enables you to identify large business-process oriented type tests (that is, the type that cross multiple components and online activities). The steps available in shell tests are intentionally limited in order to represent high-level business process flows through called test routines.

PTF variables are global, so you can set a variable in the shell test and use it in the called tests, or you can set a variable in a test and use it in the shell test or other called tests.

Sharing Test Assets

Often, test developers, application developers, testers, and others collaborate to develop tests. PTF enables you to send links to tests, test cases, and logs to other users, saving them from having to navigate through PTF Explorer to locate them.

To share the location of a test asset:

1. Copy the link to the clipboard.

- In PTF Explorer, highlight the name of a test asset and select Edit, Copy Link to Clipboard.
 - In the Log Viewer, with a log open, select Log, Copy Link to Clipboard.
2. Paste the link text into a message or document to send to another user.
 3. The recipient copies the text and selects Window, Quick Open.

The Quick Open feature is available in PTF Explorer, Test Editor, and Log Viewer.

The system automatically copies the link for the asset to the Quick Open dialog box.

4. The recipient clicks OK to open the asset.

You can also use Copy Link to Clipboard with the Quick Open feature to locate a folder in PTF Explorer.

Chapter 6

Administering PTF

Managing PTF Logs

This section describes how to use PTF Log Manager to manage test logs.

Understanding Log Manager

Over time, as you run tests, you will create a number of test logs. Because they reside in your application database, test logs, especially those containing many screen shots, can affect the storage demands on your database. PTF Log Manager enables you to minimize this demand and remove clutter from PTF Explorer.

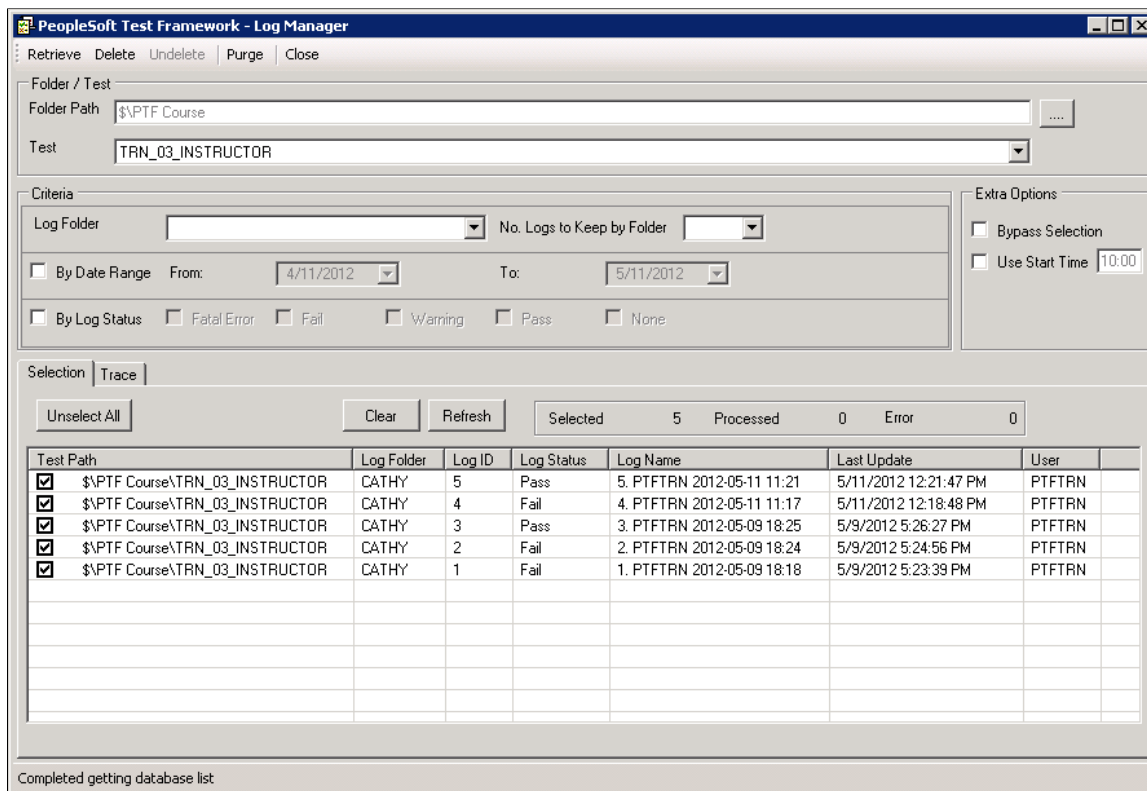
To help you decide which logs to remove from your database, Log Manager lists log entries from the test environment based on the criteria you specify. If all the fields are empty, then Log Manger lists all the logs in an environment that were created within the specified date range. The default date range is the current date.

To access the Log Manager, select Tools, Log Manager.

Note: Only an administrator (a user ID with the PTF Administrator role) is able to open Log Manager.

Image: Example of PTF Log Manager

This example illustrates the fields and controls on the PTF Log Manager page. You can find definitions for the fields and controls later on this page.



Using Log Manager Toolbar

The log manager toolbar has the following options:

Retrieve

Select to populate the selection pane based on the specified criteria.

Note: The user can cancel the retrieval process at any time by pressing the Abort button, and thus getting a partial result.

Delete

Select to delete the selected logs.

Undelete

Select to cancel logs marked for deletion. When you click Delete, the selected logs are only marked for deletion. They are not removed from the database until you click Purge Log. Until then, you can undelete logs.

Purge

Select to remove selected logs from the database if they are marked for deletion.

Close

Select to close log manager.

Using Log Manager Fields

Log Manager has these fields:

Folder / Test

Folder Path

Browse to a folder in PTF Explorer. If a folder is specified, the system retrieves only the logs in that folder, including subfolders. If no folder is specified, the system retrieves all logs in the environment.

Test

The system retrieves the logs associated with the selected test. The drop-down list is restricted to the tests in the folder specified in Folder Path.

Criteria

Log Folder

Select a log folder. Log folders associated with the selected test are available in the list.

No. Logs to Keep by Folder (number of logs to keep by folder)

Select the number of logs to include in the folder. For instance, if the folder contains six logs and you specify three, then the three newest logs will be retained (that is, they will not be included in the list as candidates for deletion) and the three oldest logs will be retrieved. Select zero to retrieve all logs.

By Date Range

The system retrieves only logs created within the date range.

By Log Status

When you select this check box, the statuses are available. Select the status(es) to list and click the Refresh button in the Selection pane. The results are updated in the Selection pane.

Extra Options

Bypass Selection

Select to perform the selected action on all log entries listed according to the criteria, regardless of user selections.

Use Start Time

Select and enter a value to start the process at the designated time.

Using the Selection Pane

The selection pane displays the logs based on the selection criteria. By default all the logs are selected.

When you click Retrieve, the system populates this pane with logs based on the criteria you specified. Using the check boxes, select the logs that will be processed when you click Delete, Undelete, or Purge.

The selection pane contains a status bar that displays the number of logs selected, processed and errors, and the following buttons:

Unselect All

Click to unselect all logs.

Clear

Click to clear the selection list.

Refresh

Click to refresh the selection after changing the criteria.

Note: The refresh is based on the new criteria only. If you change the folder path or test, you must use do a new retrieve.

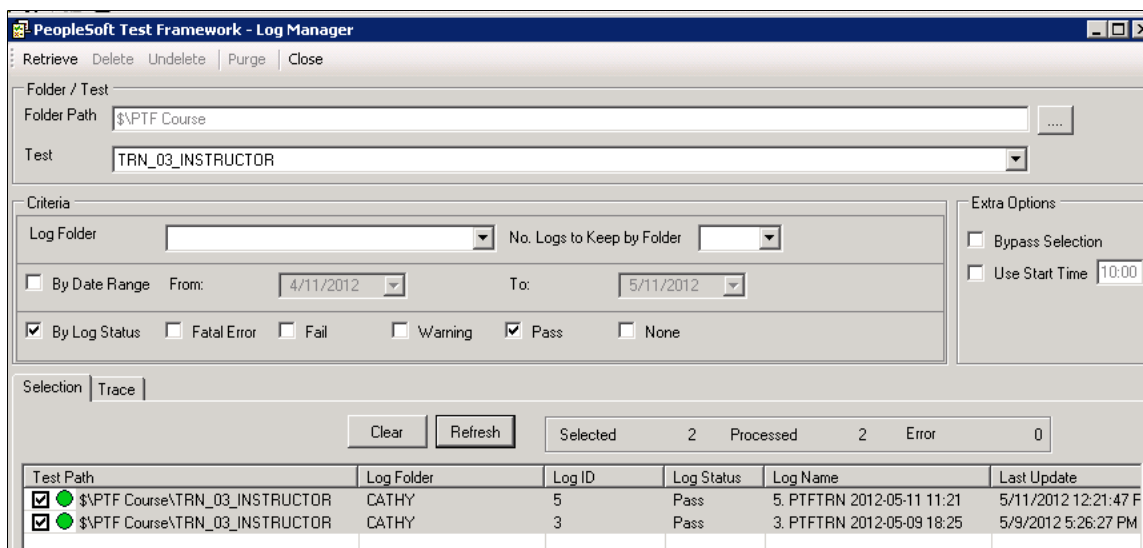
All of the columns in the selection section are resizable.

Example Purge

When you select logs in the selection pane and then select Purge from the toolbar, the selection pane status is updated and the focus is set on the log lines being processed.

Image: Example of PTF Log Manager after Purge

This example illustrates PTF Log Manager after Purge.

**Using the Trace Pane**

The trace pane displays a history of processing actions for this session.

Upgrading Tests

The Tests PT Upgrade tool enables you to update tests from previous versions when you upgrade to a newer version of PeopleTools. When you launch the PTF client, it checks for tests from prior versions, and if they exist, a prompt window appears that enables you to run the upgrade tool. You can run the upgrade tool when prompted, or defer it to a later time. You can upgrade all tests, or just those within a specific folder of the PTF Explorer tree. The Tests PT Upgrade tool makes any necessary syntax changes in the test steps, and then saves the test files. If you defer the upgrade, and you open a test from a prior release, it is automatically updated to the new release at that time.

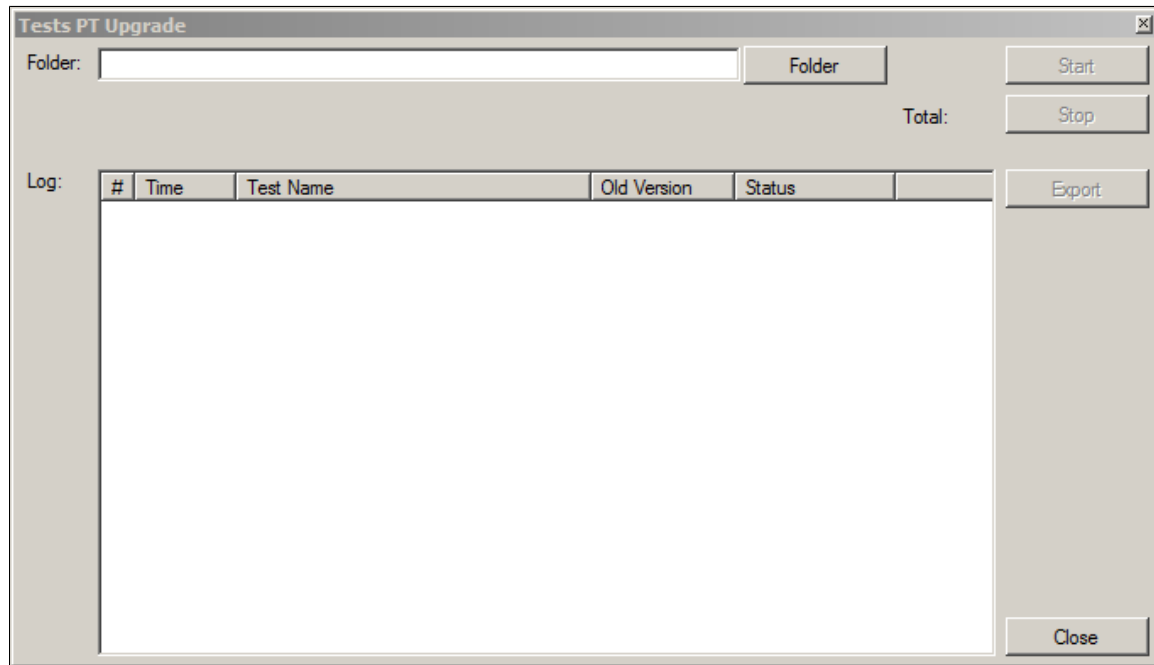
Note: You cannot reverse the upgrade process.

Using the Tests PT Upgrade Tool

Access the Tests PT Upgrade dialog box (Tools, Tests PT Upgrade, or click Yes if you are prompted to upgrade tests when you launch the PTF client).

Image: Tests PT Upgrade Dialog Box

This example illustrates the Tests PT Upgrade dialog box.



Folder

Specify the folder of the PTF Explorer tree that contains the tests you want to upgrade.

You can click the Folder button to open a view of the PTF Explorer tree and navigate to the desired folder, or enter the folder path directly into the Folder field. Use the '\$\' characters to indicate the root level of the tree.

Total

After you specify the folder, this lists the number of tests to be upgraded.

After you click the Start button, a progress bar appears next to the total, to provide a visual indicator of how many tests have been processed. The start time and an estimated end time appear below the progress bar.

Start

Click to upgrade the tests in the specified folder. This option is not available until the Folder field has been populated.

Stop

Click to interrupt the upgrade process, once it has started. This action is available only after the upgrade process has started.

Log

This grid is populated as tests are upgraded. It lists each test that is updated, the time it was updated, the version it was updated from, and its upgrade status.

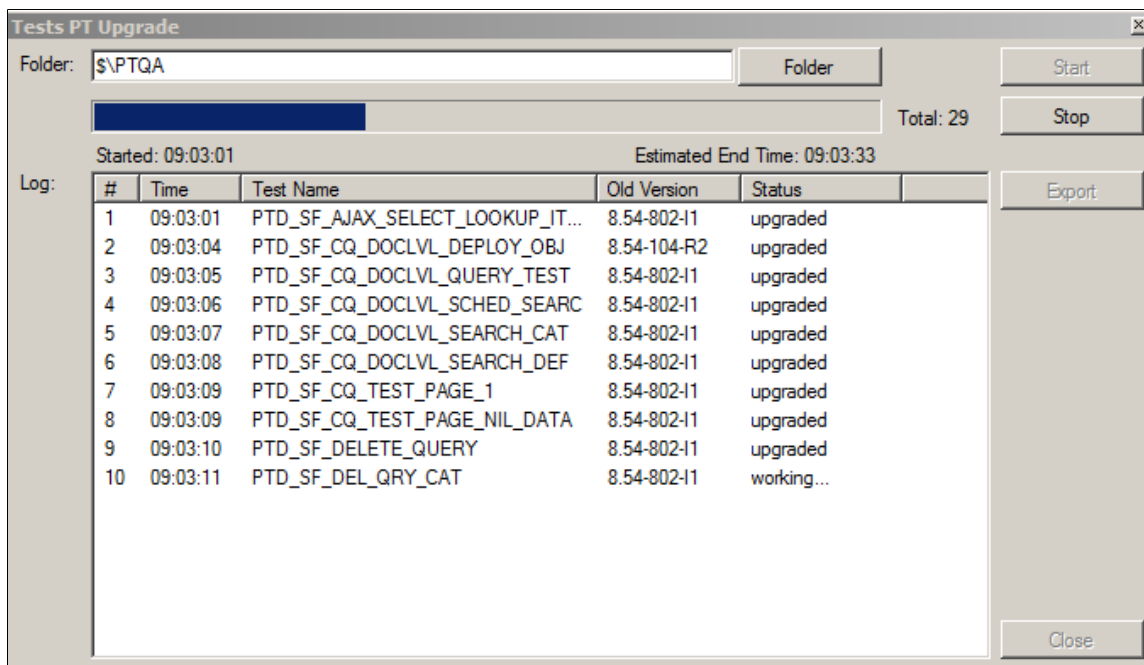
Export

Click to save the test upgrade log to a text file.

This action is available only after all tests are upgraded, or when the upgrade process has been stopped.

Image: Tests PT Upgrade Dialog Box - Processing

This example shows the Tests PT Upgrade dialog box during the upgrade process.



Working with Application Designer Projects in PTF Client

The PTF client enables you to inspect and maintain the test contents of an Application Designer project, and complete the following tasks:

- Create new Application Designer projects.
- Open existing Application Designer projects.
- Insert tests into Application Designer projects.
- Remove tests from Application Designer projects.
- Set the upgrade action type for the tests within Application Designer projects.

Access to this feature is limited to PTF Administrators. In addition, the appropriate PeopleTools security roles and permissions necessary to administer Application Designer projects are required. Only test-related data within a project is modified; any other content that exists within the project is not affected.

Note: This section assumes you are already familiar with Application Designer projects. For detailed information about Application Designer Projects, see *Application Designer Developer's Guide*.

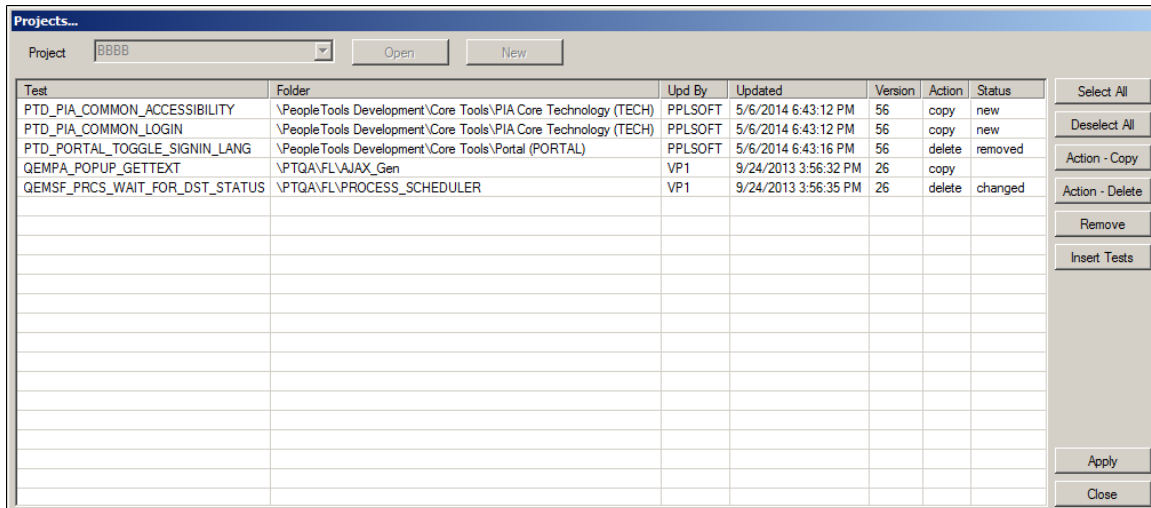
Managing Application Designer Projects in PTF Client

Use the Projects dialog box to manage Application Designer projects within the PTF client.

To access the Projects dialog box, select the @<PTF Environment Name>, Projects menu command.

Image: Projects Dialog Box

This example illustrates the fields and controls in the Projects dialog box.



Use these fields and controls to open or create a project:

Project

Select a project.

Open

Click to open the selected project. This option is not available if the Project field is blank, or a project with pending changes is open.

New

Click to create a new project. At the prompt, enter a unique project name.

This option is not available if a project with pending changes is open.

Note: •New projects are not saved until a test object is inserted and you click the Apply button.

The grid area serves as a worksheet that lists the following information about the project's tests.

Test

The test name.

Folder

The name of the folder the test belongs to.

Upd By

The user ID of the person that last updated the test.

Updated

The date and time the test was last modified.

Version

The test version.

Action	<p>The upgrade action associated with this project test record, either <i>Copy</i> or <i>Delete</i>.</p> <p>By default, the action is set to <i>Copy</i> when the test is initially inserted into the project.</p> <p>To change the upgrade action, click the Action - Copy or Action - Delete button.</p>
Status	<p>The status of the project test record. This updates automatically while you interact with the project using this dialog box to show the changes made during the current session. Once you apply the changes, the status resets.</p> <p>Values are:</p> <ul style="list-style-type: none"> • <i>New</i> Indicates the test record was inserted during this session. • <i>Changed</i> Indicates the upgrade action was changed during this session. • <i>Removed</i> Indicates the test record was set to Removed during this session. • No entry (blank). Indicates no changes were made during this session.

Initially, when you open the Projects dialog box, the grid is empty. It populates when you:

- Open a project that contains tests.
- Add tests by using the Insert button.

Use these buttons to interact with the project worksheet.

Select All	Click to select all of the tests in the grid.
Deselect All	Click to deselect all of the tests in the grid.
Action - Copy	Click to set the upgrade action type to copy.
Action - Delete	Click to set the upgrade action type to delete.
Remove	Click to remove selected tests from the project.
Insert Tests	Click to insert tests into the current project.

A window appears that enables you to select the tests from to insert. Currently active filters are applied to the view. You can

select multiple tests or folders. If a parent folder is selected, tests from its associated child folders are included.

Apply

Click to commit all pending changes to the project. The Projects dialog box remains open.

Close

Click to close the dialog box. If there are any pending changes, you are prompted to save.

Important! Project content is not dynamic; it is a snapshot of the test data at the time it was inserted into the project. If you modify a test after you have included it in a project, you must re-insert that test into the project in order for the project to contain the revised test data.

Creating a New Project

To create a new project:

1. Select the *@<PTF Environment Name>*, Projects menu command.

The Projects dialog box opens.

2. Click the New button.
3. At the prompt, enter a name for the new project, and click OK.

The Projects dialog box reappears. The Project field displays the new project name, and becomes unavailable. The Open and New buttons are disabled.

Opening an Existing Project

To open an existing project:

1. Select the *@<PTF Environment Name>*, Projects menu command.

The Projects dialog box opens.

2. Select a project from the Projects list box.
3. Click the Open button.

If any tests are currently associated with the project, they appear in the grid.

Inserting Tests into a Project

To insert tests into a project:

1. Select the *@<PTF Environment Name>*, Projects menu command.

The Projects dialog box opens.

2. Open an existing project, or create a new project.
3. Click the Insert button.

The PTF Explorer tree appears in the PTF Suite window.

4. Expand folders as needed to access and select the tests to insert (use Ctrl-Click to select multiple tests).
5. Click OK.
6. Click Apply to insert the tests into the project.

Removing Tests from a Project

To remove tests from a project:

1. Open an existing project.
2. Select one or more tests in the grid.
3. Click Remove.
4. Click Apply.

Setting the Upgrade Action Type for Test(s) in a Project

To set the upgrade action type for test(s) in a project:

1. Open an existing project.
2. Select one or more tests in the grid.
3. Click Action - Copy or Action - Delete to set the desired upgrade action type for the selected test(s).
4. Click Apply.

Migrating PTF Tests

Because PTF tests and test cases are PeopleTools managed objects, they can be copied from one database to another in the same way as other PeopleTools objects, such as record definitions, SQL definitions, and PeopleCode programs.

You can create a project (in either the PTF client or in Application Designer) that includes tests and test cases, and export the project to another database using the Copy Project tool in Application Designer.

You can also include tests and test cases in an upgrade project.

Related Links

"Inserting Definitions Into Projects" (PeopleTools 8.55: Application Designer Developer's Guide)

Performing Mass Updates

This section describes how to use the PTF Mass Update tool.

Understanding Mass Update

As the applications you are testing are updated, you will find various situations where many of your tests need to be modified. The Mass Update tool provides this capability, enabling you to perform mass updates to your tests.

For example, suppose in a new release, a component name has changed, and every test you have that references that component must now be modified in order to execute successfully. Using the Mass Update tool, you retrieve the impacted tests, search for steps that include that component and replace the previous component name with the new name.

Mass Update Steps

To perform a mass update, complete these steps:

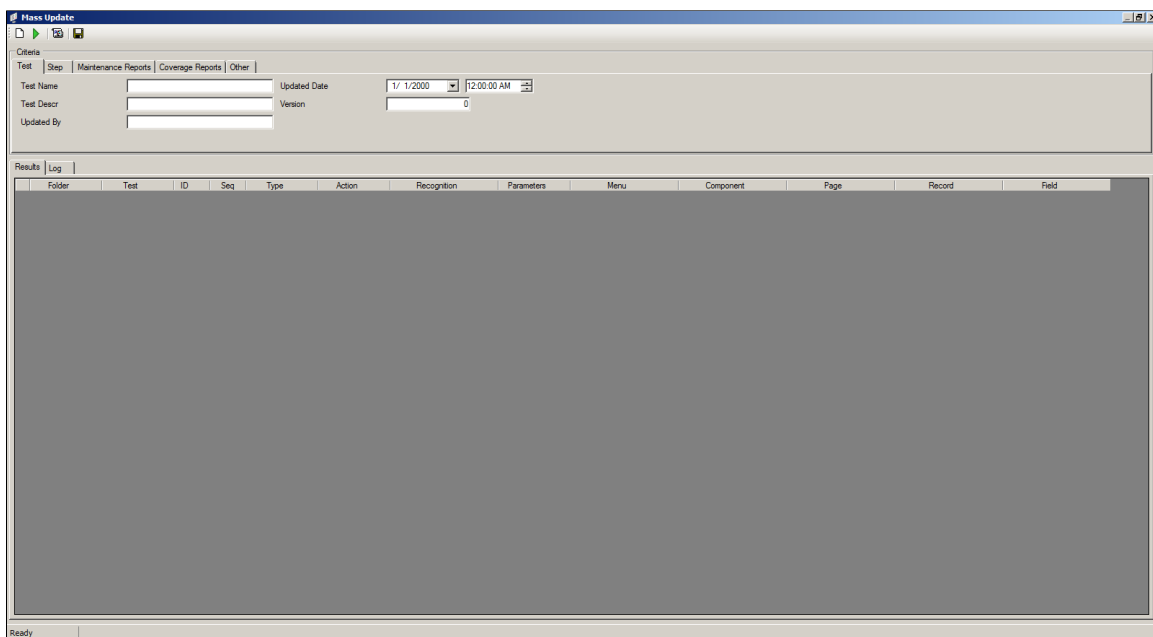
1. Open the Mass Update page.
2. Define the filter criteria.
3. Apply the filter to retrieve the test steps.
4. Find/replace the test step values that need to be modified.
5. Save.

Using the Mass Update Page

To access the Mass Update page, select Tools, Mass Update.

Image: Mass Update Page

This example illustrates the fields and controls on the Mass Update page.







When you first open the Mass Update page, the Results grid is empty; no test records appear in the grid.

When you use the mass update tool, you define and apply a filter to retrieve tests that match specific criteria, returning the results in a grid format similar to the Test editor, but including steps for *multiple* tests. The grid includes two additional columns: Folder and Test so you can identify which test the steps are from. Once the grid is populated, you can use a find/replace dialog box to search for specific test step field values, modify the steps, and save your changes.

Mass Update Toolbar

The Mass Update Toolbar includes the following icons:

- | | |
|---|---|
|  | Click to clear all criteria fields. |
|  | Click to apply the filter and populate the grid with the matching test steps. |
|  | Click to open the Find and Replace dialog box. |
|  | Click to save all modified test steps. |

Mass Update – Criteria Group Box

Use the Criteria group box to define the filter criteria. When you open the Mass Update window, if filters for the PTF Explorer Tree have been defined, they are included automatically, but you can change the filter criteria as needed; the same filter fields are used in Mass Update and the PTF Explorer Tree. For best performance, specify more criteria to reduce the number of test steps that are retrieved.

The criteria types are grouped by the following tabs: Test, Step, Maintenance Reports, Coverage Reports, Other. Click each tab to define the criteria fields for that group.

For details about the filter criteria fields, see [Defining and Applying Filters](#).

To include test cases, select the Other tab and check the Include Test Cases check box.

Mass Update – Results Grid

Use the Results grid as a worksheet area for retrieving and modifying test steps. Initially, this grid is empty. To populate the grid, define the filter criteria, then click the Retrieve icon. A status of `Processing...` appears in the status bar during retrieval. Once processing is complete, the status bar states the number of records retrieved.

Image: Mass Update Page – Results Grid

This example illustrates the Results grid after applying the filter.

Folder	Test	ID	Seq	Type	Action	Recognition	Parameters	Menu	Component	Page	Record	Field
\\PeopleTools De...	PTD_PIVOTGRI...	21	13	ComboBox	Set_Value	Name=PTPG_COL_VIEW_P...		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_OPT	PTPG_COL_VIEW	PTPG_GRIDAXIS
\\PeopleTools De...	PTD_PIVOTGRI...	22	14	Button	Click	Name=PTORM_NEXT_FB		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_OPT	PTPG_WORKREC	PTPG_NEXT_FB
\\PeopleTools De...	PTD_PIVOTGRI...	23	15	Page	Save			PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_DISP		
\\PeopleTools De...	PTD_PIVOTGRI...	24	16	Span	Click	ID=PT_CHART_SCROLL1...		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_DISP		
\\PeopleTools De...	PTD_PIVOTGRI...	25	17	CheckBox	Set_Value	Name=PT_CHART_SCROLL...		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_DISP		
\\PeopleTools De...	PTD_PIVOTGRI...	26	18	CheckBox	Set_Value	Name=PT_CHART_SCROLL...		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_DISP		
\\PeopleTools De...	PTD_PIVOTGRI...	27	19	Button	Click	Name=PT_CHART_SCROLL...		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_DISP		
\\PeopleTools De...	PTD_PIVOTGRI...	28	20	Span	Click	ID=PT_CHART_SCROLL1...		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_DISP		
\\PeopleTools De...	PTD_PIVOTGRI...	29	21	CheckBox	Set_Value	Name=PT_CHART_SCROLL...		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_DISP		
\\PeopleTools De...	PTD_PIVOTGRI...	30	22	CheckBox	Set_Value	Name=PT_CHART_SCROLL...		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_DISP		
\\PeopleTools De...	PTD_PIVOTGRI...	31	23	Button	Click	Name=PT_CHART_SCROLL...		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_DISP		
\\PeopleTools De...	PTD_PIVOTGRI...	32	24	Span	Click	ID=PT_CHART_SCROLL1...		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_DISP		
\\PeopleTools De...	PTD_PIVOTGRI...	33	25	CheckBox	Set_Value	Name=PT_CHART_SCROLL...		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_DISP		
\\PeopleTools De...	PTD_PIVOTGRI...	34	26	CheckBox	Set_Value	Name=PT_CHART_SCROLL...		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_DISP		
\\PeopleTools De...	PTD_PIVOTGRI...	35	27	Button	Click	Name=PT_CHART_SCROLL...		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_DISP		
\\PeopleTools De...	PTD_PIVOTGRI...	36	28	Button	Click	Name=PTORM_PREV_FB		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_DISP	PTPG_WORKREC	PTPG_PREV_FB
\\PeopleTools De...	PTD_PIVOTGRI...	37	29	Radio	Set_Value	Name=PTPG_WORKREC_P...		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_OPT	PTPG_WORKREC	PTPG_DISPLAYOPT
\\PeopleTools De...	PTD_PIVOTGRI...	38	30	Button	Click	Name=PTORM_NEXT_FB		PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_OPT	PTPG_WORKREC	PTPG_NEXT_FB
\\PeopleTools De...	PTD_PIVOTGRI...	39	31	Page	Save			PTPG_WIZ_MENU	PTPG_WIZ_COMP	PTPG_WIZ_DISP		
\\PeopleTools De...	PTD_PIVOTGRI...	47	32	Browser	FrameSet							
\\PeopleTools De...	PTD_PIVOTGRI...	51	38	Text	Set_Value	Name=PTPG_WORKREC1...		PTPG_WIZ_MENU	PS_PG_VIEWER	PTPG_PSPGVIEWER	PTPG_WORKREC1	PTPG_PGIDNAME
\\PeopleTools De...	PTD_PIVOTGRI...	52	39	Button	Click	Name=PTPG_WORKREC1...		PTPG_WIZ_MENU	PS_PG_VIEWER	PTPG_PSPGVIEWER	PTPG_WORKREC1	PTPG_SRCH_BTN
\\PeopleTools De...	PTD_PIVOTGRI...	53	40	Link	Click	Name=PTPG_VIEW_BTN#0		PTPG_WIZ_MENU	PS_PG_VIEWER	PTPG_PSPGVIEWER	PTPG_WORKREC1	PTPG_VIEW_BTN
\\PeopleTools De...	PTD_PIVOTGRI...	54	41	Browser	FrameSet	ptModFrame_##						
\\PeopleTools De...	PTD_PIVOTGRI...	55	43	CheckBox	Set_Value	Name=PTPGDLVWR#PTP...		PTPG_WIZ_MENU	PTPGVIEWER	PTPG_PGVIEWER	PTPGVWRGRDREC	
\\PeopleTools De...	PTD_PIVOTGRI...	59	48	Link	Click	innerText=PIVOT Grid Adm...						
\\PeopleTools De...	PTD_PIVOTGRI...	60	48	Browser	FrameSet	TacrolContent						

The grid contains the same fields as the Test Editor, with two additional columns: Folder, and Test so you can identify which test the steps are from.

Folder Lists the folder hierarchy of the test the step belongs to.

Test Lists the name of the test that contains the step.

For information about the test step fields, see [Test Step Fields](#).

Mass Update – Log Grid

The Log grid displays a log of mass update activity.

Finding and Replacing Test Step Data

Click the Find icon on the Mass Update toolbar to open the Find and Replace dialog box. Two tabs are available:

Find Select the Find tab to find test step data.

Replace Select the Replace tab to find and replace test step data.

Finding Test Step Data

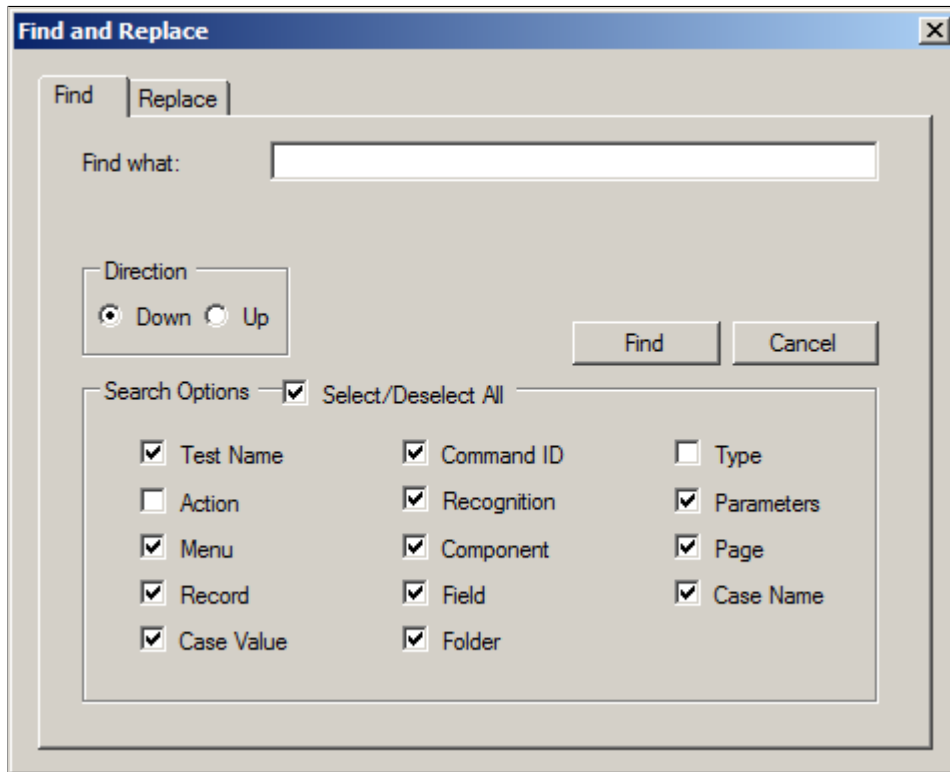
To find test step data:

1. Click the Find icon on the Mass Update toolbar.

The Find and Replace dialog box appears, with the Find tab active.

Image: Mass Update - Find and Replace Dialog Box: Find Tab

This example illustrates the fields and controls on the Find and Replace Dialog Box: Find tab.



2. In the Search Options group box, select the test step fields to include in the search.
3. In the Find what edit box, specify the search value.

For text fields, enter a search string; wildcards are supported.

For Action or Type, select a value from the list.

4. To specify the search direction, select Down or Up; the default is Down.
5. Click Find to find the next match.

Review each step and make any modifications needed.

6. Continue to click Find to review all the matching test steps.

A message appears when there are no more matches. Click OK, then click Cancel to exit the Find dialog box.

7. Click the Save icon on the Mass Update toolbar to save your changes.

Replacing Test Step Data

To replace test step data:

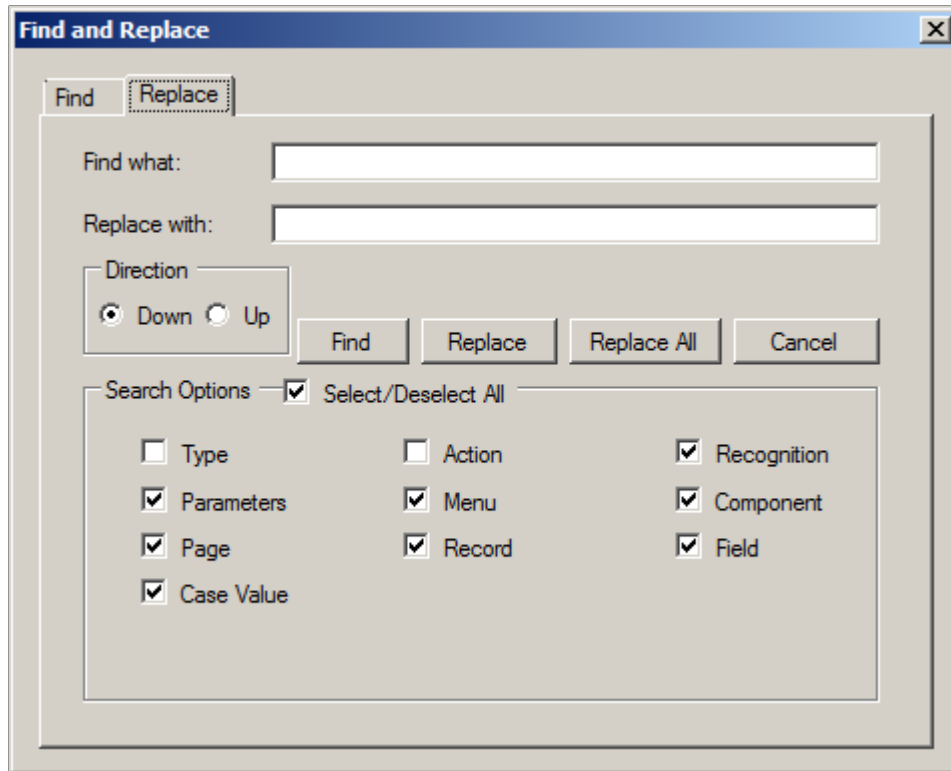
1. Click the Find icon on the Mass Update toolbar.

The Find and Replace dialog box appears, with the Find tab active.

- Click the Replace tab to access the Replace dialog.

Image: Mass Update - Find and Replace Dialog Box: Replace Tab

This example illustrates the fields and controls on the Find and Replace Dialog Box: Replace tab.



- In the Search Options group box, select the test step fields to include in the search.
- In the Find what edit box, specify the search value.

For text fields, enter a search string; wildcards are supported.

For Action or Type, select a value from the list.

- Enter the replacement value in Replace with.

For text fields, enter the text string, for Action or Type, select a value from the list.

- To specify the search direction, select Down or Up; the default is Down.
- To replace all occurrences, without individually reviewing each match, click Replace All.

A message box appears indicating the number of replacements made. Click OK, then click Cancel to close the Find and Replace dialog box.

- To review each match, click Find.

Either click Find to move to the next match without making any changes, or click Replace then Find to change the value and move to the next match.

Continue to click Find or Replace, as appropriate, to review all the remaining matching test steps.

A message appears when there are no more matches. Click OK, then click Cancel to exit the Find dialog box.

9. Click the Save icon on the Mass Update toolbar to save your changes.

Chapter 7

Identifying Change Impacts

Understanding Change Impacts

In the course of customizations and upgrades, changes are made to, among other elements, application menus, components, pages, records, and fields. Tests that were developed prior to these changes may fail when executed against the new application. For instance, if a field is deleted, moved to another page, or renamed, any step that references that field will fail. Test developers must identify and update every step in each test that is affected by the change.

One way to identify the effects on tests is to run each test against the new application and note where the test fails. This manual process is time-consuming, expensive, and prone to errors. It also fails to identify those areas in the new application that are not covered by existing tests.

Because PeopleSoft Test Framework (PTF) test assets are PeopleTools metadata and because PTF tests incorporate references to PeopleTools metadata—that is, menus, components, pages, records, and fields—PTF is able to automate the process of correlating metadata changes with existing tests.

PTF delivers two tools that help test developers to determine the effect of changes:

- Test maintenance reports

A test maintenance report correlates PeopleTools compare report data with PTF test metadata to identify certain changes to menus, components, pages, records, and fields that may impact the PTF tests.

- Test coverage reports

A test coverage report correlates PeopleTools project data with PTF test metadata to identify menus, components, pages, records, and fields that are referenced in PTF tests.

When used in conjunction with Usage Monitor, test coverage correlations can be extended to include information on additional managed objects.

A test coverage report identifies which objects included in the change project are referenced by which PTF test. Any object included in the change project that is not referenced in the PTF test metadata appears in the report identified as a coverage gap.

Defining Analysis Rules

This section discusses the page used to define analysis rules.

Defining Analysis Rules

Use the Define Analysis Rules page (PSPTTSTANLMENU) to define the priority for each of the attribute checks in a test maintenance report.

Navigation

PeopleTools, Lifecycle Tools, Test Framework, Define Analysis Rules

Image: Define Analysis Rules page

This example illustrates the fields and controls on the Define Analysis Rules page.

Analysis Category	Description	*Priority
M01	Menu does not exist	2 - Medium
M02	Component deleted from Menu	2 - Medium
M03	Component added to Menu	2 - Medium

Use the Define Analysis Rules page to define the priority for each of the attribute checks in a test maintenance report.

A test maintenance report is sorted by test name. Within each test name grouping, the report items are sorted by priority according to the values specified on the Define Analysis Rules page.

If the priority for an analysis category is set to 4 – *Ignore*, then identified impacts meeting the category criteria will not be printed in the report.

Note: Priorities are used as filters and groupings in a test maintenance report. They do not affect the actual analysis process or change what is analyzed.

The following table describes the analysis categories:

Analysis Category	Description
M01	Menu does not exist
M02	Component deleted from Menu
M03	Component added to Menu
C01	Page deleted from Component
C02	Page added to Component

<i>Analysis Category</i>	<i>Description</i>
C03	Search Record changed on Component
P01	Field deleted on Page
P02	Required Field added to Page
P03	Fieldname changed on Page
P04	Field Type changed on Page
P05	Field Label changed on Page
P06	Non-Required Field added to Page
P07	Recname changed on Page
P08	Recname & Fieldname changed on Page
R01	RecordField now required on Page
R02	RecordField no longer required on Page
R03	RecordField is now a Search Key
R04	RecordField no longer a Search Key
R05	RecordField is now a List Box Item
R06	RecordField no longer a List Box Item
F01	Field Type changed
F02	Field Length changed
F03	Field Format changed
F04	Field Decimal Positions changed
X01	Translate Value does not exist
X02	Translate Value added

Creating Test Maintenance Reports

This section discusses how to create test maintenance reports using the Create Test Maintenance Report wizard.

Access the Create Test Maintenance Report wizard (PeopleTools, Lifecycle Tools, Test Framework, Create Test Maintenance Report).

The wizard consists of four steps:

- Step 1: Manual Tasks
- Step 2: Analyze Compare Data
- Step 3: Select an Analyzed Project
- Step 4: Generate Report

Step 1 of 4: Manual Tasks

For a given change project the tasks for Step 1 only need to be executed once.

The first step in creating a test maintenance report comprises five manual tasks that you will complete in Application Designer to create a compare report that PTF will use as a basis for the maintenance report.

Perform these tasks to create a compare report from a project file.

Access the Create Test Maintenance Report wizard (PeopleTools, Lifecycle Tools, Test Framework, Create Test Maintenance Report).

Image: Create Test Maintenance Report Wizard: Step 1 of 4

This example illustrates the fields and controls on the Create Test Maintenance Report Wizard: Step 1 of 4.

As you progress through the Test Maintenance wizard the wizard tracks which tasks you have completed and which page you are on. When you exit the wizard and then return to the wizard again, the wizard sends you to the last visited page.

This tracking is done according to user ID. This means that if two users share the same user ID, the second user might not enter the first page when accessing the wizard. The wizard will take the second user to where the previous user left the wizard.

If two users with different user IDs work on the same project, the wizard does not track the state for the second user. For instance, if the first user completed the tasks for Step 1, the second user needs to check the five tasks on Step 1 to bypass that step.

Task 1: Import only the project definition for the change project.

Suppose you are about to apply a change project named SA_PROJ_UPGD. Before the change project is applied, import the change project, SA_PROJ_UPGD, as a project definition only.

1. In Application Designer, select Tools, Copy Project, From File.
2. In the selection box, highlight the change project, SA_PROJ_UPGD.
3. Click Select.

The Copy From File dialog box appears.

4. Click the Deselect All button to deselect all of the definition types so that only the empty project structure is imported.

At this point, you are importing only the names of the definitions into the project. None of the actual definitions in the upgrade project are copied.

If a definition exists in the original database with the same name as a definition in the upgrade project, the upgrade project in the database will (correctly) contain the original, unmodified definition of the object.

5. Click Copy.

Task 2: Rename the change project imported in Task 1 to create the snapshot project.

Create a copy of the change project, SA_PROJ_UPGD.

Select File, Save Project As and name the new project SA_PROJ_SNAP.

Creating the Snapshot Project is required to avoid naming conflicts later in the process.

Task 3: Create a snapshot of the original metadata definitions by exporting the snapshot project created in Task 2 to file.

Export the snapshot project, SA_PROJ_SNAP, to file:

1. Select Tools, Copy Project, To File.
2. If this project is large, as upgrades often are, you can save time during the compare process by deselecting all definitions in the Copy Options window, except for the five definitions that are referenced by PTF tests:
 - Menus
 - Components
 - Pages
 - Records
 - Fields

3. Accept the defaults and click OK.

Task 4: Import the change project (definition and objects).

Import the original change project, SA_PROJ_UPGD, from file:

1. Select Tools, Copy Project, From File.
2. Include all the definition types.

At this point, the database contains the new metadata.

Task 5: Compare the snapshot project (from file) to the database, saving the compare output to tables.

Compare the current (target) database with the pre-change (source) project, SA_PROJ_SNAP:

1. Select Tools, Compare and Report, From File.
2. Highlight the snapshot project, SA_PROJ_SNAP, and click the Select button.

The Replace existing SA_PROJ_SNAP? dialog box appears.

3. Click Yes.

The Compare and Report From File dialog box appears.

4. Click Options to access the Upgrade Options dialog box.
5. Select the Report Options tab.
6. Select the Generate Output to Tables check box.
7. Select the Report Filter tab.
8. Click Select All.
9. Click OK.
10. Click Compare.

The wizard will use data from this compare report to create the test maintenance report.

Step 2 of 4: Analyze Compare Data

This example shows Step 2 of 4:

Image: Create Test Maintenance Report Wizard: Step 2 of 4

This example illustrates the fields and controls on the Create Test Maintenance Report Wizard: Step 2 of 4.

Create Test Maintenance Report Step 2 of 4

Create a Test Maintenance Report based on compare data and test metadata.

1 2 3 4

Restart
< Previous
Next >

Analyze Compare Data

Analyze existing compare data and generate data on impact to PeopleSoft Test Framework metadata.

Select a Compare Report to Analyze Find | View All | First 1-2 of 2 Last

Project Name	Compare Date/Time	Compare Type	Analyze	Last Analysis
INSTRUCTOR	05/08/2012 9:23:17AM	<input type="text" value=""/>	Analyze	
PO_PROJ_SNAP	05/07/2012 5:52:15PM	From File	Analyze	

For a given change project the tasks for Step 2 only need to be executed once. This analysis is based only on the compare data, not on the test data, so changes to tests do not affect the result. The relationship between this analysis and test data is calculated in Step 3: Generate Impact Report. You may generate multiple Impact Reports based on a single analysis as tests change.

In this step, PTF analyzes data from the compare you performed in Wizard Step 1, Task 5.

1. For the project compare report you created in Wizard Step 1, Task 5, specify whether the Compare Type is *From File* or *To Database*.
2. Click Analyze.

This will launch the Application Engine program PSPTANALYSIS.

3. When the analysis is complete a pop-up appears. Click Next to continue.

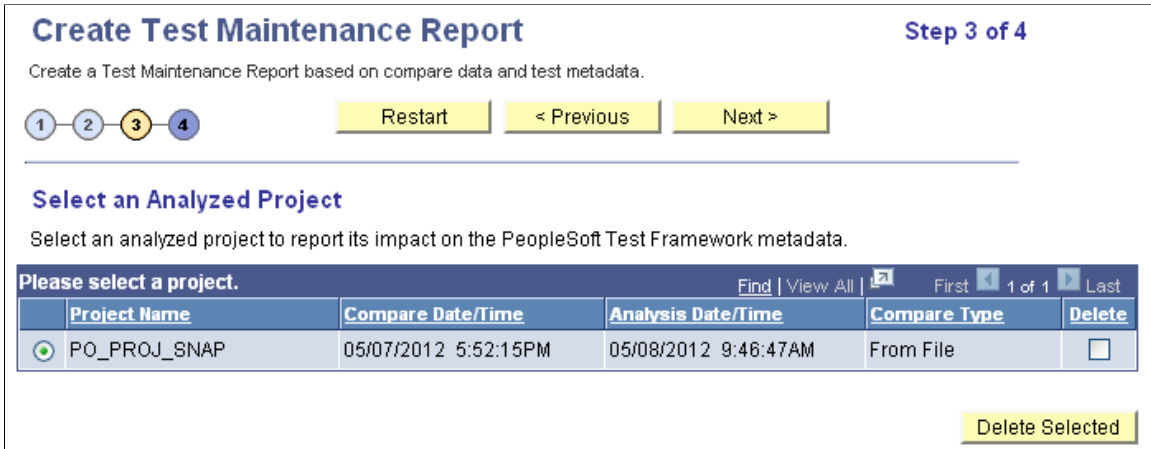
Note: A project name can appear more than once in the list because the Compare Output to Table feature stores data by both project name and compare date/time. The most recent compare will appear first in the list.

Step 3 of 4: Select an Analyzed Project

This example shows Step 3 of 4:

Image: Create Test Maintenance Report Wizard: Step 3 of 4

This example illustrates the fields and controls on the Create Test Maintenance Report Wizard: Step 3 of 4.



Use this page to select the analysis data PTF will use to generate the report.

Sets of compare data are listed by project name, the date and time the compare was generated, and the date and time the analysis was run.

Select the Delete check box and click Delete Selected to delete analyses.

Step 4 of 4: Generate Report

This example shows Step 4 of 4:

Image: Create Test Maintenance Report Wizard: Step 4 of 4

This example illustrates the fields and controls on the Create Test Maintenance Report Wizard: Step 4 of 4.

Favorites | Main Menu > PeopleTools > Lifecycle Tools > Test Framework > Create Test Maintenance Report

Create Test Maintenance Report Step 4 of 4

Create a Test Maintenance Report based on compare data and test metadata.

1 2 3 4
 Restart
< Previous
Next >

Generate Report

Generate a report of the impact to existing tests for the selected compare data.

Folder | \\PeopleTools Development \\Core

Search For Tests By | Name |

Select Test(s) for Test Maintenance Report		
Test Parent Folder	Test Name	Description
<input type="checkbox"/> \\PeopleTools Development \\Core\\TA	PTD_PIA_TA_HTMLSTR_01	PTD_PIA_TA_HTMLSTR_01

Report Format

PIA

 BI Publisher

For a given change project you will use this page to generate multiple reports as changes are made to the PTF test metadata. As you make changes to tests based on the results of this report, rerun the report to verify that the issues were corrected. You can run this report repeatedly as you make changes to tests without having to redo steps 1 through 3 of the Create Test Maintenance Report wizard.

To generate a report:

1. Select one or more tests in the Select Test(s) for Test Maintenance Report grid to use for the report.

Use the Folder and Search for Tests By fields along with the Search button to filter the list of tests that are available to select for the report:

- Select a value in the Folder list to specify the PTF Explorer Tree level to include. To include test from all folders, leave this field blank.
- To include only tests that match a test name or description, select *Name* or *Desc* from the Search for Tests By list, then enter the search string in the adjacent edit box.

Click Search to populate the Select Test(s) for Test Maintenance Report grid with the tests that match the criteria specified.

2. Select the report format.
3. Click Generate Report to create the report.

The following fields are on the Create Test Maintenance Report Wizard: Step 4 of 4 page:

Folder	To limit the report to tests from a specific folder, select the level of the PTF Explorer Tree to include. To include tests from all folders, leave this field blank.
Search for Tests By	Select <i>Name</i> to search for reports by test name, or <i>Desc</i> to search for reports by their description. Enter the search string in the adjacent edit box. To retrieve all reports, leave the edit box blank.
Search	Click to populate the Select Test(s) for Test Maintenance Report grid with the tests that match the criteria specified.
Select All	Click to select all tests that are listed in the Select Test(s) for Test Maintenance Report grid.
Clear All	Click to deselect all tests that are listed in the Select Test(s) for Test Maintenance Report grid.
Report Format	Select the report output format. Options are: <ul style="list-style-type: none"> • PIA - View the report in the application browser. • BI Publisher - Create an XML formatted text file.
Generate Report	Click to create the report.

Running a Test Report from PTF Client

As you modify a test based on the results of a maintenance report, you can validate the test by running a test report from the PTF client. The report is generated for a single test, using analyses generated in Step 2 of the Test Maintenance Wizard.

1. In PTF Explorer, right-click on a test.
2. From the context menu, select Validate Test.
3. The Validate Test – Analysis Project list dialog box appears.
4. Select an analysis from the list and click Open.

The test maintenance report opens in a new window in the PTF client.

Interpreting Test Maintenance Reports

A test maintenance report lists the tests that have been impacted by changes to menus, components, records, pages, and fields, and details the changes that impacted those tests. You can choose to output the report to PIA or to BI Publisher .

Image: Example of a Test Maintenance Report in BI Publisher Format

This example illustrates a test maintenance report in PI Publisher format.

Test Maintenance Report			
Project Name SA_PROJ_SNAP_852			
Test Name: INSTRUCTORS			
2 - Medium	F01F Field	INTERNAL_EXTERNAL	Changed
		Step ID: 31 Seq. Nbr: 31 Status: Active Language: EN	
		Pre Object Value: INTERNAL_EXTERNAL	
		Post Object Value: INTERNAL_EXTERNAL.Character	
2 - Medium	F04F Field	INTERNAL_EXTERNAL	Changed
		Step ID: 31 Seq. Nbr: 31 Status: Active Language: EN	
		Pre Object Value: INTERNAL_EXTERNAL	
		Post Object Value: INTERNAL_EXTERNAL.0	
2 - Medium	P01F Page	PSU_INSTR	Deleted
		Step ID: 32 Seq. Nbr: 32 Status: Active Language: EN	
		Pre Object Value: PSU_INSTR_TBL.MANAGER	
		Post Object Value:	
2 - Medium	P05F Page	PSU_INSTR	Changed
		Step ID: 11 Seq. Nbr: 11 Status: Active Language: EN	
		Pre Object Value: PSU_INSTR_TBL.FIRST_NAME.First Name	
		Post Object Value: PSU_INSTR_TBL.FIRST_NAME.First	
Test Name: RESERVED WORDS			
2 - Medium	C01F Component	PSU_STUDENT	Deleted
		Step ID: 6 Seq. Nbr: 5 Status: Active Language: ENG	
		Pre Object Value: PSU_STUDENT_SKED	
		Post Object Value:	
2 - Medium	C01F Component	PSU_STUDENT	Deleted
		Step ID: 7 Seq. Nbr: 6 Status: Active Language: ENG	
		Pre Object Value: PSU_STUDENT_SKED	
		Post Object Value:	
2 - Medium	C01F Component	PSU_STUDENT	Deleted
		Step ID: 8 Seq. Nbr: 7 Status: Active Language: ENG	
		Pre Object Value: PSU_STUDENT_SKED	
		Post Object Value:	
2 - Medium	C01F Component	PSU_STUDENT	Deleted
		Step ID: 13 Seq. Nbr: 8 Status: Active Language: ENG	
		Pre Object Value: PSU_STUDENT_SKED	
		Post Object Value:	

Image: Example of a Test Maintenance Report in PIA Format

This example illustrates a test maintenance report in PIA format.

Project Name: SA_PROJ_SNAP													
Compare Date/Time: 07/16/2010 5:14:06PM Analysis Date/Time: 07/16/2010 5:15:09PM													
Tests with Impacted Objects													
Test Name	Priority	Category	Object Type	Object Name	Market	Command ID	Seq Nbr	Status	Object Action	Pre Object Value	Post Object Value	Description	
INSTRUCTORS	2 - Medium	P001	Pages	PSU_INSTR			32	32	Active	Deleted	PSU_INSTR_TBL.MANAGER		Field deleted on Page
INSTRUCTORS	2 - Medium	P003	Pages	PSU_INSTR			0	0	Active	Added		PSU_INSTR_TBL.CURR_DEV_FLAG	Field added to Page (Compare Type File)
INSTRUCTORS	2 - Medium	P003	Pages	PSU_INSTR			0	0	Active	Added		PSU_INSTR_TBL.CURR_DEV_FLAG	Field added to Page (Compare Type File)
INSTRUCTORS	2 - Medium	P008	Pages	PSU_INSTR			11	11	Active	Changed	PSU_INSTR_TBL.FIRST_NAME First Name	PSU_INSTR_TBL.FIRST_NAME First	Field Label changed on Page
INSTRUCTORS	2 - Medium	P008	Pages	PSU_INSTR			32	32	Active	Changed	PSU_INSTR_TBL.MANAGER Manager	PSU_INSTR_TBL.MANAGER	Field Label changed on Page
STUDENTS	2 - Medium	P003	Pages	PSU_STUDENT_PERS			0	0	Active	Added		PSU_STUDENT_TBL.PHONE	Field added to Page (Compare Type File)
TRN_STUDENT_04	2 - Medium	P003	Pages	PSU_STUDENT_PERS			0	0	Active	Added		PSU_STUDENT_TBL.PHONE	Field added to Page (Compare Type File)

From a PIA report, you can click the Download icon to output the report to a spreadsheet. The following example shows a report in spreadsheet format:

Image: Example of a Test Maintenance Report in Spreadsheet Format

This example illustrates a test maintenance report that has been downloaded in spreadsheet format.

1	Test Name	Priority	Category	Object Type	Object Name	Market	Step ID	Seq Nbr	Status	Object Action	Pre Object Value	Post Object Value	Description
2	INSTRUCTORS	2 - Medium	F01F	Field	INTERNAL_EXTERNAL		31	31	Active	Changed	INTERNAL_EXTERNAL	haracter	Type File)
3	INSTRUCTORS	2 - Medium	F04F	Field	INTERNAL_EXTERNAL		31	31	Active	Changed	INTERNAL_EXTERNAL	INTERNAL_EXTERNAL	Field Decimal Positions changed (Compare Type File)
4	INSTRUCTORS	2 - Medium	P01F	Page	PSU_INSTR		32	32	Active	Deleted	PSU_INSTR_TBL.MANAGER		Type File)
5	INSTRUCTORS	2 - Medium	P05F	Page	PSU_INSTR		11	11	Active	Changed	PSU_INSTR_TBL.FIRST_NAME First Name	PSU_INSTR_TBL.FIRST_NAME First	Field Label changed on Page (Compare Type File)
6	RESERVED_WORDS	2 - Medium	C01F	Component	PSU_STUDENT	Global	6	5	Active	Deleted	PSU_STUDENT_SKED		Page deleted from Component (Compare Type File)
7	RESERVED_WORDS	2 - Medium	C01F	Component	PSU_STUDENT	Global	7	6	Active	Deleted	PSU_STUDENT_SKED		Page deleted from Component (Compare Type File)
8	RESERVED_WORDS	2 - Medium	C01F	Component	PSU_STUDENT	Global	8	7	Active	Deleted	PSU_STUDENT_SKED		Page deleted from Component (Compare Type File)
9	RESERVED_WORDS	2 - Medium	C01F	Component	PSU_STUDENT	Global	13	8	Active	Deleted	PSU_STUDENT_SKED		Page deleted from Component (Compare Type File)
10	RESERVED_WORDS	2 - Medium	C01F	Component	PSU_STUDENT	Global	14	9	Active	Deleted	PSU_STUDENT_SKED		Page deleted from Component (Compare Type File)

Image: Example of a Test Maintenance Report in PTF Client

This example illustrates a test maintenance report in the PTF Client.

AnalysisCategory	Priority	Step ID	LanguageCode	SeqNbr	CommandStatus	ObjectType	ObjectDescription	ObjectName
P01F	2	32	EN	32	A	5	Page	PSU_INSTR
P05F	2	11	EN	11	A	5	Page	PSU_INSTR
F01F	2	31	EN	31	A	2	Field	INTERNAL_EX
F04F	2	31	EN	31	A	2	Field	INTERNAL_EX

The following columns are on a Test Maintenance report (test data may be positioned differently depending on the report format):

Column Name	Description
Test Name	The test that is impacted by a change.
Priority	A test maintenance report is sorted by test name. Within each test name grouping, the report items are sorted by priority, according to the values specified on the Define Analysis Rules page. See Defining Analysis Rules .
Category	Which category the change belongs to, as detailed on the Define Analysis Rules page. See Defining Analysis Rules .

Column Name	Description
Object Type	The type of definition that was changed: <ul style="list-style-type: none"> • Menu • Component • Page • Record • Field
Object Name	The name of the object that was changed.
Market	For components, the market; for instance, GBL.
Step ID	A unique and unchanging identifier for a step in a test. Each step has a Step ID and a Sequence Number, but the Sequence Number, unlike the Step ID, changes when steps are added or deleted from a test.
Seq. Nbr	The sequence number for the test step reflects the relative run order position of the step within the test.
Status	Indicates whether, within the test, the step is active or inactive.
Object Action	Indicates whether the object was: <ul style="list-style-type: none"> • Changed • Added • Deleted
Pre Object Value	The value before the object was changed.
Post Object Value	The value after the object was changed.
Description	A brief description of the change.

Understanding Test Coverage Reports

In addition to being a record and playback tool, PTF is one element of the broader PeopleTools Lifecycle Management framework, which provides greater visibility into how organizations use their PeopleSoft environments, and how changes to PeopleSoft managed objects might impact their business processes.

Documenting business process tests in PTF enables you to correlate business processes to the underlying managed objects. Using PTF in conjunction with Usage Monitor provides you with even greater levels of information about the objects used in your system.

Test Coverage reports leverage PTF integration with PeopleTools to provide this information to you in a usable form.

Creating Test Coverage Reports

By default, a test coverage report correlates PeopleTools project data with PTF test metadata to identify components, menus, pages, records and fields that are referenced in PTF Tests.

When used in conjunction with Usage Monitor, test coverage correlations can be extended to include information on additional PeopleTools managed objects, such as PeopleCode.

A test coverage report identifies which objects included in the change project are referenced by which PTF test. Any object included in the change project that is not referenced in the PTF test metadata will appear in this report, identified as a coverage gap.

All the projects in the database are listed, sorted with most recent first. Click the column headings to change the sort order.

PTF generates a coverage report based on a change project. Select the project you want PTF to use to generate the report.

The wizard consists of two steps:

- Step 1: Select a Project.
- Step 2: Generate Report.

Step 1 of 2: Select a Project

Use the Create Test Coverage Report page (PSPTTSTCOVERAGESEL) to select the projects to include in the test coverage report.

Navigation

PeopleTools, Lifecycle Tools, Test Framework, Create Test Coverage Report

Image: Create Test Coverage Report Wizard: Step 1 of 2

This example illustrates the fields and controls on the Create Test Coverage Report Wizard: Step 1 of 2. You can find definitions for the fields and controls later on this page.

Create Test Coverage Report

Create a Test Coverage Report based on a project definition and test metadata.

1
 2

Please select a Project.

Select a Project for the Test Coverage Report			Find View 100 First 1-15 of 202 Last
<input type="radio"/>	Project Name	Project Description	Last Update Date/Time
<input type="radio"/>	QE854_90311	Increm Proj for QE8.54_90311	05/06/2014 6:44:27PM
<input type="radio"/>	PTFTLSPROJ		05/06/2014 6:43:01PM
<input type="radio"/>	PPLDELETE	Deleted PeopleTools Objects	05/06/2014 6:42:34PM
<input type="radio"/>	PPLTLS84CUR	Composite PeopleTools Maintena	05/06/2014 6:40:38PM
<input type="radio"/>	PPLTOOLS	Composite PeopleTools Project	05/06/2014 6:38:26PM
<input type="radio"/>	PT854_90311	Increm Proj for PT8.54_90311	05/06/2014 3:24:25PM
<input type="radio"/>	QEDMO854	Comp Proj for QE8.54	05/06/2014 8:53:14AM
<input type="radio"/>	QEDMOALL	Composite QE Project	05/06/2014 8:51:08AM
<input type="radio"/>	QE_ASIVADAS_854_90311		05/05/2014 5:34:54PM
<input type="radio"/>	PT854_902R1	Increm Proj for PT8.54_902R1	04/29/2014 11:08:52AM
<input type="radio"/>	QE854_90211	Increm Proj for QE8.54_90211	04/22/2014 5:01:28PM
<input type="radio"/>	PTPCRECOMPILE		04/22/2014 3:57:08PM
<input type="radio"/>	PT854_90211	Increm Proj for PT8.54_90211	04/22/2014 10:23:00AM
<input type="radio"/>	QE854_901R1	Increm Proj for QE8.54_901R1	04/15/2014 4:31:16PM
<input type="radio"/>	PT854_901R1	Increm Proj for PT8.54_901R1	04/14/2014 4:10:34PM

Select a change project to be correlated with PTF metadata and click Next.

Step 2 of 2: Generate Report

This example shows step 2 of 2:

Image: Create Test Coverage Report Wizard: Step 2 of 2

This example illustrates the fields and controls on the Create Test Coverage Report Wizard: Step 2 of 2.

Create Test Coverage Report

Create a Test Coverage Report based on a project definition and test metadata.

1
2

Previous
Next

Project Name PT854_901R1

Last Update Date/Time 04/14/2014 4:10:34PM

Folder

Search For Tests By Name

Search

Generate report for either all Tests or select a maximum of 10 Tests

Select Test(s) for the Test Coverage Report		
Test Parent Folder	Test Name	Description
<input type="checkbox"/>	\\PeopleTools Development\\Analytic Tools\\Analytic Engine (ACE)	PTD_ACE_ACCESS_CUBE_FILTER
<input type="checkbox"/>	\\PeopleTools Development\\Analytic Tools\\Analytic Engine (ACE)	PTD_ACE_GEN_TESTCASES
<input type="checkbox"/>	\\PeopleTools Development\\Analytic Tools\\Analytic Engine (ACE)	PTD_ACE_MDL_VWR
<input type="checkbox"/>	\\PeopleTools Development\\Analytic Tools\\Analytic Engine (ACE)	PTD_ACE_Z_DUMMY
<input type="checkbox"/>	\\PeopleTools Development\\Porting, Database and Platform\\Automated Config Manager (ACM)	PTD_ACM_TEMPLATE_CREATION ACM Template Creation
<input type="checkbox"/>	\\PeopleTools Development\\Porting, Database and Platform\\Automated Config Manager (ACM)	PTD_ACM_TEMPLATE_RUN ACM Template Run Configuration

Select All
Clear All

Report Format

PIA
 BI Publisher

Include Usage Monitor Data

Usage Monitor Rpt

Generate Report

To generate the report:

1. Select one or more tests in the Select Test(s) for the Test Coverage Report grid to use for the report.

Use the Folder and Search for Tests By fields along with the Search button to filter the list of tests that are available to select for the report:

- Select a value in the Folder list to specify the PTF Explorer Tree level to include. To include tests from all folders, leave this field blank.
- To include only tests that match a test name or description, select *Name* or *Desc* from the Search for Tests By list, then enter the search string in the adjacent edit box.

Click Search to populate the Select Test(s) for the Test Coverage Report grid with the tests that match the criteria specified.

2. Select the report format.
3. Click Generate Report to create the report.

The following fields are on the Create Test Coverage Report Wizard: Step 2 of 2 page:

Folder	To limit the report to tests from a specific folder, select the level of the PTF Explorer Tree to include. To include tests from all folders, leave this field blank.
Search for Tests By	Select <i>Name</i> to search for reports by test name, or <i>Desc</i> to search for reports by their description. Enter the search string in the adjacent edit box. To retrieve all reports, leave the edit box blank.
Search	Click to populate the Select Test(s) for Test Coverage Report grid with the tests that match the criteria specified.
Select All	Click to select all tests that are listed in the Select Test(s) for Test Coverage Report grid.
Clear All	Click to deselect all tests that are listed in the Select Test(s) for Test Coverage Report grid.
Report Format	Select the report output format. Options are: <ul style="list-style-type: none"> • PIA - View the report in the application browser. • XML Publisher - Create an XML formatted text file.
Usage Monitor Rpt	Select to include Usage Monitor data in a test coverage report. You need to generate Usage Monitor data while creating your tests. See Using Usage Monitor Data with PTF .
Generate Report	Click to create the report.

Using Usage Monitor Data with PTF

PTF used by itself tracks the use of five step types – records, menus, fields, pages, and components. You can use PTF together with Usage Monitor to track additional PeopleTools managed objects used in the course of a test. This data can then be correlated with change projects (that is, fixes or bundles) to determine which tests need to be executed to test the change project.

For example, suppose you have 100 PTF tests in your test repository. You receive a bundle from PeopleSoft that consists of updates to ten PeopleCode objects. Based solely on the data stored by PTF, you cannot determine which PTF tests you need to run to test the bundle, because PTF does not track references to PeopleCode.

You can use Usage Monitor in conjunction with PTF to address this issue. Usage Monitor tracks references to additional PeopleTools managed objects, including PeopleCode. When you associate a test

and test case with Usage Monitor all the actions taken while Usage Monitor is active are associated with the test and test case you specified.

A test coverage report correlates project data, PTF data, and Usage Monitor data to identify, in this example, which PTF Tests in the test repository reference one or more of the PeopleCode objects delivered in the bundle.

Configuring Usage Monitor

Your system administrator must configure Performance Monitor and Usage Monitor before you use Usage Monitor with PTF.

See *PeopleTools: Performance Monitor*

Generating Usage Monitor Data

Follow these steps to generate Usage Monitor data along with PTF test data:

1. Create a new test in PTF.
2. Launch the Recorder.
3. Hook the browser.
4. Start recording.
5. In the PeopleSoft application browser, select Usage Monitoring from the main menu.

Note: Your PTF environment and the application you are testing must be on the same database.

6. The Test Data page appears.
7. Enter Test Name and Test Case.
8. Click Start test.
9. Record the test steps.
10. When you are finished with the test steps, return to the Usage Monitoring page and select Stop Test.
11. In the PTF Recorder click the Stop Recording icon.

Note: The manual steps of starting and stopping the Usage Monitor are included here during the recording process to capture them in the PTF test. PTF does not have to be in record mode to generate Usage Monitor data.

Administering Usage Monitor for PTF

Before Usage Monitor data can be used in a coverage report it must be aggregated.

To aggregate the Usage Monitor Data:

1. Navigate to PeopleTools, Process Scheduler, System Process Requests.

2. Enter a run control ID or create a new one.
3. Click Run.
4. Select Usage Monitor Aggregator (PTUMAGR) from the list.
5. Click OK to Run the process.

Note: We recommend that you schedule the Usage Monitor Aggregator Process to execute hourly or daily to keep the data in the aggregated data table current. Once data has been aggregated the raw data can be purged to reduce disk space usage.

To verify that Usage Monitor is collecting data, run this query in your query tool:

```
SELECT * FROM PSPMTRANS35_VW;
```

The PSPMTRANS35_VW represents the correct logical view of the raw Usage Monitor data.

Note: For any Usage Monitor data to appear in the view, the Usage Monitor buffers need to have been flushed at least once. By default the buffer size is set to 500 unique object references. While you are configuring and testing your Usage Monitor setup you might find it useful to reduce the size of the buffer (PeopleTools, Performance Monitor, Administration, System Defaults).

To verify that Usage Monitor Data has been aggregated correctly, run this query in your query tool:

```
SELECT * FROM PSPTUMPMTAGR;
```

Interpreting Test Coverage Reports

Use a Test Coverage report to determine which objects in a change package have tests and which do not. Objects without a test represent a potential gap in the test coverage.

This is an example of a test coverage report in PIA format:

Image: Example of a Test Coverage Report

This example illustrates a test coverage report in PIA format.

Project Name: SA_PROJ_SNAP_852					
Impacted Objects and Objects with No Coverage					
Test Name	Object Type	Object Name	Object Detail	Test Metadata	Usage Monitor Data
** No Test Coverage **	Page	PSU_COURSE		No Data	No Data
** No Test Coverage **	Page	PSU_COURSE_MATL		No Data	No Data
** No Test Coverage **	Page	PSU_INSTR_PHOTO		No Data	No Data
** No Test Coverage **	Panel Group PeopleCode	PSU_CUST	GBL.SavePostChange	No Data	No Data
** No Test Coverage **	Panel Group Record Field PeopleCode	PSU_INSTR	GBL.DERIVED_ED_SVCS.DELETE_BTN	FieldChange	No Data
** No Test Coverage **	Panel Group Record Field PeopleCode	PSU_INSTR	GBL.DERIVED_ED_SVCS.INSERT_BTN	FieldChange	No Data
** No Test Coverage **	Translate	INTERNAL_EXTERNAL	0.2010-07-16	No Data	No Data
INSTRUCTORS	Component	PSU_INSTR		Active	No Data
INSTRUCTORS	Field	INTERNAL_EXTERNAL		Active	No Data
INSTRUCTORS	Page	PSU_INSTR		Active	No Data
INSTRUCTORS	Record	PSU_INSTR_TBL		Active	No Data
RESERVED_WORDS	Component	PSU_STUDENT	GBL	Active	No Data

The following columns are on a Test Coverage report (test data may be positioned differently depending on the report format):

Column Name	Description
Test Name	The test that is impacted by a change.
Object Type	The type of definition that was in the change project.
Object Name	The primary key of the object.
Object Detail	The additional keys (as applicable) required to uniquely identify the object.
Test Metadata	<p>If the object is referenced in a PTF test, this value will be <i>Active</i> or <i>Inactive</i>, reflecting whether within the test, the referenced step is active or inactive.</p> <p>If the object is not referenced in a PTF test, this value will be <i>No Data</i>. This scenario can occur when the Include Usage Monitor Data checkbox is selected and the value in the Usage Monitor Data column is Yes.</p>
Usage Monitor Data	When the Include Usage Monitor Data checkbox is selected this value will be <i>Yes</i> or <i>No</i> . When the Include Usage Monitor Data checkbox is deselected the value will be set to <i>No Data</i> .

Running Test Details Reports

A test details report is an HTML file with details for a PTF test and its associated test cases, including comments in rich text format with images.

Use the Create Test Details Report page (PSPTTSTDTLRUNCNTL) to run a test details report.

Navigation

PeopleTools, Lifecycle Tools, Test Framework, Create Test Details Report

Image: Create Test Details Report Page

This example illustrates the fields and controls on the Create Test Details Report page.

Create Test Details Report

Create a Test Details Report based on test metadata.

[Report Manager](#) [Process Monitor](#) [Run Report](#)

Folder

Search For Tests By

[Search](#)

Select from 1 to 10 Tests			
	Test Parent Folder	Test Name	Description
<input type="checkbox"/>	\PeopleTools Development\Analytic Tools\Charting (CHART)	PTD_CHART_ORG_BASICS	Basic tests for Org Chart
<input type="checkbox"/>	\PeopleTools Development\Analytic Tools\Charting (CHART)	PTD_CHART_RB_BASICS	Basic test for RB chart
<input type="checkbox"/>	\PeopleTools Development\Analytic Tools\Charting (CHART)	PTD_CHART_Z_DUMMY	

To run the Test Detail report:

1. Select a minimum of one to a maximum of ten tests in the Select from 1 to 10 Tests grid to use for the report.

Use the Folder and Search for Tests By fields along with the Search button to filter the list of tests that are available to select for the report:

- Select a value in the Folder list to specify the PTF Explorer Tree level to include. To include tests from all folders, leave this field blank.
- To include only tests that match a test name or description, select *Name* or *Desc* from the Search for Tests By list, then enter the search string in the adjacent edit box.

Click Search to populate the Select from 1 to 10 Tests grid with the tests that match the criteria specified.

2. Click Run Report to create the report.

Note: The run control is automatically generated by the system and the report is scheduled. You can use the Process Monitor link to check the report status and the Report Manager link to view the report.

Image: Example Test Details Report

This example illustrates a test details report.

PeopleSoft Test Framework					
Test Details Report					
Test Name:	PB_USER_PROFILE_SHORT	Type:	Test	Message Recognition:	Y
Example for Test Details					
Test Comments:					
This test example has the first few steps removed to show details for a typical step. All test cases have been removed except CASE_01 to simplify the report.					
Test Case Information					
Test Case Name:		CASE_01			
Comments:					
Test Steps					
1	Active	Page.Prompt	MAINTAIN_SECURITY.USER_SAVEAS.GBL	Command Id: 10	Language: ENG
		<i>Object Properties</i>	<i>Menu:</i> MAINTAIN_SECURITY	<i>Component:</i> USER_SAVEAS.GBL	
			<i>Page:</i> (SEARCH)	<i>PageField:</i>	
			<i>Record:</i>	<i>Field:</i>	
Comments:					
The step uses the Page Prompt construct instead of explicit navigation.					
Test Case Details:					
CASE_01			Value: update		

Creating a Test Compare Report

Use the Create Test Compare Report page (PSPTTSTCOMPARESEL) to run a test compare report based on a project file.

See [Creating Test Maintenance Reports](#)

Navigation

PeopleTools, Lifecycle Tools, Test Framework, Create Test Compare Report

Image: Create Test Compare Report Page

This example illustrates the fields and controls on the Create Test Compare Report Page. You can find definitions for the fields and controls later on this page.

Create Test Compare Report
Create a Test Compare Report based on output data from previous project compares.

1 2 Previous Next

Project Name EP92PROD_C358_CHANGED
Run Date/Time 05/30/14 9:38:52AM

*Search For Tests By Name Search

Generate report for either all Tests or select a maximum of 10 Tests

Select Test(s) to Create Compare Report Find | View All | [?](#) First 1 of 1 Last

	Test Name	Description
<input checked="" type="checkbox"/>	TC_12_06_COPY_TEMPLATE_03	My Profile page

Select All Clear All

Report Format

PIA Excel

Generate Report

To generate the report:

1. Select one or more tests in the Select Test(s) to Create Compare Report grid to use for the report.

To list only tests that match a test name or description, select *Name* or *Desc* from the Search for Tests By list, enter the search string in the adjacent edit box, then click Search to populate the Select Test(s) to Create Compare Report grid with the tests that match the criteria specified.

2. Select the report format.
3. Click Generate Report to create the report.

The following fields are on the Create Test Compare Report page:

Search for Tests By

Select *Name* to search for reports by test name, or *Desc* to search for reports by their description.

Enter the search string in the adjacent edit box. To retrieve all reports, leave the edit box blank.

Search

Click to populate the Select Test(s) to Create Compare Report grid with the tests that match the criteria specified.

Select All Click to select all tests that are listed in the Select Test(s) to Create Compare Report grid.

Clear All Click to deselect all tests that are listed in the Select Test(s) to Create Compare Report grid.

Report Format Select the report output format. Options are:

- PIA - View the report in the application browser.
- Excel - Create a Microsoft Excel file.

Generate Report Click to create the report.

Image: Example of a test compare report in PIA format

This example illustrates a test compare report in PIA format.

Project Name EP92PROD_C358_CHANGED		Report Type PIA			
Run Date/Time 05/30/2014 9:38:52AM					
Test Compare Report					
		Find View All [?] First 1-11 of 11 Last			
Test Name	Test Case Name	Command ID	Attribute	Source	Target
TC_12_06_COPY_TEMPLATE_03			Version	8.53.12A	8.53.10A-B1
TC_12_06_COPY_TEMPLATE_03		17	Rec&Field		GROUP_NAME
TC_12_06_COPY_TEMPLATE_03		65	Object ID	TUS\$53\$&25591_Row1	TUS\$67\$&25591_Row1
TC_12_06_COPY_TEMPLATE_03		67	Object ID	ES\$56\$&25591_Row1	ES\$51\$&25591_Row1
TC_12_06_COPY_TEMPLATE_03		68	Object ID	SURE\$58\$&25591_Row1	SURE\$52\$&25591_Row1
TC_12_06_COPY_TEMPLATE_03		69	Object ID	\$60\$&25591_Row1	\$53\$&25591_Row1
TC_12_06_COPY_TEMPLATE_03		74	Object ID	ID=VENDOR_NAME1&25591_Row2	ID=VENDOR_NAME1&50\$&25591_Row2
TC_12_06_COPY_TEMPLATE_03		75	Object ID	TUS\$53\$&25591_Row2	TUS\$67\$&25591_Row2
TC_12_06_COPY_TEMPLATE_03		77	Object ID	ES\$56\$&25591_Row2	ES\$51\$&25591_Row2
TC_12_06_COPY_TEMPLATE_03		78	Object ID	SURE\$58\$&25591_Row2	SURE\$52\$&25591_Row2
TC_12_06_COPY_TEMPLATE_03		79	Object ID	\$60\$&25591_Row2	\$53\$&25591_Row2
<input type="button" value="Return"/>					

Creating Test Matrix Reports

The Matrix Details report enables you to view the top level parent and child test(s) for the test(s) that you specify.

To create a Matrix Details report, access the Matrix Details Report page (PSPTTSTRELGEN).

Navigation

PeopleTools, Lifecycle Tools, Test Framework, Create Test Matrix Report

Image: Matrix Details Report Page

This example illustrates the fields and controls on the Matrix Details Report page.

Test Matrix Report

Search Criteria

Folder

Search For Tests By

Project

Generate report for either all Tests or select a maximum of 10 Tests

Select Test(s) to Matrix Report			
	Test Parent Folder	Test Name	Description
<input type="checkbox"/>	PTQA\TESTSCRIPTS\SES_SearchFrameWork	PTD_SF_DELETE_QUERY	
<input type="checkbox"/>	PTQA\TESTSCRIPTS\SES_SearchFrameWork\DELETE_QUERY	PTD_SF_DEL_QRY_CAT	
<input type="checkbox"/>	PTQA\TESTSCRIPTS\SES_SearchFrameWork\DELETE_QUERY	PTD_SF_DEL_QRY_DEF	
<input type="checkbox"/>	PTQA\TESTSCRIPTS\SES_SearchFrameWork\DELETE_QUERY	PTD_SF_DEL_QRY_DEPLOY	
<input type="checkbox"/>	PTQA\TESTSCRIPTS\SES_SearchFrameWork\DELETE_QUERY	PTD_SF_DEL_QRY_SRCH_INDEX	
<input type="checkbox"/>	PTQA\TESTSCRIPTS\SES_SearchFrameWork\DELETE_QUERY	PTD_SF_DEL_QRY_SRCH_INDY_FULL	
<input type="checkbox"/>	PTQA\TESTSCRIPTS\SES_SearchFrameWork\DELETE_QUERY	PTD_SF_DEL_QRY_TEST_PAGE1	
<input type="checkbox"/>	PTQA\TESTSCRIPTS\SES_SearchFrameWork\DELETE_QUERY	PTD_SF_DEL_QRY_TEST_PAGE_SEC	
<input type="checkbox"/>	PTQA\TESTSCRIPTS\SES_SearchFrameWork\DELETE_QUERY	PTD_SF_DEL_QRY_TST_PAGE	

To generate the report:

1. Select one or more tests in the Select Test(s) to Matrix Report grid to use for the report.

Use the Folder and Search for Tests By fields along with the Search button to filter the list of tests that are available to select for the report:

- Select a value in the Folder list to specify the PTF Explorer Tree level to include. To include tests from all folders, leave this field blank.
- To include only tests that match a test name or description, select *Name* or *Desc* from the Search for Tests By list, then enter the search string in the adjacent edit box.
- To include only tests within a specified Application Designer project, select a Project.

Click Search to populate the Select Test(s) to Matrix Report grid with the tests that match the criteria specified.

2. Click Generate Report to create the report.

The following fields are on the Matrix Details Report page:

- Folder** To limit the report to tests from a specific folder, select the level of the PTF Explorer Tree to include. To include tests from all folders, leave this field blank.

- Search for Tests By** Select *Name* to search for reports by test name, or *Desc* to search for reports by their description.

Enter the Search string in the adjacent edit box. To retrieve all reports, leave the edit box blank.

- Project** Select a project from the list to limit the report to tests from a specific Application Designer project.

- Search** Click to populate the Select Test(s) to Matrix Report grid with the tests that match the criteria specified.

- Select All** Click to select all tests that are listed in the Select Test(s) to Matrix Report grid.

- Clear All** Click to deselect all tests that are listed in the Select Test(s) to Matrix Report grid.

- Generate Report** Click to create the report.

Image: Matrix Details Report Example

This example illustrates the generated Matrix Details report.

Personalize Find View All [Icons]						First	1-12 of 12	Last
Test Parent Folder	Test Name	Test Type	Top Parent	Selected Test	Child Test			
1 \PTQA\TESTSCRIPTS\SES_SearchFrameWork	PTD_SF_DELETE_QUERY	Shell Test	X					
2 \PTQA\TESTSCRIPTS\SES_SearchFrameWork\DELETE_QUERY	PTD_SF_DEL_QRY_SRCH_INDEX	Test		X	X			
3 \PTQA\TESTSCRIPTS\SES_SearchFrameWork\DELETE_QUERY	PTD_SF_DEL_QRY_SRCH_INDX_FULL	Test		X	X			
4 \PTQA\TESTSCRIPTS\SES_SearchFrameWork\DELETE_QUERY	PTD_SF_DEL_QRY_DEF	Test			X			
5 \PTQA\FLAJAX_Gen	PTD_SF_AJAX_SELECT_LOOKUP_ITEM	Library			X			
6 \PTQA\FLAJAX_Gen	PTD_SF_FRAMESET	Library			X			
7 \PTQA\TESTSCRIPTS\SES_SearchFrameWork\DELETE_QUERY	PTD_SF_DEL_QRY_TST_PAGE	Test			X			
8 \PTQA\FL\SES	PTD_SF_DEL_QRY_SEARCH	Library			X			
9 \PTQA\TESTSCRIPTS\SES_SearchFrameWork\DELETE_QUERY	PTD_SF_DEL_QRY_TEST_PAGE_SEC	Test			X			
10 \PeopleTools Development\Search Framework	PTD_SF_PRCS_WT_FR_DIST_ST	Library			X			
11 \PTQA\TESTSCRIPTS\SES_SearchFrameWork\DELETE_QUERY	PTD_SF_DEL_QRY_CAT	Test			X			
12 \PTQA\TESTSCRIPTS\SES_SearchFrameWork\DELETE_QUERY	PTD_SF_DEL_QRY_DEPLOY	Test			X			

Return

Querying PTF Report Tables

You can query these tables using PS Query to produce your own maintenance and coverage reports:

- PSPTTSTMAINRPT

This PTF table contains the data that is used to create the Test Maintenance report.

- PSPTTSTCOVRGRPT

This PTF table contains the data that is used to create the Test Coverage report.

Chapter 8

Incorporating Best Practices

Incorporating PTF Best Practices

This section discusses how to incorporate PTF best practices.

Keep your Desktop Simple

Close all programs other than PTF and the PeopleSoft application browser during recording and execution of PTF tests. Other applications – especially those that alert the user with spontaneous notifications, such as email and instant messaging programs – can interfere with PTF and the PeopleSoft application browser and thus cause unexpected results during record and playback.

Adopt Naming Conventions

You should understand these PTF test asset naming limitations:

- Test names and test case names must be unique.

Tests and test case are PeopleTools managed objects, a very important type of PeopleSoft metadata. As a result, test names must be unique to a database and test case names must be unique to a test.

- Test and test case names are limited to letters, numbers, and underscore characters.

The first character of any test or test case name must be a letter.

- Test and test case names are limited to 30 characters.

You will find it easier to manage test assets if you adopt a systematic naming convention that reflects important characteristics of the tests and test cases you create. Here are a few suggested test characteristics that you can incorporate in your PTF test names:

- Functionality being tested.

Include a brief indication of the functional area or business process validated by the test.

- Priority. A short code, such as “P1,” to indicate priority can help testers more easily locate the tests that are critical or likely to be run most often.

- Execution order.

PTF Explorer sorts tests alphabetically within each folder. It is sometimes useful to view tests in order of intended execution, especially if some tests depend on other tests having been run previously in the same database. Include a numeric component early in the name of the test to make alphabetic order equal to execution order.

It may seem simpler to use folders to categorize tests by the above characteristics. However, folder names are often not visible to the tester and do not appear on some PTF reports, such as the Test Maintenance Report. So, regardless of whether you categorize tests by folder, it is helpful to build test characteristics into the names of your tests.

The following three test names are examples of a naming convention that reflects functionality, priority, and intended execution order:

- WRKFRC_P1_01_HIRE_REQD_FLDS
- WRKFRC_P1_02_PERSON_CHNG 21
- WRKFRC_P1_03_JOB_DATA_UPDAT

Record First

A critical best practice when automating with PTF is to record first, and record whenever possible, to drive as much information, including PeopleTools metadata, into the recorded test as possible.

You must assertively record every test step from the test. That is, even if the input fields are already set to the expected value, you must explicitly set or verify that value when recording. The recorder only captures actions that you take against objects during record time. If you skip an object during recording because it is already set to the desired value, the test will ignore it and fail if the default value of the same object is different during execution.

When you record a date field that includes a calendar object that enables the user to select a date, it is best to explicitly enter the date value in the edit box.

You may need to perform two actions instead of one. For instance, if the test instructions say to select a check box but the check box is already selected, then you will need to pause the recording, deselect the check box, restart the recording, and then select the check box. Alternatively, you could click the check box twice and then delete the extra step from your test.

If you need to add a new step to a test, select the spot in the test where the new step will occur and record the step at the insertion point. Recording a step is more likely to drive accurate information into your test than trying to construct the entire step by editing the fields through the Test Editor.

Related Links

[Recording Tests](#)

Document Tests

Make it part of your automation routine to take advantage of the description fields in PTF tests and test cases as often as possible. PTF test and test case names tend to be short and abstract, so longer descriptions will help you and future testers understand the functional purpose of available PTF tests and test data.

You can find description fields in:

- Test description.
- Test properties.

- Test case description.
- Test case properties.
- Step information.

Use the Log actions to make your tests self-documenting. For instance, you can add a Log.Message before a Test.Exec step to describe the test you are calling, and you can put a Log.Message in the called test to document what the test does.

You can add comments in RTF format to the test, test case, and steps to document your tests. Comments can include application snapshots.

Clean Up Tests

Immediately after recording a test, review the recorded test data in the Test Editor and correct actions taken inadvertently during recording, such as:

- Unnecessary or extra clicks on a clickable item, such as a check box.
- Clicks on the wrong objects, such as links and images.
- Incorrect text entered into text boxes.

Revise the recorded values in the Value field (for editable fields) and delete unneeded steps.

This might be a good time to evaluate whether you should incorporate reserved words and variables to replace static values that may be different when the test is executed.

Related Links

[Make Tests Dynamic](#)

Use Configuration and Execution Options

Employ configuration options and execution options to take login information out of the test whenever appropriate. Suppose you have the following manual test step:

1. Log into your database as a user with the PeopleSoft User role.

Coding user-specific information into many tests may make future updates difficult if user IDs in the test environment are changed.

The following example shows how to make a recorded test easier to maintain by setting configuration options to use Browser.Start_Login (it automatically inactivates the sequence of steps that records logging in (Steps 2 through 4) and replaces those steps with the single action, Browser.Start_Login) and using execution options for the user ID and password at the login page:

Image: Example of using Browser.Start_Login

This example illustrates inactivating the sequence of steps that record logging in (Steps 2 through 4) and replacing those steps with the single action, Browser.Start_Login, which takes the user ID and password from the execution options and enters them at the login page:

Seq	ID	Comment	Active	Scroll ID	Type	Action	Recognition	Parameters	Value
1			<input checked="" type="checkbox"/>		Browser	Start_Login			
2			<input type="checkbox"/>		Test	Set_Value	Name=userid		PTDOCL
3			<input type="checkbox"/>		Pwd	Set_Value	Name=pwd		1ENC29CDD008E830562C4010DE2C487016EC80509EEC
4			<input type="checkbox"/>		Button	Click	Name=Submit		
5			<input checked="" type="checkbox"/>		Link	Click	id=pfhnavbca_PORTAL_ROOT_OBJECT		
6			<input checked="" type="checkbox"/>		Link	Click	Name=fdra_MANAGE_ASSETSIndex=0		
7			<input checked="" type="checkbox"/>		Link	Click	id=fdra_MANAGE_ASSETS1		
8			<input checked="" type="checkbox"/>		Link	Click	id=fdra_USE3111		
9			<input checked="" type="checkbox"/>		Link	Click	mmerText=Enter Allocation Basis		
*			<input checked="" type="checkbox"/>						

Related Links

[Defining PTF Configuration Options](#)

[Configuring Execution Options in PTF Client](#)

[Start_Login](#)

Use Page Prompting

Use of PTF page prompting steps make tests more robust and repeatable by simplifying test data and replacing it with intelligence built into the step.

Page prompting functions replace explicit navigation in the test and instead directly access the component search page by URL manipulation. A test with the navigation explicitly defined can break when the navigation changes, even though the application is working fine. Page prompting avoids this problem.

Tests that use page prompting are more repeatable. Consider the following test instruction:

1. At the Define Calculation search page, add a calculation with the name KUSPTEST or, if it already exists, update it.

Using page prompting, a single step can navigate directly to the search page and either update or add a key, whichever is needed.

PTF can be configured to insert page prompt constructs automatically during recording. The default behavior is set by the PTF administrator on the Define Configuration Options page and can be overridden during recording using the recorder utility tools.

See the Page type actions Prompt and PromptOK for more details.

Related Links

[Prompt](#)

[PromptOK](#)

[Using the PTF Test Recorder](#)

[Defining PTF Configuration Options](#)

Use the Process Step Type

The Process step type with a Run action is useful for running processes through Process Scheduler.

When you select the *Use Process Run* check box before you record a process request, all actions in the run request standard page will be replaced by the step action Process.Run and any Process Monitor actions that are recorded will be ignored.

You can set the *Use Process Run* check box using PeopleSoft Internet Architecture (select PeopleTools, Lifecycle Tools, Test Framework, Define Configuration Options) for all tests, or you can select the check box from the Toolbar when you are ready to record a test (Configure Recording Settings icon).

The Process.Run step will handle the execution and return the process status. You can modify the parameters on the Process.Run step to indicate the expected results.

Suppose you have the following test scenario:

From the Request Calculation page, click the Process Monitor link.
In Process Monitor, click the Refresh button until either Success or Error appears in the Run Status column for your process instance.

The Process.Run step will capture all of the process information and you can modify the step parameters in PTF.

Related Links

[Process](#)

Make Tests Dynamic

Many tests involve values and conditions that are not known until a test runs. Advanced functionality in PTF enables you to build tests that adapt to the application when necessary.

These test scenarios show examples of how to exploit the dynamic test features of PTF:

- Variables

Requirement: A test that creates an entry in the application with a system-generated ID number and later has to verify that same ID number elsewhere in the application.

Solution: Store the system-generated ID to a variable in one step, and then reference that variable to verify the value in another step.

- Conditional logic

Requirement: A test that specifies that the user click an icon to expand a section of a page, but only if that section is not already expanded.

Solution: Place the step in an If_Then construct that evaluates whether the section is expanded.

- Scroll handling

Requirement: A test that updates an item in a grid regardless of the position of the item in the grid.

Solution: Use a Scroll action to identify the row by key rather than by position.

- Reserved words

Requirement: A test that instructs the user to enter the date the test is executed into the application.

Solution: Use the #TODAY reserved word to enter the current date.

Dynamic steps improve the reusability and maintainability of tests.

As you are recording, be sure to make a note of situations that require dynamic handling so you can take advantage of the appropriate dynamic construct that will ensure that the test will work at execution time.

Related Links

[Using Variables](#)

[Using Conditional Logic](#)

[Incorporating Scroll Handling](#)

[Using Reserved Words](#)

Reduce Duplication

Avoid creating tests and test steps that are duplicated elsewhere. When multiple tests validate similar functionality, it increases the complexity of test maintenance and the amount of manual work necessary to add or change test functionality later.

Follow these recommendations to help keep test duplication to a minimum:

- Drive similar tests into test cases.

Do this any time multiple tests have the same logic and where the only difference among the tests is the data entered or validated. A test to hire two employees or to create ten new vouchers would be a good candidate for taking advantage of test cases.

- Isolate sequences of test steps into called tests or libraries whenever the steps are identical.

A procedure that verifies or sets the same setting on an installation table would be a good candidate for putting into a library if multiple products or tests modify the same setting.

Related Links

[Calling Tests](#)

Chapter 9

Using the PTF Test Language

Understanding the PTF Test Structure

A PTF test is composed of a series of steps. This example shows the steps in a simple test.

Image: Example of a PTF test

This example illustrates the fields and controls for a simple PTF test.

Seq	ID	Comment	Active	Scroll ID	Type	Action	Recognition	Parameters	Value
1	1		<input checked="" type="checkbox"/>		Browser	Start_Login			
2	2		<input checked="" type="checkbox"/>		Browser	FrameSet			
3	18		<input checked="" type="checkbox"/>		Page	Prompt	WEB_PROFILE.WEB_PROFILE.GBL		update
4	46		<input checked="" type="checkbox"/>		Browser	FrameSet	TargetContent		
5	47		<input checked="" type="checkbox"/>		Button	Click	id=stCSearch		
6	49		<input checked="" type="checkbox"/>		Browser	FrameSet	TargetContent		
7	50		<input checked="" type="checkbox"/>		Link	Click	id=SEARCH_RESULT1		
8	51		<input checked="" type="checkbox"/>		Browser	FrameSet	TargetContent		
9	20		<input checked="" type="checkbox"/>		Check-Box	Get_Property	Name=PSWEBPROFWRK_DISABLEFLUID	prop=Value.ret=&disablefluidprop	
10	21		<input checked="" type="checkbox"/>		Check-Box	Get_Property	Name=PSWEBPROFWRK_PTOVERIDEFLUIDFORD	prop=Value.ret=&overidefluidfordesktopprop	
11	22		<input checked="" type="checkbox"/>		Check-Box	Set_Value	Name=PSWEBPROFWRK_DISABLEFLUID		N
12	23		<input checked="" type="checkbox"/>		Check-Box	Set_Value	Name=PSWEBPROFWRK_PTOVERIDEFLUIDFORD		Y
13	24		<input checked="" type="checkbox"/>		Page	Save			

Each step in a test is composed of the following fields:

Seq (sequence)

A system-generated sequence number. Test steps execute according to the Seq order. When you copy, move, add, or delete a step, Seq is refreshed.

ID

A system-generated unique identifier for each step, in a test. The ID value does not change when you move, add, or delete a step.

Test maintenance reports use the ID value.

Comment

Click in this field to add a comment for the step.

Active

Deselect this field to inactivate a step. PTF skips inactive steps when executing the test. This field is grayed for inactive steps.

Scroll ID

This field is required only for scroll handling.

Type

The type of application object to take an action on or to validate. Common step types are Text, Checkbox, Browser, and so on.

Action

The action the test is to take on the object. The two most common actions used on a Text object, for example, would be Set_Value and Verify.

Recognition

The means that PTF uses to identify the object within the application. Commonly, this is the HTML ID property.

Parameters

The conditions that apply to this specific step, if applicable.

Note: This field was introduced in release 8.54; in previous releases, parameters were specified as part of the Recognition field. When you save a test from release 8.53 or lower, PTF automatically moves any parameters from the Recognition field to the Parameters field.

Field Label

Contains the label text of the page control referenced in the Recognition field.

The value for the Show Field Label option in the Local Options dialog box determines whether this column appears:

- If Column is selected, the Field Label column appears.
- If Tooltip is selected, the column does not appear; instead, the field label appears in a tooltip when you position the mouse cursor over the Value field.

For more information about setting local options, see [Configuring Local Options](#).

Value

In a typical recorded step, this is the value the tester entered for an object.

In a step recorded in verify mode, this would be the value that was present in the object when it was verified.

In general, a test has a single step for each instruction in a manual test case. For example, consider the following manual test instruction:

12. Enter the value "KU0001" into the text box labeled "Employee ID".

This test instruction could be represented in PTF as a step, as shown in this example:

Image: Example of a PTF test step

This example illustrates the step to enter a value in an edit box.

Seq	ID	Comment	Active	Scroll ID	Type	Action	Recognition	Parameters	Value
1	2		<input checked="" type="checkbox"/>		Text	Set_Value	Name=EMPLID		KU0001

The PTF test language syntax and vocabulary are designed to read like a technical version of English. As a result, the function of most steps should be apparent from their construction and the context of surrounding steps.

For a detailed reference of PTF step types and actions, see [PTF Test Language](#)

PTF Test Language

This section discusses the components of the PTF test language.

Validation

Each step type allows only certain actions. Similarly, each action can only be used with certain step types.

The PTF Test Editor automatically limits your choice of actions based on the step type selected.

For example, if a step has the Type field set to *Text* and the Action field set to *Set_Value*, you can change the Action field to *Verify* since it is included in the list of available values for a text object.

This example shows a drop-down list with *Verify* as one of the values for the Action field when the Type field is set to *Text*:

Image: Action Drop-Down List for Text

This example illustrates the drop-down list showing available actions for the step type:Text.

Text	▼	Set_Value	▼
Link	▼	Exists	
Link	▼	GetLabel	
Link	▼	Get_Property	
Link	▼	Set_Value	
Link	▼	Verify	

Parameters

Typically, you place parameters in the Parameters field and use the following structure:

```
param=value;
```

Separate parameters with a semi-colon.

For example:

```
prcname=RCOM01; wait=true;
```

With a Radio object, you can also place parameters in the Value field.

See [Radio](#).

Steps that return a value require the parameter `ret=&variable;`

For example:

```
ret=&chart_val;
```

The system ignores unneeded parameters. For example, `Browser.Start` and `Browser.Start_Login` do not take any parameters, so the system ignores any values in the Recognition field for `Browser.Start` or `Browser.Start_Login`.

Prompting

The PTF Test Editor provides context sensitive help for recognition and parameters. In the Recognition column or Parameters column you can either press the F4 key, or double-click and then click the More icon that appears, to view the help and enter the corresponding values.

For examples see [Context Sensitive Help within Grid for Function Parameter Details](#)

Variables

Variables enable you to work with steps in which the information or values are not known when you create the test or the information or values for a step change each time the test executes.

You prefix variables with an ampersand (&).

In this example, the first step stores the value in the userid field to the variable *&USERID*. The second step populates the pwd field with the value in the variable *&USERID*.

Image: Example of Using Variables in Test Steps

This example illustrates how variables can be used in test steps.

Type	Action	Recognition	Parameters	Value
Text	Get_Property	Name=userid	ret=&USERID	
Pwd	Set_Value	Name=pwd		&USERID

Related Links

[Variable](#)

[Using Variables](#)

[System Variables](#)

Reserved Words

Similar to variables, you use reserved words to supply information that is not known until a test executes.

Prefix reserved words with a pound sign (#) and use them in the Value field of a test to verify a condition, change the value in an application field, or both. In this example, the *#TODAY* reserved word sets the EFFDT_FROM field to the current date.

Image: Example of using the #TODAY reserved word

This example illustrates how the reserved word *#TODAY* can be used in a test step.

Type	Action	Recognition	Parameters	Value
Text	Set_Value	Name=EFFDT_FROM		#TODAY

Related Links

[Using Reserved Words](#)

[Reserved Words](#)

System Variables

System variables are predefined variables that PTF populates at runtime. System variables provide data about the current environment, execution options, test, and application.

Because system variables are replaced with a text string at runtime, you can place a system variable wherever you would place a text string. Commonly, test developers assign a system value to a user-defined variable.

The following example shows two methods of assigning system variables to user-defined variables:

Image: Example of using system variables

This example illustrates 2 methods for assigning system variables to user-defined variables.

Variable	Set_Value	&User	%eo.dbuser%
Variable	Set_Value	&Test=%test%	

Related Links

[System Variables](#)

Syntax Check

If the syntax or value in a test step is not correct, the test will produce an error when the test is executed. The Check Syntax option allows a user to validate the parameters provided in their steps either on save or on demand prior to execution.

Check Syntax will:

- Display an error message for any invalid or missing required parameter values.
- Display a error message for any invalid optional parameter values.
- Display a warning message when the return value of a step has not been entered.
- Display an error window listing all errors and warnings found in sequential order for a test.

Note: There may be multiple errors in a single step.

- Display an error when syntax is incorrect or missing, such as verifying that all Conditional.If steps are paired with Conditional.End_if steps.

Note: Check syntax validates only active steps. Only syntax is validated, not data.

Check Syntax does not validate:

- PTF functions.
- System variables.
- Reserved words.
- Condition sign.

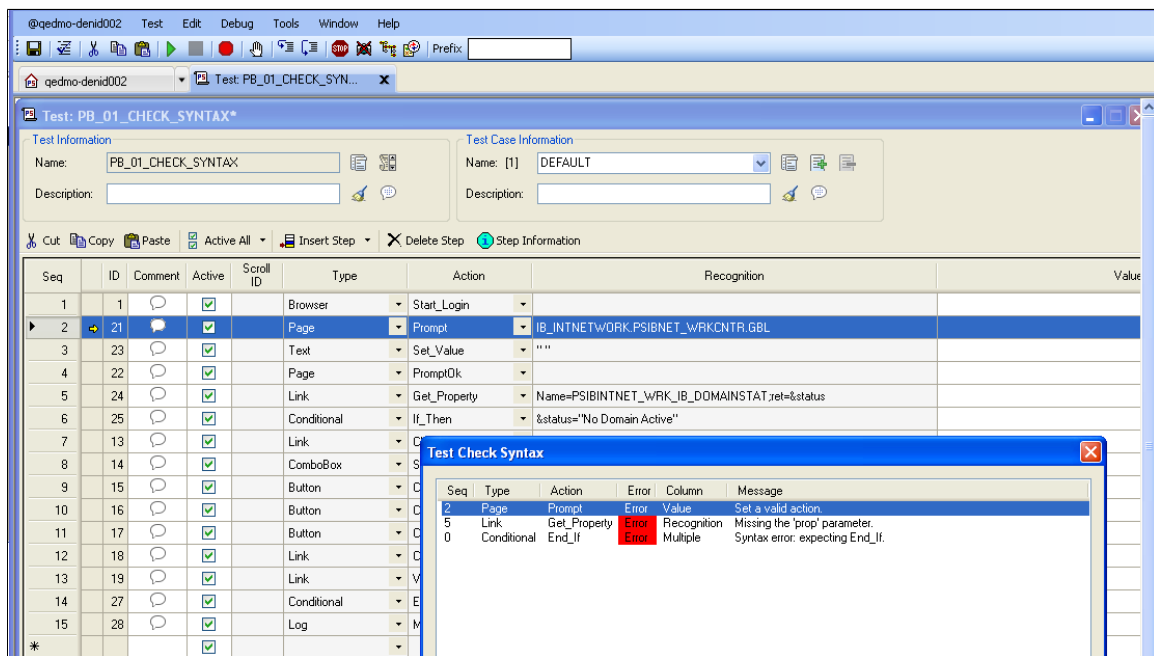
- File and folder names.
- Duplicated parameters.
- Missing values.
- Parameter that is not available for the action.
- Missing parameter name.

Running On Demand Syntax Check

After you have edited your test, select Test, Check Syntax or click the Check Syntax icon. The errors and warnings will be displayed. When you click on the error or warning in the Test Check Syntax window, the step is highlighted in the test.

Image: Test Syntax Check Showing 2 Errors

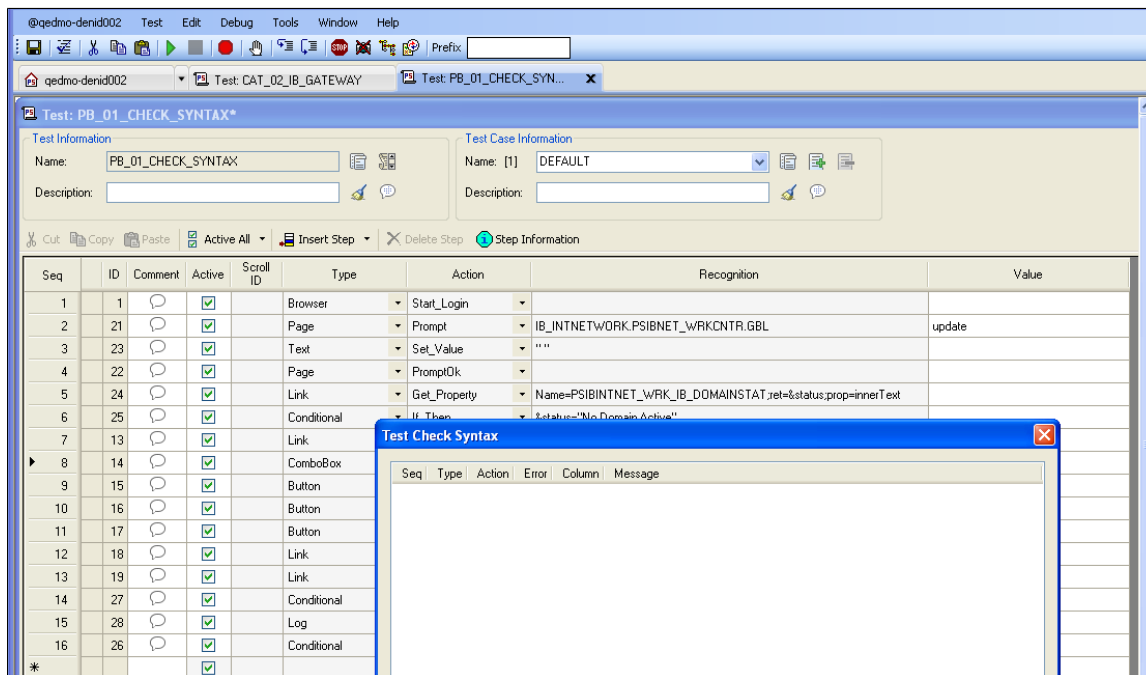
This example illustrates the check syntax dialog box with 2 errors.



If the syntax check does not detect any errors or warning, the dialog box will not return any rows of data:

Image: Test Syntax Check Showing No Errors or Warnings

This example illustrates the check syntax dialog box with no errors.



Automatically Running Syntax Check on Save

To have Syntax Check run automatically every time you save a test:

1. In PTF Explorer, click on your environment name and select Local Options.
2. In the Auto-Check Syntax field select *Yes* and click OK.
3. When you save a test, you will be prompted to run the syntax check.

Context Sensitive Help within Grid for Function Parameter Details

For any step type/action that includes parameters, you can press the F4 key or double-click in the Parameters cell to display and enter the parameters. For each parameter, an explanation and example are displayed. This is an example of the step Process.Run:

Image: Parameter Dialog Box for Step:Process.Run

This example illustrates the fields and controls on the Parameter dialog box for Step:Process.Run.

prctype	BI Publisher
wait	false
outtype	Web
outformat	XML
outfile	C:\PTFTEST\Output
expected	Success
ret	&Status

Outfile is the name of the output file location.
Example: \\abc1000svr\pubs\temp

OK Cancel

For some step types, for example Query, you can also double-click in the Value column to display and enter parameters.

Chapter 10

Test Language Reference

Step Types

This section lists each of the PTF step types and defines the actions associated with the step type. The step types are listed in alphabetical order.

Browser

Use the Browser step type to manage browser instances during test execution.

Note: Beginning with release 8.55, PTF supports running multiple browser instances from a single PTF client. This enables you to toggle between any or all open browsers. For example, you could use this functionality to verify that an action from one browser instance would change the data displayed in another browser instance.

To identify and manage each instance, you can assign a name, using a string or a variable, when it is launched. If you do not specify a name parameter, then PTF will assign “browser_0” as a name for the first browser, then “browser_1” for the second browser, and so on. If the 2nd browser is opened due to using a specific FormFactor, then PTF uses the name “FormFactor”, similarly if it is opened due to a query, the name assigned is “query”.

PTF also assigns each instance an index value automatically, and you can use the index to refer to a specific instance. The first browser instantiated is assigned an index value of 0. The next browser instantiated will have an index value of 1, and so on. If a given browser is closed, then the index value for any browsers that were instantiated after that browser will decrease by 1, in other words the index is dynamically updated by PTF.

These are the actions associated with the Browser step type.

Close

Description

Closes the current browser window (that is, the one with the execution focus).

CloseAll

Description

Closes all browser instances that were instantiated by the PTF client.

Note: Only browser instances that were instantiated by the *same* PTF client execution instance are closed.

Count

Description

Determines the number of currently open PTF client-initiated browser instances.

Note: Counts only browser instances that were instantiated by the *same* PTF client execution instance.

Parameters

`ret=&variable` Specify a variable to store the return value.

Exists

Description

Checks if a browser instance exists. Evaluates to true if it exists, false if it does not.

Parameters

`name=value` The name of the browser instance, such as browser1.

`ret=&variable` Optional parameter. Specify a variable to store the return value.

FrameExists

Description

Checks if a frame exists on a browser page. Specify the frame name in the value column.

Parameters

`expected=value` Optional parameter. If used, PTF writes either a Pass or Fail for the step in the test log, based on whether the matching frame exists. If this parameter is not included, then only step information is logged during execution.

For example:

`expected=true` Logs an error when the frame is *not* found; logs `Passed` if found, logs `Failed` if not found.

`expected=false` Logs an error when the frame *is* found; logs `Passed` if not found, logs `Failed` if found.

`ret=&variable` Optional parameter. Specify a variable to store the return value.

FrameSet

Description

Sets the focus in a browser frame.

Embedded frames include a number, such as ptModFrame_1. You can substitute ## for the number, for example ptModFrame_## and PTF will search through all frames starting with ptModFrame_ and use the one with the highest number.

Get_Active

Description

Gets the currently active browser instance's name or index. Use in combination with Set_Active when working with multiple browser instances, to control which one is active.

Parameters

name=<value>

index=<value>

ret=&variable Optional parameter. Specify a variable to store the return value.

Example

The following example shows a PTF test step that uses the Browser.Get_Active type/action.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Browser	Get_Active		ret=&hook1	

Set_Active

Description

Sets the active browser instance, when working with multiple browser instances. Use in combination with Get_Active to control which browser instance is active when working with multiple browsers instances.

Example

The following example shows a PTF test step that uses the Browser.Set_Active type/action.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Browser	Set_Active	&hook1		

Set_URL

Description

Sets the portal type. Uses the URL for the selected portal type defined in the execution option to access the component. This is the URL that is used with Page.Prompt.

Example

The following example shows a PTF test step that uses the Browser.Set_URL type/action.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Browser	Set_URL	EMPLOYEE		

Start

Description

Starts the browser instance where the test will be executed. Uses the URL from the currently active execution option.

Start_Login

Description

Starts the browser instance where the test will be executed and logs into the PeopleSoft application. Uses the URL, user ID, password, and form factor from the currently active execution option, and the language selected in the test Language field.

Parameters

name=*value*

Optional parameter.

Assigns a name to the browser instance. This enables you to manage multiple browser instances by referring to them by their assigned name. You can use a string or a variable to specify the name.

WaitForNew

Description

Waits for a new browser instance to open, then continues test execution in that browser instance.

Specify a timeout value in seconds in the Value column. The default is 10 seconds.

Note: Microsoft Internet Explorer must be set to open pop-ups in a new window for PTF to recognize new browser windows. To set this option, in Internet Explorer select Tools, Internet Options. Click the Tabs button, then select Always open pop-ups in a new window, then click OK.

Parameters

name=*value*

Optional parameter.

Assigns a name to the new browser instance. This enables you to manage multiple browser instances by referring to them by their assigned name. You can use a string or a variable to specify the name.

max=*value*

True – maximizes the window for the new browser instance.

False – keeps the existing window size for the new browser instance.

The default is true.

Example

The following example shows a PTF test step that uses the Browser.WaitForNew type/action. This step launches a new test application browser instance with the window maximized, assigns the name “browser1” to the browser instance, and waits 30 seconds before proceeding to the next test step. Test execution then continues in the new browser instance.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Browser	WaitForNew		name="browser1"; max=true	30

Button

This step type performs an action on a button. These are the actions associated with the Button step type.

Click

Description

The description for this action is in the Common Actions section.

See [Click](#).

GetText

Description

Returns the text value for the specified chart section.

Parameters

<code>chart=<i>value</i></code> ;	The index for the chart image on the page.
<code>idx=<i>value</i></code> ; <code>url=<i>value</i></code> ; <code>alt=<i>value</i></code> ;	The section recognition string. It can be the section index, the section URL, or the alternative text.
<code>ret=<i>&variable</i></code> ;	The return value.

Example

This is an example of the GetText action for a Chart step type:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Chart	GetText	chart=0	idx=3;ret=&chart_val	
Log	Message	The value for index 3 is &chart_val		
Chart	GetText	chart=0	idx=2;ret=&chart	
Log	Message	The value for index 2 is &chart_val		

SectionCount

Description

Counts the number of regions in a chart.

Parameters

<code>chart=<i>value</i></code>	The index for the chart image on the page.
<code>ret=<i>&variable</i></code>	The variable to store the return value.
	The Chart.SectionCount returns the real value (not zero-based). If you are using this variable in combination with zero-based indexes, such as a For loop, you may need to subtract 1.

Example

This example illustrates using the SectionCount action with a loop to retrieve the text for each section.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Browser	Start_Login			
Browser	FrameSet	TargetContent		
Page	Prompt	QE_CHART_MENU. QE_CHART2.GBL	urltype=default	update
Text	Set_Value	Name=QE_CHART _RECORD_ QE_CHART_ CATEGORY		CAR SALES
Page	PromptOk			
Chart	SectionCount	chart=0	ret=&chart_count	
Log	Message	ChartCount => &chart_count		
Loop	For	&var=0 to Subtract(&chart_count 1)		
Chart	GetText	chart=0	idx=&var;ret=&chart _value	
Log	Message	ChartValue => &chart_value		
Loop	Next			

CheckBox

These are the actions associated with the CheckBox step type.

Exists

Description

Checks whether the object exists on the page.

See [Exists](#).

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Property](#).

Get_Style

Description

Gets the style value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Style](#).

GetLabel

Description

Gets the label of the specified HTML object.

See [GetLabel](#).

Set_Value

Description

Sets the field value in a browser object.

See [Set_Value](#).

Verify

Description

Compares the value in the browser to the expected value, and adds a pass or fail log entry for the validation.

See [Verify](#).

ComboBox

These are the actions associated with the ComboBox step type.

Exists

Description

Checks whether the object exists on the page.

See [Exists](#).

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Property](#).

Get_Style

Description

Gets the style value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Style](#).

GetLabel

Description

Gets the label of the specified HTML object.

See [GetLabel](#).

Set_Value

Description

Sets the field value in the ComboBox to the value specified in the Value column.

You can also use this action with the `#LIST#` reserved word to verify whether items exist in the list.

When used with `#LIST#`, `Set_Value` returns an error if the expected value (the bracketed value) is not the same as the actual value.

Example

This example sets the field value to French and verifies that the values English, French, and Finnish exist in the list.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
ComboBox	Set_Value	name=PSOPRDEFN _LANGUAGE_CD		#LIST#English [French] Finnish

Related Links

[#LIST#](#)

ValueExists

Description

Checks whether a specified value exists in a combo box. Enter the value to be verified in the Value field. The full text entry or the metadata translation (XLAT) value can be specified for the value; PTF will consider either as a match.

Parameters

expected=*value*

Specify `expected=true` or `expected=false`. Logs a Pass or Fail based on whether the ret parameter matches the expected parameter.

ret=*&variable*;

ret=true – the value exists

ret=false – the value does not exist

Example

This example checks if the PA_CALCULATION_TYPE combo box contains *Automatic*, and logs a Fail if it does not exist.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
ComboBox	ValueExists	name=PA_ CALCULATION_ TYPE	expected=true;re t=&CalcType	Automatic

Verify

Description

Compares the value in the browser to the expected value and adds a Pass or Fail log entry for the validation. Use a vertical pipe (|) to separate values to be verified. Use square brackets ([]) to specify which value is expected to be selected.

Example

This example verifies that the field value is set to Two and verifies that the values One and Three exist:

Type	Action	Recognition	Parameters	Value
ComboBox	Verify	name=DUMMY_ FIELD		One [Two]]Three

Command

Use the Command step to run a command line program. The command step has one associated action: Exec.

Exec

Description

Executes a command line program.

Parameters

name=*commandname* The command name as defined on the Define Configuration Options page.

See [Defining PTF Configuration Options](#).

timeout=*seconds* The number of seconds before the program times out.

ret=*&variable* The variable to store the return value.

Example

This is an example of a command line program that requires multiple parameters.

Type	Action	Recognition	Parameters	Value
Command	Exec	name=TEXT_COMP	timeout=60; ret=&return	WORDVP c:\temp \base.txt c:\temp \actual.txt c:\temp \comparison.doc

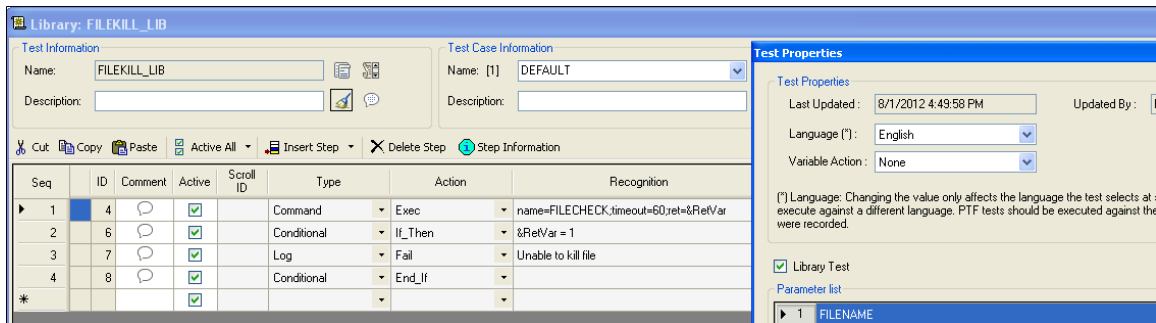
Example

If the command line program is used in multiple tests, you should create a library test for the command. This example shows a test library created for the FILEKILL command. The command has one parameter FILENAME.

Note: FILEKILL and FILECHECK commands are *examples* of possible utilities to call from PTF, they are not delivered with PTF. Customers are responsible for creating their own command line programs.

Image: Example Command in a Library Test

This example illustrates the FILEKILL command.exec step, it has one parameter FILENAME. This command will look for a specific file and delete it if it exists. If the file cannot be deleted it returns a 1.



This is an example of the calling test, which illustrates how the command can be called from a test. This test will delete the file before running a process to create a new file.

Type	Action	Recognition	Parameters	Value
Browser	Start_Login			
Variable	Set_Value	&filename		c:\Labs\files\menu.pdf
Test	Exec	FILEKILL_LIB; FILENAME=&filename		DEFAULT
Browser	FrameSet	TargetContent		
Page	Prompt	PROCESS_ SCHEDULER. PRCSMULTI.GBL		add update
Text	Set_Value	Name=PRCSRUNCNTL _RUN_CNTL_ID		test
Page	PromptOk			
Process	Run	prcname=XRFWIN	prctype=BI Publisher;outtype=File; outformat=PDF;outfile=&filename	
Text	Exec	FILECHECK_LIB;FILENAME =&filename		DEFAULT

Conditional

You can control the flow of execution of tests using conditional constructs. A conditional construct begins with an `If_Then` action and ends with an `End_If` action. You can optionally include an `Else` action.

Conditional constructs can be nested.

These are the actions associated with the Conditional step type.

Else

Description

(Optional) If the logical expression evaluates to `False`, the system executes the steps between the `Else` step and the `End_If` step.

End_If

Description

The close statement of the `If_Then` construct.

If_Then

Description

The first step in a conditional construct. The system evaluates the logical expression in the `Recognition` field of the `If_Then` step. If the expression evaluates to `True`, the system executes the steps between the `If_Then` step and the `End_If` step or the `Else` step, if it exists. If the expression evaluates to `False`, the system jumps to the `Else` step, if it exists, or to the `End_If` step if there is no `Else` step, and continues execution.

`If_Then` supports these logical operators:

`<>`, `>=`, `<=`, `>`, `<`, `=`

You can use the `AND` and `OR` logical operators to specify multiple conditions.

Example

This example shows the use of multiple conditions and nested conditionals:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Variable	Set_Value	&Integer		3
Conditional	If_Then	&Integer>=1 AND &Integer<=10 OR &Integer=0		

Type	Action	Recognition	Parameters	Value
Log	Message	Determine if integer is odd or even		
Log	Message	Enter first nested conditional		
Conditional	If_Then	&Integer=0		
Log	Message	Do not divide by 0		
Conditional	Else			
Variable	Set_Value	&Half=Divide(&Integer 2 dec=1)		
Log	Message	Enter second nested conditional		
Variable	Set_Value	&Odd=Substr(&Half 3)		
Conditional	If_Then	&Odd=5		
Log	Message	The number is odd		
Conditional	Else			
Log	Message	The number is even		
Conditional	End_If			

DataMover

This is the action associated with the DataMover step type.

Exec

Description

Executes a PeopleSoft Data Mover script. Specify the script name in the Value column. PTF uses the locations specified in the Data Mover section of the Execution Options - PeopleTools tab as the paths for the DMS files. If the DMS Output Path is not defined, it uses the system temp folder.

Example

This example shows the use of the Exec action:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
DataMover	Exec			C:\Temp\emp_imp.dms

DateTime

These are the actions associated with the DateTime step type.

Exists

Description

Checks whether the object exists on the page.

See [Exists](#).

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Property](#).

Get_Style

Description

Gets the style value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Style](#).

GetLabel

Description

Gets the label of the specified HTML object.

See [GetLabel](#).

Set_Value

Description

Sets the field value in a browser object.

See [Set_Value](#).

Verify

Description

Compares the value in the browser to the expected value, and adds a pass or fail log entry for the validation.

See [Verify](#).

Div

Use the Div step type to interact with Div objects. These are the actions associated with the Div step type:

Click

Description

Performs a mouse click on the specified object.

See [Click](#).

Exists

Description

Checks whether the object exists on the page.

See [Exists](#).

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Property](#).

Get_Style

Description

Gets the style value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Style](#).

ScrollIt

Description

Activates the scroll in a Div object to refresh the data with the next set of records. The Recognition field should contain the Div id property.

Typically this step.action is inserted during recording (recommended), by using the recorder toolbar; however, you can also use the Message tool to determine the Div id property when adding Div.ScrollIt steps as you edit a test. A Div.ScrollIt step should be added for each browser refresh that is required to access the desired record; multiple Div.ScrollIt steps may be needed.

Execution

Use the Execution actions in shell tests to modify the behavior of tests during execution. You typically set these options as you are developing tests to facilitate the development process.

The Execution step type is available only in shell tests.

Set_Options

Description

Override the default execution option. Specify the name of a valid execution option in the Recognition column.

Skip_Login

Description

Specify *True* or *False* in the Recognition column. Specify True to skip Browser.Start_Login steps. Specify False to execute Browser.Start_Login steps.

Skip_PageSave

Description

Specify *True* or *False* in the Recognition column. Specify True to skip Page.Save steps. Specify False to execute the Page.Save steps. You would, for instance, select this option to avoid creating duplicate values if you plan to run a test repeatedly.

This action overrides the Skip PageSave setting in Execution Options.

Skip_RunRequest

Description

Specify *True* or *False* in the Recognition column. Specify True to skip Process.Run steps. Specify False to execute the Process.Run steps.

This action overrides the Skip RunRequest setting in Execution Options.

StopOnError

Description

Specify *True*, *False* or *ALL* in the Recognition column.

Specify True to stop execution on the current called Test when a Fail is logged.

Specify False to continue execution when the called test encounters a Fail.

Specify ALL to stop shelltest execution when any called test encounters a Fail.

This action overrides the Stop on Error setting (located in the Debug menu of the Test Editor).

Related Links

[Configuring Execution Options in PTF Client](#)

File

The File step type corresponds to the object in the Peoplesoft Internet Architecture that enables users to upload and download files to/from the PeopleSoft application.

These are the actions associated with the File step type.

Download

Description

Used to download a file.

The File.Download step needs to be preceded by the appropriate click (such as Image.Click, Link.Click, or Button.Click). In the Value column specify a full file path name.

Select the File Download Prompt check box in the Settings dialog box to be prompted for a file download path when recording the test.

To set the file download prompt check box:

1. Click the Show Test Recorder icon.

2. Click the Configure recording settings icon.
3. Select File Download Prompt check box and click OK.
4. Record your test and you will be prompted for the file download path when you record a download step.

Example

The following examples show the use of File.Download.

This example illustrates the Download action for downloading a Query to Excel

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Link	Click	Name =QRYRUNEXCEL \$0		
File	Download			C:\TEMP\EXCEL_ TEST.XLS

This example illustrates the Download action from a button.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Button	Click	Name=FILE_ ATTACH_WRK_ ATTACHDET		
File	Download			C:\TEMP\TEXT1. TXT

Upload

Description

Uploads a file from an HTML file object.

In the Recognition column specify the name of the HTML file object. In the Value column specify a full file path name.

Example

This example shows the use of the Upload action:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
File	Upload	name=#OrigFileName		C:\TEMP\MyFile. TXT

Upload_ByLink

Description

Uploads a file from an HTML link object.

In the Recognition column specify the name of the HTML link object. In the Value column specify a full file path name.

Header

These are the actions associated with the Header step type:

Exists

Description

Checks whether the object exists on the page.

See [Exists](#).

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Property](#).

Get_Style

Description

Gets the style value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Style](#).

Verify

Description

Compares the value in the browser to the expected value, and adds a pass or fail log entry for the validation.

See [Verify](#).

HTMLTable

These are the actions associated with the HTMLTable step type.

CellClick

Description

Clicks on a specific HTMLTable cell based on the index parameter.

Parameters

`index=I/R/C`: The table, row, column index string.

For example:

```
index=&CellIndex
```

```
index=1/2/3 ;
```

In the second example, CellClick clicks on the third column of the second row of the first table.

CellClickOnChkB

Description

Clicks the check box specified in the table cell location based on the index parameter.

Parameters

`index=I/R/C`: The table, row, column index string.

For example:

```
index=&CellIndex
```

```
index=1/2/3;
```

In the second example, this function clicks on a checkbox within the third column of the second row of the first table.

`chkidx=value`; The CheckBox object index inside the cell.

`check=value`; `check=Y` – Select the checkbox.

`check=N` – Clear the checkbox.

This parameter is optional. The default value is Y.

CellClickOnImage

Description

Clicks the image specified in the table cell location based on the index parameter.

Parameters

<code>index=<i>I/R/C</i></code>	The table, row, column index string. For example: <code>index=&CellIndex</code> <code>index=1/2/3;</code> In the second example, this function clicks on an image within the third column of the second row of the first table.
<code>alt=<i>value</i></code>	Optional parameter. The alt property value of the HTML image to click.
<code>title=<i>value</i></code>	Optional parameter. The title property value of the HTML image to click.

CellClickOnLink

Description

Clicks the link specified in the table cell location based on the index parameter.

Parameters

<code>index=<i>I/R/C</i>:</code>	The table, row, column index string. For example: <code>index=&CellIndex</code> <code>index=1/2/3;</code> In the second example, this function clicks on a link within the third column of the second row of the first table.
<code>link=<i>value</i>;</code>	The link text value.

CellExists

Description

Determines whether a cell exists.

Parameters

`index=I/R/C`:

The table, row, column index string.

For example:

```
index=&CellIndex
```

```
index=1/2/3;
```

In the second example, this function verifies whether a cell exists within the third column of the second row of the first table.

`ret=&variable`

The return value.

True – the cell exists.

False – the cell does not exist.

CellGetIndex

Description

Searches the page for the text value specified in the text parameter and returns the index string for the first cell that contains the text. The index string is in the form of *I/R/C*, where *I* is the table index, *R* is the row number, and *C* is the column number. Other actions, such as `CellClick`, `CellGetValue`, and so on, use an index string to reference a specific cell.

Use the `GetCellIndex` button on the recorder toolbar to capture the text value and return a variable for the index.

Parameters

`text=value`;

The text to look for on the page.

`index=value`;

Optional. If a value is entered it is used to start the search for the text. If the value is blank, PTF will start to look for the text in the index 1/1/1.

`equal=value`;

`equal=true` performs an exact match on the text to search for. This is the default for this optional parameter.

`equal=false` uses a LIKE statement when performing the search.

`expected=value`

Optional. If used, PTF writes either a Pass or Fail for the step in the test log based on whether a matching object exists. If this parameter is not included, then only step information is logged during execution.

For example:

`expected=true` Logs an error when the expected value is *not* found; logs Passed if found, logs Failed if not found.

`expected=false` Logs an error when the expected value *is* found; logs Passed if not found, logs Failed if found.

`ret=&variable`

The return value is an index string in the form of I/R/C, where I is the table index, R is the row number, and C is the column number.

For example:

```
ret=&CellIndex
```

Example

This example illustrates using the CellGetIndex to return a variable for the index, then creating a variable for the html cell that is to the left of the original cell using the sum function, and then clicking that cell. In this example PTF will start to look for the text in table 7, row 1, column 1.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Browser	Start_Login			
Browser	FrameSet			
Link	Click	id=pttabpercontent		
Browser	FrameSet	PtModFrame_##		
HTMLTable	CellGetIndex	text=BI Publisher	index=7/1/1;equal=true;ret=&htmlindex	
Variable	Set_Value	&htmlindex		sum(&htmlindex -1 3 "/")
HTMLTable	CellClick	index=&htmlindex		
Page	Save	Name=PORTAL_HPWRK_HTMLAREA		

CellGetValue

Description

Returns the contents of an HTMLTableCell.

Use the CellGetValue button on the recorder toolbar to capture the index value and return a variable for the value.

Parameters

`index=I/R/C:`

The table, row, column index string.

For example:

```
index=&CellIndex
```

```
index=1/2/3;
```

In the second example, this function returns the contents of the third column of the second row of the first table.

`ret=&variable`

The return value.

ColCount

Description

Returns the number of columns for the HTMLTable row.

Parameters

`table=value;`

The table index.

`row=value;`

The row index.

`index=I/R/C:`

An index string in the form of I/R/C, where I is the table index, R is the row number, and C is the column number.

As an alternative to specifying the table and row parameters, you can specify an index string, such as the return value from a CellGetIndex action.

For example:

```
index=&CellIndex;
```

`ret=&variable`

The return value.

RowCount

Description

Returns the number of rows for the HTMLTable.

Parameters

`table=value;`

The table index.

`index=I/R/C:`

An index string in the form of I/R/C, where I is the table index, R is the row number, and C is the column number.

As an alternative to specifying the table parameter, you can specify an index string, such as the return value from a CellGetIndex action.

For example:

```
index=&CellIndex;
```

ret=*&variable*

The return value.

Image

These are the actions associated with the Image step type.

Click

Description

Performs a mouse click on the specified object.

See [Click](#).

Exists

Description

Checks whether the object exists on the page.

See [Exists](#).

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Property](#).

Get_Style

Description

Gets the style value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Style](#).

RightClick

Description

Performs a right-click on the image. This action is supported only for a related-content image glyph.

Label

These are the actions associated with the Label step type:

Exists

Description

Checks whether the object exists on the page.

See [Exists](#).

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Property](#).

Get_Style

Description

Gets the style value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Style](#).

Verify

Description

Compares the value in the browser to the expected value, and adds a pass or fail log entry for the validation.

See [Verify](#).

Link

These are the actions associated with the Link step type.

Click

Description

Performs a mouse click on the specified object.

See [Click](#).

Exists

Description

Checks whether the object exists on the page.

See [Exists](#).

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Property](#).

Get_Style

Description

Gets the style value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Style](#).

SaveTargetAs

Description

Performs a right-click, save as action on a link. Enter the link name in the Recognition column and the fully-qualified file name in the Value column.

Example

This example shows test steps that save a report to a target location. From the View Log/Trace page (PMN_CDM_INDEX) in the Process Monitor, it clicks on the report name link, verifies it is the correct report, and then saves the report to a file.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Link	Click	Name=PMN_ DERIVED_INDEX_ BTN		
Link	Verify	Name=URL\$0		DDDAUDIT_532. PDF
Link	SaveTargetAs	Name=URL\$0		C:\TEMP \DDDAUDIT.PDF

Verify

Description

Compares the value in the browser to the expected value, and adds a pass or fail log entry for the validation.

See [Verify](#).

Log

Use the Log step type to add entries to the PTF execution log.

Text specified in the Recognition field is written to the log and is displayed as a line in the tree view and in the Message field in the Details pane. Text in the Value field is written to the log and is displayed in the Details field in the Details pane when the corresponding line is selected in the tree view.

These are the actions associated with the Log step type.

Fail

Description

Logs an entry with a status of Fail.

Message

Description

Logs a message with a status of Info.

Note: A message is not written to the log if the Verbose field is set to *False* in execution options.

Pass

Description

Logs an entry with a status of Pass.

SnapShot

Description

Logs an entry with an image of the current screen.

Warning

Description

Logs an entry with a status of Warning.

Example

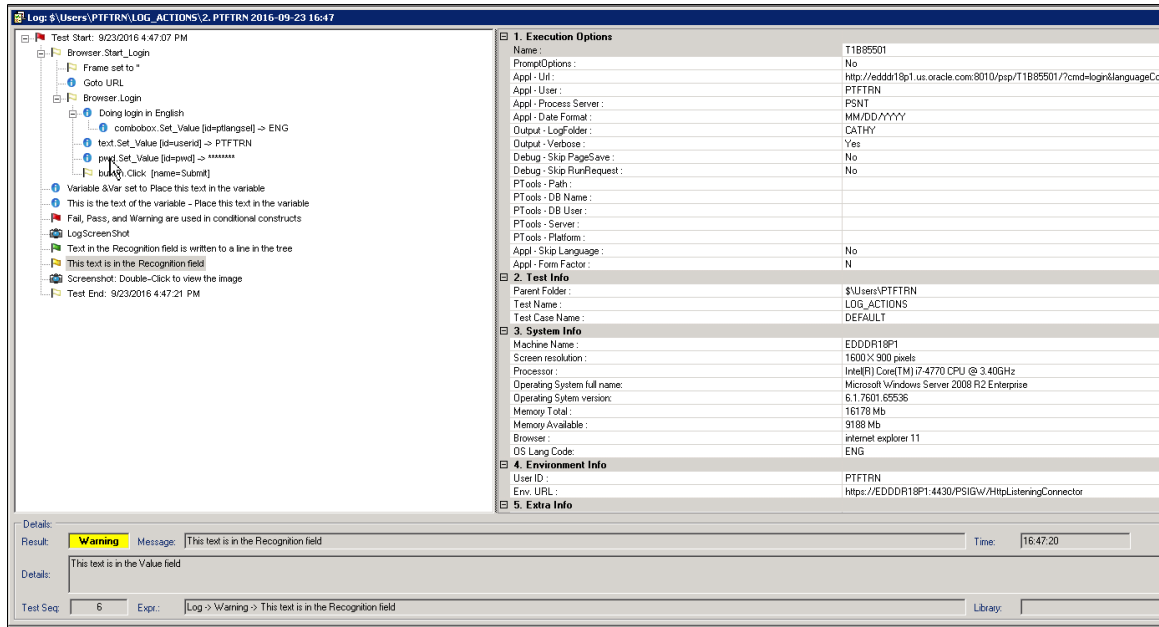
This example illustrates Log actions:

Type	Action	Recognition	Parameters	Value
Browser	Start_Login			
Variable	Set_Value	&Var		Place this text in the variable
Log	Message	This is the text of the variable - &Var		
Log	Fail	Fail, Pass, and Warning are used in conditional constructs		
Log	Pass	Text in the Recognition field is written to a line in the tree		Text in the Value field appears in the Detail pane.
Log	Warning	This text is in the Recognition field		This text is in the Value field
Log	SnapShot	Double-Click to view the image		

This log example shows how text from the Log actions appears in the Log Viewer:

Image: Example of a PTF log

This example shows the test execution log that is generated when the test above is executed.



LongText

These are the actions associated with the LongText step type.

Exists

Description

Checks whether the object exists on the page.

See [Exists](#).

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Property](#).

Get_Style

Description

Gets the style value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Style](#).

GetLabel

Description

Gets the label of the specified HTML object.

See [GetLabel](#).

Set_Value

Description

Sets the field value in a browser object.

See [Set_Value](#).

SetValue_InModal

Description

Use the `SetValue_InModal` action to set the value of a long text field on a modal page.

Verify

Description

Compares the value in the browser to the expected value, and adds a pass or fail log entry for the validation.

See [Verify](#).

Loop

These are the actions associated with the Loop step type.

Do

Description

Executes the steps between a Loop.Do step and a Loop.End_Loop step, until a Loop.Exit_Loop step is encountered.

Example

This example illustrates a Loop.Do construct.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Variable	Set_Value	&Var =0		
Loop	Do			
Log	Message	The variable is &Var		
Conditional	If_Then	&Var>3		
Loop	Exit_Loop			
Conditional	End-If			
Loop	End_Loop			

End_Loop

Description

Terminates a Loop.Do or Loop.While construct.

Exit_Loop

Description

Exits a Loop.Do, Loop.For, or Loop.While construct. Execution continues with the step after the End_Loop step. Typically, a Loop.Exit_Loop step is placed within a conditional construct.

For

Syntax

&variable=begin_value to end_value;

Description

Executes the steps between a Loop.For step and a Loop.Next step until the expression in the Recognition field evaluates to False, at which point the execution skips to the step immediately after the Loop.Next step.

Parameters

<i>&variable</i>	The variable to be used in the comparison. This variable is incremented in the Loop.Next step.
<i>begin_value</i>	The starting value.
<i>end_value</i>	The ending value.

Next

Description

Terminates a Loop.For construct. Loop.Next increments the variable in the Loop.For step.

Example

This example illustrates using the Loop.For with Loop.Next to terminate the loop.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Variable	Set_Value	&Var =0		
Loop	For	&Var =1 to 5		
Log	Message	The variable is &Var		
Loop	Next			

While

Description

Executes the steps between a Loop.While step and a Loop.End_Loop while the expression in the Recognition field evaluates to True.

When the expression evaluates to false, execution skips to the step after the Loop.End_Loop step.

Example

This example illustrates a Loop.While construct.

Type	Action	Recognition	Parameters	Value
Variable	Set_Value	&Var =0		
Loop	While	&Var <=3		
Log	Message	The variable is &Var		
Variable	Set_Value	&Var = add[&Var 1]		
Loop	End_Loop			

MsgBox

Use the MsgBox step to insert manual steps in your test for steps that cannot be automated in PTF. For example, if your test includes a step to drag and drop on a page, you will not be able to automate the drag and drop feature. By using the MsgBox step, the message will be displayed allowing the user to perform the action manually, once the user dismisses the message box PTF execution will continue.

All of the actions associated with MsgBox step type use the same parameters.

Actions for MsgBox Step Type

The following actions are associated with the MsgBox step type, all of the actions use the same parameters.

Action	Description
AbortIgnoreRetry	Creates a message box that contains 3 buttons- Abort, Ignore, and Retry.
OKCancel	Creates a message box that contains 2 buttons- OK and Cancel.
OKOnly	Creates a message box that contains 1 button- OK.
RetryCancel	Creates a message box that contains 2 buttons- Retry and Cancel.
YesNo	Creates a message box that contains 2 buttons- Yes and No.
YesNoCancel	Creates a message box that contains 3 buttons- Yes, No, and Cancel.

Parameters

prompt=*value*;

The text string displayed in the message dialog box.

To include a line break within a text string, use <NL>. For example:

```
prompt=Newline Message Prompt!!
<NL>Line1<NL>Line2<NL>Line3
```

Creates the following text string:

```
Newline Message Prompt!!
Line1
Line2
Line3
```

title=*value*;

The text string displayed in the title bar of the message dialog box.

ret=*&variable*;

The variable name that will store the integer indicating which button the user clicked.

Return Values

The return value is based on the button that was pressed by the user during test execution:

Button Pressed	Return Value
OK	1
Cancel	2
Abort	3
Retry	4
Ignore	5
Yes	6
No	7

Example

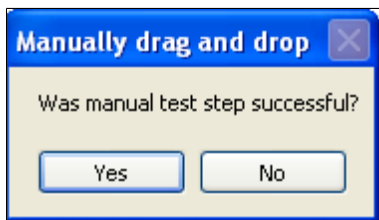
In this example variables are created to store the text for the prompt and title. The message box is created as a YesNo message box. If the Yes button is pressed by the user, the return value will be 6.

Type	Action	Recognition	Parameters	Value
Variable	Set_Value	&TestStep		Manually drag and drop
Variable	Set_Value	&Prompt		Was manual test successful?
MsgBox	YesNo		prompt=&Prompt;title=&TestStep;ret=&ret	

Type	Action	Recognition	Parameters	Value
Conditional	If_Then	&ret=6		
Log	Pass	Manual step &TestStep Passed		
Conditional	Else			
Log	Message	Manual step &TestStep Failed		
Conditional	End_If			

Image: Example of the Message Box displayed in test execution

Based on the test steps in the example above, the message box is created.



MultiSelect

MultiSelect allows you to select multiple values. The values are separated by a pipe (|). These are the actions associated with the MultiSelect step type.

Exists

Description

Checks whether the object exists on the page.

See [Exists](#).

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Property](#).

Get_Style

Description

Gets the style value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Style](#).

GetLabel

Description

Gets the label of the specified HTML object.

See [GetLabel](#).

Set_Value

Description

Sets the field value in a browser object.

See [Set_Value](#).

Verify

Description

Checks whether the object exists on the page.

See [Verify](#).

MultiSelect Example

This is an example of using MultiSelect on the Portal Layout page. In this example, you select 3 values in the first column (col0) and move them to the second column (col1), then verify the values.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Browser	Start_login			
Browser	FrameSet			
Link	Click	id=pttabperlayout		
Browser	FrameSet	ptModFrame_##		

Type	Action	Recognition	Parameters	Value
MultiSelect	Set_Value	Name=col0		Calculator[Calendar Dictionary
Image	Click	Name=moverightimg		
MultiSelect	Verify	Name=col1		Calculator[Calendar Dictionary
Page	Save	Name=PORTAL_HPWRK_HTMLAREAS17\$		

Number

These are the actions associated with the Number step type:

Exists

Description

Checks whether the object exists on the page.

See [Exists](#).

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Property](#).

Get_Style

Description

Gets the style value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Style](#).

GetLabel

Gets the label of the specified HTML object.

See [GetLabel](#).

Set_Value

Sets the field value in a browser object.

See [Set_Value](#).

Verify

Compares the value in the browser to the expected value, and adds a pass or fail log entry for the validation.

See [Verify](#).

Page

These are the actions associated with the Page step type.

Expand

Description

Attempts to expand all the collapsed sections on the current page.

Go_To

Description

Accesses a page by selecting a page tab. Enter the tab name in the Recognition field.

Example

This example illustrates the Go_To action for a page.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Page	Go_To	Roles		

Prompt

Description

Opens a component based on the *MENU.COMPONENT.MARKET* recognition string.

If the component has a search page, use the Page.PromptOk action to close the search page.

In the Value field, you must provide an action. The valid values are:

- add
- add update
- add correct
- update
- update all
- correction

Parameters

urltype=*value*

Specify the URL format (if it contains ‘psp’ or ‘psc’). Valid values are:

- content
- default
- portal

waitForTime=*value*

Specify the time in seconds. The test execution will proceed after the specified time. Using the parameter you can provide time for the page to load before the next steps in the test are executed.

Note: Use the `waitForTime` parameter with `urltype=portal`.

PromptOK

Description

Closes the search page. If the action specified in the Value field is *update*, then this action selects the first returned value.

Example

This example illustrates the use of the Prompt and PromptOK actions.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Page	Prompt	SQA_MENU.SQA_SIMPLECOMP.GBL	urltype=portal	add update
Text	Set_Value	name=SQA_SIMPLEREC_SQA_DATAID		US001
Page	PromptOK			

Save

Description

Attempts to save the current page. This action checks for the SkipSavePage flag in the execution options. For non-standard pages, the non-standard save object is recorded in the recognition column.

Example

This example illustrates a non-standard page save, where the non-standard save object is defined in the recognition field.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Page	Save	Name=PORTAL _HPWRK_ HTMLAREA		

SecPage_Close

Description

Closes the secondary page. No parameters are used.

SecPage_Open

Description

Opens a secondary page.

Parameters

page= *value* Specify the name of the secondary page.

Example

This example illustrates the SecPage_Open action.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Page	SecPage_Open	Name=SQA_ SIMPLEREC	page=SQA_ SIMPLESUBPAGE	

Process

The Process actions run processes through Process Scheduler.

Run

Description

Runs a Process Scheduler process.

Parameters

<code>prcname=<i>value</i>;</code>	The process name.
<code>prctype=<i>value</i>;</code>	The process type.
<code>wait=<i>value</i>;</code>	True - the test waits for the process to finish. False - the test does not wait for the process to finish. The default is False.
<code>outtype=<i>value</i>;</code>	The process output type.
<code>outformat=<i>value</i>;</code>	The process output format.
<code>outfile=<i>value</i>;</code>	The process output file.
<code>expected=<i>value</i>;</code>	Defines the expected status for the process when it completes. Expected status is based on status values returned in the Run Status column in Process Monitor. For example: <code>expected=Success;</code> <code>expected=No Success;</code> If the final status equals the expected status, then a Pass is logged; if not, a Fail is logged.
<code>ret=<i>&variable</i></code>	The return value. True - the process completed successfully. False - the process did not complete successfully.

Example

This example shows use of the Run action, followed by a conditional If_Then to verify that the process ran to success.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Browser	Start_Login			
Browser	FrameSet	TargetContent		

ProcessingTimeout= <i>value</i> ;	Overwrites the local option value (in minutes).
ExceptionTimeout= <i>value</i> ;	General timeout, in minutes, for all the states that are not in the local option.
QueuedResult= <i>value</i> ;	Overwrites the local option value. Valid values are <i>FAIL</i> and <i>WARN</i> .
PostingResult= <i>value</i> ;	Overwrites the local option value. Valid values are <i>FAIL</i> and <i>WARN</i> .

Related Links

[Configuring Local Options](#)

Pwd

These are the actions associated with the Pwd step type.

Exists

Description

Checks whether the object exists on the page.

See [Exists](#).

GetLabel

Description

Gets the label of the specified HTML object.

See [GetLabel](#).

Set_Value

Description

Sets the field value in a browser object.

See [Set_Value](#).

Query

These are the actions associated with the Query step type.

Exec

Description

Runs a public query in PeopleSoft Query and downloads the results. To run a private query, use the Exec_Private action.

Note: Context sensitive help is available by double-clicking in the Value column to display and enter the query parameters.

Parameters

outFile= <i>value</i> ;	The query output file.
outFolder= <i>value</i> ;	The folder where the result will be saved. If this parameter is missing, the system will use the value in the Local Options dialog box.
outFormat= <i>value</i> ;	The file format that will be used to download the result file. If this parameter is missing, the system will use the value in the Local Options dialog box.
param= <i>value</i> ;	The list of comma delimited values for all the query parameters.
0rows= <i>value</i> ;	If the query returns zero rows, add a log entry of type value. Valid values are Pass, Fail, or Warning.
Nrows= <i>value</i> ;	If the query returns one or more rows, add a log entry of type value. Valid values are Pass, Fail, or Warning.

Exec_Private

Description

Runs a private query in PeopleSoft Query and downloads the results.

Parameters

outFile= <i>value</i> ;	The query output file.
outFolder= <i>value</i> ;	The folder where the result will be saved. If this parameter is missing, the system will use the value in the Local Options dialog box.
outFormat= <i>value</i> ;	The file format that will be used to download the result file. If this parameter is missing, the system will use the value in the Local Options dialog box.
param= <i>value</i> ;	The list of comma delimited values for all the query parameters.
0rows= <i>value</i> ;	If the query returns zero rows, add a log entry of type value. Valid values are Pass, Fail, or Warning.

Nrows=*value*;

If the query returns one or more rows, add a log entry of type *value*. Valid values are Pass, Fail, or Warning.

Note: Query steps do not support simultaneous usage of both Orows= and Nrows= parameters in the same step.

Use the following formats to specify query parameters:

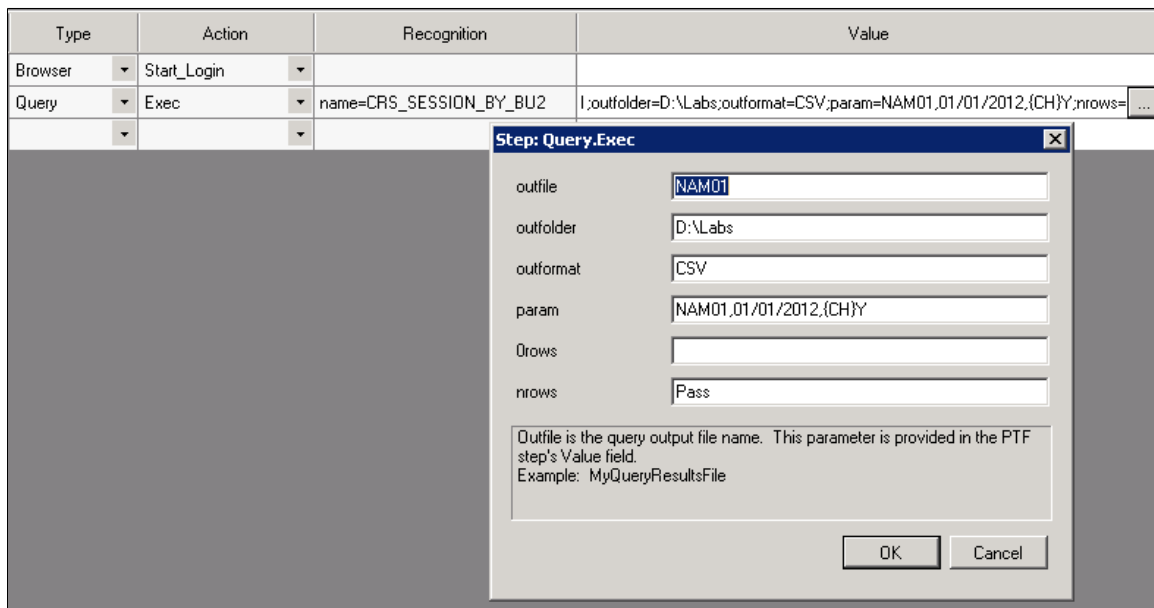
Page Control	Format
Text	param= <i>value</i>
Radio button	param={RB} <i>value</i>
Combo box	param={CB} <i>value</i>
Check box	param={CH} <i>value</i>
Text box	param={EB} <i>value</i>

Example

This example shows the Query.Exec step type, with the dialog box for the values open. In this example, the query has 3 prompts: a text field, a date field and a check box. The query is expected to return rows.

Image: Example of Query.Exec Action

This example illustrates entering the value parameters for the Query.Exec action.



Radio

These are the actions associated with the Radio step type.

Exists

Description

The value property is required to validate whether a radio button exists on the page. It can be defined in the Parameters field using `value=value` or in the Value field.

See [Exists](#).

Example

In the following example, in the first step the value property is set in the Value field. In the second step, the value is set in the Parameters field using the `Value=` parameter.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Radio	Exists	Name=SQA_ SIMPLEREC_ SQAOPTION	ret=&RadioEx1	2
Radio	Exists	Name=SQA_ SIMPLEREC_ SQAOPTION	ret=&RadioEx2;value=3	

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Property](#).

Get_Style

Description

Gets the style value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Style](#).

GetLabel

Description

Gets the label of the specified HTML object.

See [GetLabel](#).

Set_Value

Description

Sets the field value in a browser object.

See [Set_Value](#).

Verify

Description

Compares the value in the browser to the expected value, and adds a pass or fail log entry for the validation.

See [Verify](#).

Range

These are the actions associated with the Range step type.

Exists

Description

Checks whether the object exists on the page.

See [Exists](#).

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Property](#).

Get_Style

Description

Gets the style value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Style](#).

GetLabel

Gets the label of the specified HTML object.

See [GetLabel](#).

Set_Value

Sets the field value in a browser object.

See [Set_Value](#).

Verify

Compares the value in the browser to the expected value, and adds a pass or fail log entry for the validation.

See [Verify](#).

RichText

These are the actions associated with the RichText step type.

GetLabel

Description

Gets the label of the specified HTML object.

See [GetLabel](#).

Set_Value

Description

Sets the field value in a browser object.

See [Set_Value](#).

Scroll

These are the actions associated with the Scroll step type.

The Scroll ID field is required for all Scroll action types.

Related Links

[Incorporating Scroll Handling](#)

Action

Description

Takes an action based on the row specified by Key_Set. Specify the action in the Value column.

This table lists the valid values for the Action action.

Action Value	Description
ins	Insert a row. If the current row is the first row, then use the existing row.
ins+	Always insert a row.
delins	Delete all rows, and then insert into the first row.
del	Delete the row specified by the Key_Set action.
delall	Delete all rows. No further processing.
delsel	Look for the row and delete it. Do not add a fail if the row does not exist.
upd	Find a specified row to work with. If the row is not found, insert a new row or use the first row for the first time.
upd+	Find a specified row to work with. If the row is not found, always insert a new row.
sel	Find a row specified by the Key_Set action.
find	Use the scroll Find link. Format: find= <i>text_to_find</i>
not	Check that the row is not in the scroll. Add a Fail to the log if the row is found.

Parameters

`ret=&variable;`

The return value. It returns the position index for the field acted upon, based on the row identified using `Key_Set`.

`expected=value;`

Optional. Applies only to the find action. If used, PTF writes either a Pass or Fail for the step in the test log based on whether a matching object exists. If this parameter is not included, then only step information is logged during execution.

For example:

`expected=true` Logs an error when the expected value is *not* found; logs Passed if found, logs Failed if not found.

`expected=false` Logs an error when the expected value *is* found; logs Passed if not found, logs Failed if found.

Definition

Description

Use the Definition action to override the default name object of scroll buttons and the scroll parent. This is necessary, for example, when a page uses custom objects (such as links or pushbuttons) to handle conventional scroll actions such as adding or deleting rows from the scroll.

Parameters

`def=value;`

The default name object type. Example:

```
def=PARENT;
```

Valid def values are:

- ADD: Overrides the Add new row button.
- DEL: Overrides the Delete row button.
- NEXT: Overrides the Next button.
- PREV: Overrides the Previous button.
- FIRST: Overrides the First button.
- LAST: Overrides the Last button.
- PARENT: Reassigns the parent for a specific scroll area.

`type=value;`

The object type for the new action.

Example 1:

```
type=Image;
```

value=*value*; Example 2:
`type=PushButton;`
 The scroll parent variable or the new object recognition string.

Example 1:

`value=&Scr1;`

Example 2:

`value=Name=$ICField3$newm$0$$img$0;`

start=*value*; Used to set the scroll area index when the index is not \$0 .

In this example the index will start the scroll index at \$1.

`def=START;Value=1`

skipvalidation=*value*; Used to skip the validation if the defined scroll has 1 or multiple rows in the scroll when the user deletes or adds a row to it.

PTF handles scroll validation assuming the new row is added to the bottom of the scroll area. There are many cases where the row is added as the top row which will cause the PTF internal scroll validation to fail after the row is inserted. Use the SKIPVALIDATION to handle this situation.

Example 1:

This example will skip the validation when a single row is inserted or deleted from the scroll area.

`def=SKIPVALIDATION;type=SINGLE_ROW`

Example 2:

This example will skip the validation if multiple rows have been inserted or deleted from the scroll area.

`def=SKIPVALIDATION;type=MULTIPLE_ROW`

Examples

This example uses the Definition action with the `def=ADD` parameter to assign the Add action to the image `$ICField3$newm$0$$mg$0`:

Type	Action	Recognition	Parameters	Value
Scroll	Definition	value=Name=\$ICField3\$newm\$0\$\$mg\$0	def=ADD;type=Image	

In this example the Definition action with a `def=PARENT` parameter reassigns the parent:

Scroll ID	Type	Action	Recognition	Parameters	Value
1	Scroll	Key_Set	Name=PSC_SCR _REC01_PSE_ KEY_LVL1	type=Text	ROW1
1	Scroll	Action		ret=&Scr1	upd
	Text	Verify	Name=PSC_SCR _REC01_PSE_ KEY_LVL&Scr1		ROW1
	ComboBox	Set_Value	Name=PSC_ SCR_REC01 _PSE_KEY_ COMBO&Scr1	type=Text	second
	Text	Set_Value	Name=PSC_SCR _REC01_CON_ CHAR_01&Scr1		updated

ModalGrid_Close

Description

Closes a modal grid.

ModalGrid_Open

Description

Opens a modal grid.

Reset

Description

Resets all the scroll variables and closes the scroll section in the log.

RowCount

Description

Returns the number of rows for the defined scroll.

Example

This example illustrates using the RowCount action to perform a loop:

Scroll ID	Type	Action	Recognition	Parameters	Value
	Browser	Start_Login			
	Browser	FrameSet	TargetContent		
	Page	Prompt	MAINTAIN_ SECURITY. USERMAINT. GBL		update
	Text	Set_Value	Name =PSOPRDEFN_ SRCH_OPRID		STU1
	Page	PromptOk			
	Page	Go_To	Roles		
1	Scroll	Key_Set	name= PSROLEUSER _VW_ ROLENAME	type=Text	Dummy
1	Scroll	RowCount		ret=&Count	
	Variable	Set_Value	&EndCount		Add(&Count -1)
	Loop	For	&RowCount =0 to &EndCount		
	Text	Get_Property	name= PSROLEUSER_VW_ROLENAME \$&RowCount	prop=Value;ret=&Value	
	Log	Message	The role of row number &RowCount is &Value		
	Loop	Next			

Span

These are the actions associated with the Span step type.

Click

Description

Performs a mouse click on the specified object.

See [Click](#).

Exists

Description

Checks whether the object exists on the page.

See [Exists](#).

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Property](#).

Get_Style

Description

Gets the style value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Style](#).

GetLabel

Description

Gets the label of the specified HTML object.

See [GetLabel](#).

MouseOver

Description

Fires the mouseover event to show a popup page. Enter the page name in the Recognition field.

See "Using Pop-up Pages" (PeopleTools 8.55: Application Designer Developer's Guide)

MouseOverClose

Description

Fires the mouseout event to close a popup page. Enter the page name in the Recognition field.

Example

The following examples show how MouseOver and MouseOverClose can be used with popup pages.

To record a verification of a field value in a mouseover popup window:

1. From the recorder tool bar, click and drag the Target icon for the Verify action and hover over the field with a popup window.
2. Wait until the popup window appears.
3. Move the cursor to the field you want to check, then release the mouse button.
4. PTF Recorder generates the following steps:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Browser	Start			
Span	MouseOver	id=QE_EMPLOYEE_EMPLID		en
Span	Verify	Comment=QE_EMPL2_DEPTID		22000
Span	MouseOverClose	id=QE_EMPLOYEE_EMPLID		

To record an action for an object inside a mouseover popup window:

1. Click and drag the Target icon for the Verify action and hover over the field with a popup window.
2. Wait until the popup window appears, then release the mouse button.
3. Perform the action you want to record, such as clicking a URL link.
4. PTF Recorder generates the following steps:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Browser	WaitForNew			
Span	MouseOver	id=QE_EMPLOYEE_EMPLID		
Link	Click	Name=QE_EMPL_PHOTO2_URL		
Span	MouseOverClose	id=QE_EMPLOYEE_EMPLID		

Related Links

"Using Pop-up Pages" (PeopleTools 8.55: Application Designer Developer's Guide)

Verify

Description

Compares the value in the browser to the expected value, and adds a pass or fail log entry for the validation.

See [Verify](#).

Example

This step validates a Span object that contains informational text:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Span	Verify	ClassName=PSTEXT		Select an option

Test

This is the action associated with the Test step type.

Exec

Description

Calls another test or library test.

Specify the child test or library test name in the Recognition field and one or more test case names in the Value field, separated by commas.

You can also click in the Recognition field then click the ellipsis icon and select a test case to populate the Recognition and Value fields with the test name and the test case name.

Test.Exec supports the use of parameters that enable you to pass dynamic values from a calling test to a library test.

Right-click the test name in the Recognition field and select Open Test to open the child test.

Use the #IGNORE reserved word in the Value field to skip the call to the child test for a given test case.

See [Using Parameters with Library Tests](#).

Example

This test calls the test CHILD_ONE with the CASE_01 test case, then skips the call to CHILD_TWO:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Test	Exec	CHILD_ONE		CASE_01
Test	Exec	CHILD_TWO		#IGNORE

Text

These are the actions associated with the Text step type.

Exists

Description

Checks whether the object exists on the page.

See [Exists](#).

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Property](#).

Get_Style

Gets the style value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

See [Get_Style](#).

GetLabel

Description

Gets the label of the specified HTML object.

See [GetLabel](#).

Set_Value

Description

Sets the field value in a browser object.

See [Set_Value](#).

Verify

Description

Compares the value in the browser to the expected value, and adds a pass or fail log entry for the validation.

See [Verify](#).

UsageMonitor

Use the UsageMonitor step type to control managed object tracking during test execution. This step type must be used as a pair of start/stop actions. Also, the Environment is Usage Monitor Enabled option, which is located on the Advanced Execution Options dialog (or page), must be selected for usage monitoring to be active.

See [Configuring Execution Options in PTF Client](#) or [Configuring Execution Options in PeopleSoft Internet Architecture](#)

These are the actions associated with the UsageMonitor step type.

Start

Description

Initiates usage monitoring.

Stop

Terminates usage monitoring.

Variable

This is the action associated with the Variable step type.

Set_Value

Description

Assigns a value to the given variable.

Example

This example includes test steps that use the Variable step type to set a value for a variable:

Type	Action	Recognition	Parameters	Value
Log	Message	This is a test of variables.		
Variable	Set_Value	&Var1		This is Var1.
Log	Message	The value of Var1 = &Var1		
Variable	Set_Value	&Var2		This is Var2
Log	Message	The value of Var2 = &Var2		
Variable	Set_Value	&Var3		&Var1
Log	Message	Var3 is a clone of Var1. The value is &Var3		

Related Links

[Variables](#)

Wait

Use the **Wait** step type to pause test execution.

These are the actions associated with the **Wait** step type.

For_Seconds

Description

Wait the number of seconds specified in the Recognition field before proceeding to the next step.

For_Value

Description

Wait until the field contains the value specified in the Value field.

Parameters

type=*value*

Specify the object type, such as Text, Combobox, Link, and so on.

timeout=*value*

[Optional] Specify the time out value, in seconds. The default is 300 seconds.

refresh=*value*

[Optional] Specify the identifier of the target refresh button, link, or image. If the refresh= parameter is specified, PTF waits five seconds between each validation process and the click on the Refresh button.

The Refresh parameter can be defined in the below formats:

- *refresh=<buttonName>*- the identifier will be converted to *Name=<buttonName>*
- *refresh=Name=<buttonName>*-the identifier will be converted to *Name=<buttonName>*
- *refresh=Id=<buttonId>*- the identifier will be converted to *Id=<buttonId>*
- *refresh=InnerText=<buttonInnerText>*- the identifier will be converted to *InnerText=<buttonInnerText>*

Example

This example shows PTF test steps that use the **Wait** step type:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Wait	For_Seconds	60		
Wait	For_Value	id=PB_GO_ALL\$0	type=Link;timeout=120P refresh=Name=#ICRefresh	

Common Actions

The actions in this section can be used with multiple step types. The step types that support the action are listed with each action.

Click

Description

Performs a mouse click on the specified object.

The following step types support this action:

- Button.
- Image.

- [Link](#).
- [Number](#).
- [Range](#).
- [Span](#).

Example

This example shows the use of the Click action with a Button object and a Link object:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Button	Click	Name=PB_FILTER		
Link	Click	Name=LAST_ NAME\$0		

Exists

Description

Checks whether the object exists on the page.

The following step types support this action:

- [Button](#).
- [CheckBox](#).
- [ComboBox](#).
- [Header](#).
- [Image](#).
- [Label](#).
- [Link](#).
- [LongText](#).
- [MultiSelect](#).
- [Pwd](#).
- [Radio](#).
- [Span](#).
- [Text](#).

Parameters

`ret=&variable;`

`ret=True` – the object exists

`ret=False` – the object does not exist

`expected=value`

Specify `expected=True` or `expected=False`. Logs a Pass or Fail based on whether the `ret` parameter matches the `expected` parameter.

Example

This example shows the use of the Exists action:

Type	Action	Recognition	Parameters	Value
Text	Exists	Name=USERID;ret&exists		

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

Use the HTML Browser feature of the Message tool to identify the properties and values of an object.

See [Using the Message Tool](#).

Some objects have properties that are different from what you might expect. For example:

- The value property for a check box returns Y for selected, N for deselected.
- Combo boxes return the translate value of the selection for the value property.
The full text of the selected item is available as the text property.
- Radio buttons return the translate value of the selection for the value property.
- The label of the selected item is a separate label object.

The following step types support this action:

- [Button](#).
- [CheckBox](#).
- [ComboBox](#).
- [Div](#).
- [Header](#).
- [Image](#).

The following step types support this action:

- Button
- CheckBox
- ComboBox
- DateTime
- Div
- Header
- Image
- Label
- Link
- LongText
- MultiSelect
- Number
- Radio
- Range
- Span
- Text

Parameters

prop=*value*;

Where value equals the property name.

Supported property values are: font-face, font-family, font-weight, font-style, color, font-size, background-color.

ret=*&variable*;

The return value.

GetLabel

Description

Gets the label of the specified HTML object.

The following step types support this action:

- CheckBox
- ComboBox
- LongText

- [MultiSelect](#).
- [Number](#).
- [Pwd](#).
- [Radio](#).
- [Range](#).
- [RichText](#).
- [Span](#).
- [Text](#).

Set_Value

Description

Sets the field value in a browser object.

The following step types support this action:

- [CheckBox](#).
- [ComboBox](#).
- [LongText](#)

Use the [SetValue_InModal](#) action to set the value of a long text field on a modal page.

- [MultiSelect](#).
- [Number](#).
- [Pwd](#).
- [Radio](#).
- [Range](#).
- [Text](#).

Example

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
LongText	Set_Value	name=SQA_SIMPLEREC_SQA_DESCRIPTION	long	
Text	Set_Value	name=SQE_SIMPLEREC_PSE_DESCR	ebox	

Type	Action	Recognition	Parameters	Value
CheckBox	Set_Value	name=SQA_SIMPLEREC_SQA_CHECKBOX	Y	
ComboBox	Set_Value	name=SQA_SIMPLEREC_SQA_COMBO_OPTION	2	
Radio	Set_Value	name=SQA_SIMPLEREC_SQA_OPTION	2	

Verify

Description

Compares the value in the browser to the expected value, and adds a pass or fail log entry for the validation.

The following step types support this action:

- Button
- CheckBox.
- ComboBox.
- Header.
- Label.
- LongText.
- MultiSelect.
- Number.
- Radio.
- Range.
- Span.
- Text.

Example

Type	Action	Recognition	Parameters	Value
Text	Verify	ID=PSE_DATES_SQA_DATE	#TODAY	

Reserved Words

This section describes PTF reserved words, listed in alphabetical order.

#CHECK#

Description

The #CHECK# reserved word modifies the behavior of a Set_Value action to be more like a Verify action. This can be useful when you want to set data in a particular field for one test case and verify the data in the same field for a different test case.

Note: If the values are not equal, PTF will always try to update the value (unless the object is display-only). If the values are equal, PTF will not update the value.

Example

For example, a text box could be set and verified with the following two steps:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Text	Set_Value	ID=PA_CLC_ SUMMARY_CALC _NAME		KUSPTEST
Text	Verify	ID=PA_CLC_ SUMMARY_CALC _NAME		KUSPTEST

Suppose, however, that the test calls for using two test cases: the first test case sets the calculation name equal to KUSPTEST, the second test case verifies the value of KUSPTEST.

The test case that sets the value to KUSPTEST would be constructed as shown in the first step of the previous example. The test case that verifies the value as KUSPTEST would be constructed as shown in the following example:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Text	Set_Value	ID=PA_CLC_ SUMMARY_CALC _NAME		#CHECK#KUSPTEST

#DIS#

Description

This reserved word verifies a value and also checks whether the object is display-only. It logs a Fail if the object is not display-only or if the expected value does not match the application value.

If you use #DIS# without a value, then the value is ignored and #DIS# only checks for whether the field is disabled.

This reserved word is useful when, for example, PeopleCode is expected to make an object visible but not editable.

Example

The following example checks whether the Benefit Commencement Date field date is display-only and the value is equal to 07/12/2000:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Text	Set_Value	Name=PA_CALCULATION_BEN_CMDT_DATE		#DIS#07/12/2000

#DTTM

Description

Similar to #TODAY, the #DTTM reserved word inserts the current date and time into a field in the application.

Related Links

See Also:[#TODAY](#)

#EXIST#

Description

Verifies the existence of a field.

If the field exists in the application, a Pass is logged. If the field is not found, a Fail is logged.

If a value is passed after the closing # and the field exists, PTF tries to set the field to that value.

Example

In this example, the first step checks for whether the Benefit Plan field exists in the application and logs a Fail if it is not found. The second step not only checks for the existence of the field, it attempts to enter the value *KUHP* into it:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Text	Set_Value	Name=PA_CLC_PLN_INPT_BENEFIT_PLAN		#EXIST#

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Text	Set_Value	Name=PA_CLC _PLN_INPT_ BENEFIT_PLAN		#EXIST#KUHP

Related Links

[#NOTEXIST#](#)

#FAIL#

Description

This reserved word works like #CHECK# but does not update the value after performing the comparison. If a mismatch is found, a Fail is logged; otherwise, a Pass is logged. You would use #FAIL# rather than #CHECK# when you do not want to update the field if a mismatch exists.

Example

In this example, the PTF test logs a Fail if the Summary Calculation Name field is not equal to KUSPTEST:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Text	Set_Value	ID=PA_CLC_ SUMMARY_CALC _NAME		#FAIL#KUSPTEST

Related Links

[#WARN#](#)

#IGNORE

Description

Place the #IGNORE reserved word in the Value field of a Test.Exec step to skip the call to the child test. Suppose you have a parent test with two test cases. In the first test case, the parent test calls a child test. In the second test case, the parent does not call the child. Use the #IGNORE reserved word in the Value field with the second test case to skip calling the child for that test case.

In this example, the first step for the test case calls the test CHILD_ONE with test case CASE_01. The second step skips the call to CHILD_TWO for this test case.

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Test	Exec	CHILD_ONE		CASE_01

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Test	Exec	CHILD_TWO		#IGNORE

#LIKEF#

Description

The #LIKEF# and #LIKEW# reserved words are similar to the #FAIL# and #WARN# reserved words except that they look for similar values, not an exact match. If a similar match is not found, #LIKEF# logs a Fail and #LIKEW# logs a Warning.

Similar to the behavior of #FAIL# and #WARN# (and unlike the behavior of #CHECK#), if the comparison fails, PTF does not update the value. The steps only affect the error state of the execution log.

This table details ways #LIKEW# or #LIKEF# can match strings:

<i>Type of Match</i>	<i>Pattern</i>	<i>Match (Log a Pass)</i>	<i>No Match (Log a Fail or Warn)</i>
Multiple characters	a*a	aa, aBa, aBBBa	ABC
Multiple characters	*ab*	abc, AABBB, Xab	aZb , bac
Multiple characters	ab*	abcdefg, abc	cab, aab
Special character	a[*]a	a*a	Aaa
Single character	a?a	aaa, a3a, aBa	ABBBa
Single digit	a#a	a0a, a1a, a2a	aaa, a10a
Range of characters	[a-z]	f, p, j	2, &
Outside a range	[!a-z]	9, &, %	b, a
Not a digit	[!0-9]	A, a, &	0, 1, 9
Combined	a[!b-m]#	~ An9, az0, a99	abc, aj0

Example

Suppose a test requires verification of only the first several characters of a text entry. In the following example, the first step logs a Fail if the first two characters of the Benefit Plan field are not equal to US. The second step logs a Fail unless the first two characters of the Benefit Plan field are equal to US and the last character is equal to 1:

Type	Action	Recognition	Parameters	Value
Text	Set_Value	Name=PA_BENEFIT_PLAN		#LIKEF#US*
Text	Set_Value	Name=PA_BENEFIT_PLAN		#LIKEF#US*1

#LIKEF# and #LIKEW# only compare the date text of a date/time value. For example, some fields contain the current date and time. Use the #LIKEF##TODAY* construction to compare just the date portion of a Datetime field and ignore the time portion.

For example:

Type	Action	Recognition	Parameters	Value
Text	Set_Value	Name=PA_PROP_VOUCH_ DTTM		#LIKEF ##TODAY*

#LIKEW#

Description

The #LIKEW# reserved word is used the same as #LIKEF# except it logs a Warning rather than a Fail. For complete details for using #LIKEW# see #LIKEF#.

Related Links

[#LIKEF#](#)

#LIST#

Description

This reserved word checks the values of a ComboBox. It works with either the full text entries in the combo box or the metadata translation (XLAT) values of the entries.

Use #LIST# with a Set_Value action to check one or more values and then set an item in a drop-down list box, list the items separated by a vertical pipe (|) and place brackets ([]) around the item that you want to select. If the value in the field is not the same as the value in brackets, PTF logs an error and sets the value in the field to the value in brackets.

Example

This example shows the use of the #LIST# reserved word:

In this example, the first step verifies the existence of items in the list. The second step verifies that the items exist and verifies that Individual is selected.

Type	Action	Recognition	Parameters	Value
ComboBox	Set_Value	Name=PA_CALC_CALC_TYPE		#LIST# Individual Pre-Defined Group Pre- Defined List
ComboBox	Set_Value	Name=PA_CALC_CALC_TYPE		#LIST# [Individual] Pre-Defined Group Pre- Defined List

This example is similar to the previous example, but it refers to the entries by the metadata translation (XLAT) values rather than the text that actually appears in the combo box:

Type	Action	Recognition	Parameters	Value
ComboBox	Set_Value	Name=PA_CALC_CALC_TYPE		#LIST# G L
ComboBox	Set_Value	Name=PA_CALC_CALC_TYPE		#LIST#[I] G L

#NOTEXIST#

Description

The opposite of the #EXIST# reserved word, #NOTEXIST# verifies that a field does not exist.

If the field does not exist, a Pass is logged. If the field does exist, a Fail is logged.

Related Links

[#EXIST#](#)

#NOTHING

Description

PTF ignores any step with a Set_Value or Verify action where the Value field is empty, or blank. The #NOTHING reserved word enables you to use SET_VALUE to set a field to blank or select a blank value from a drop-down list box. You can use #NOTHING with a Verify action to verify that a field is blank.

The #NOTHING reserved word does not have a closing pound sign (#). It cannot be used in combination with other reserved words.

Note: Leaving the Value field of a test step blank does not have the same effect as using the #NOTHING reserved word. PTF ignores any Set_Value or Verify action where the Value field is blank.

Example

In the following example, in the first step #NOTHING selects a blank value in the Calculation Reason field and then, in the next step, it verifies that the field is blank:

Type	Action	Recognition	Parameters	Value
ComboBox	Set_Value	Name=PA_CALC_REASON		#NOTHING
ComboBox	Verify	Name=PA_CALC_REASON		#NOTHING

#PREFIX#

Description

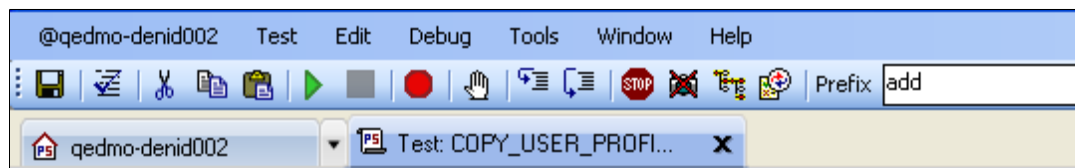
The #PREFIX# reserved word substitutes the text in the Prefix field in the Test Editor for the #PREFIX# string in the Value field. This substitution is useful when developing a test that adds new data. It enables you to modify each new added record slightly so that the test is able to successfully add unique data each time the test is executed.

Example

Suppose you entered *add* in the Prefix field, as in this example:

Image: Example of Prefix field

This example illustrates the Prefix field with the value *add*.



The following test step would enter the value *addUSER* into the DERIVED_CLONE_OPRID field:

Type	Action	Recognition	Parameters	Value
Text	Set_Value	Name=DERIVED_CLONE_OPRID		#PREFIX#USER

Note: The #PREFIX# reserved word can only be used at the beginning of the text in the Value field.

#TODAY

Description

Substitutes the current date.

Note: The #TODAY reserved word does not have a closing pound sign (#). It cannot be followed by another reserved word.

Example

Suppose you have the following test instruction:

12. Enter the current date into the Event Date field.

The following step sets the value of the Event Date field to the date at the moment of test execution:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Text	Set_Value	ID=PA_CLC_EMP_VW_ EVENT_DT		#TODAY

You can use the + or – operators in conjunction with the #TODAY reserved word to reference a date in the future or past. In this example, the test verifies that the calculation event date is 10 days in the future:

<i>Type</i>	<i>Action</i>	<i>Recognition</i>	<i>Parameters</i>	<i>Value</i>
Text	Set_Value	ID=PA_CLC_EMP_VW_ EVENT_DT		#TODAY+10

#WARN#

Description

This reserved word works the same as #CHECK# but does not update the value after performing the comparison. If a mismatch is found, a Warning is logged; otherwise, a Pass is logged.

You would use #WARN# rather than #CHECK# when you do not want to update the field if a mismatch exists.

Related Links

[#FAIL#](#)

System Variables

System variables are populated by PTF at runtime. The following table lists PTF system variables.

<i>Variable</i>	<i>Description</i>
%case%	Current test case name.
%component%	Current component name.

Variable	Description
%env.appserver%	Application server name.
%env.database%	Database name and database type. For example: T852U11 / DB2
%env.toolsrel%	The PeopleTools version.
%eo.dbname%	Execution option database name.
%eo.dbuser%	Execution option database user name.
%eo.logfolder%	Execution option log folder.
%eo.name%	Current execution option name.
%eo.platform%	Execution option platform.
%eo.pserver%	Execution option process server.
%eo.pshome%	Execution option PS_HOME path.
%eo.url%	Execution option URL.
%eo.user%	Execution option user.
%eo.verbose%	Execution option verbose flag.
%frame%	The current frame name.
%log.id%	Current log number.
%menu%	Current menu name.
%page%	Current page name.
%pid%	The last process instance number.
%test%	Current test name.
%test.path%	Full parent folder path for the executing test

Functions

This section lists the PTF functions.

Note: Regarding the use of strings in functions: Typically, variables are used within functions, however, if you are using a string directly, it must be enclosed in quotes.

Add

Syntax

Add(*number1*|*number2*[|*number3*]...)

Description

Use the Add function to add a series of numbers. Decimals and negative numbers are supported.

Parameters

<i>number1</i>	Number to be added.
<i>number2</i>	Number to be added.
<i>number3</i>	(Optional) A series of additional numbers to be added.

Returns

Sum of the numbers in the parameters.

Example

The following table presents examples of using Add.

<i>Expression</i>	<i>Result</i>
Add(10 -12 -3 4 6)	5
Add(10.43 10.55 -6.789 -178)	-163.809

Concat

Syntax

Concat(*string1*|*string2*[|*string3*...])

Description

Concatenates the strings in the parameters.

Parameters

<i>string1</i>	Beginning string for concatenation
<i>string2</i>	A string to be concatenated to <i>string1</i> .
<i>string3</i> ...	(Optional) Additional strings to be concatenated.

Returns

Returns a string resulting from concatenating the strings in the parameters.

Example

The following table presents examples of using the Concat function:

<i>Expression</i>	<i>Result</i>
Concat("hello " and "welcome " to "PTF")	hello and welcome to PTF

Date

Syntax

Date()

Description

Returns the current date, using the date format specified in the current execution option.

Returns

The current date.

Example

The following table presents an example of using the Date function, where the current date is July 4, 2011:

<i>Expression</i>	<i>Result</i>
Date()	07/04/2011

Day

Syntax

Day(*date_value*)

Description

Returns the day portion of the date value provided as a parameter.

Parameters

date_value A date value, such as the value returned by the Date() function.

Returns

Returns the day portion of the date value provided as a parameter.

Example

The following table presents examples of using the Day function; the second example assumes that the current day of the month is 13:

<i>Expression</i>	<i>Result</i>
Day(February 13, 2012)	13
Day(Date())	13

Divide

Syntax

Divide(*number1*|*number2*|*dec=dec_places*)

Description

Use the Divide function to perform division. Decimals and negative numbers are supported. Optionally specify the number of decimal places.

Parameters

<i>number1</i>	The dividend.
<i>number2</i>	The quotient.
<i>dec=dec_places</i>	(Optional) The number of decimal places. The maximum is 10. The default is zero. Note that "dec=" must be included in the parameter.

Returns

The result of dividing *number1* by *number2* rounded to *dec=dec_places* decimals.

Example

The following table presents examples of using the Divide function:

<i>Expression</i>	<i>Result</i>
Divide(75 13.5)	6
Divide(-75 13.5 dec=5)	-5.55556

GetField

Syntax

GetField(*string*, *segment*, *delimiter*)

Description

GetField returns the substring from a specified segment of a character-delimited text string.

Parameters

<i>string</i>	A character-delimited text string.
<i>segment</i>	An integer specifying which segment of the string will be returned, counting left to right. Specify a negative integer to count right to left.
<i>delimiter</i>	The character that delimits each segment in the string.

Returns

Returns the substring between the delimiters in the specified segment of the string.

Example

The following table presents examples of using GetField.

Expression	Result
GetField("a/b/c" 1 "/")	a
GetField("a/b/c" 2 "/")	b
GetField("a/b/c" 5 "/")	blank
GetField("a/b/c" -1 "/")	c

Hour

Syntax

Hour(*time_value*)

Description

Returns the hour portion of the time value provided as a parameter.

Parameters

time_value A time value, such as the value returned by the Time() function.

Returns

Returns the hour portion of the time value provided as a parameter.

Example

The following table presents examples of using the Hour function; the second example assumes that the current time is between 1:00 PM and 1:59:59 PM.:

<i>Expression</i>	<i>Result</i>
Hour(13:07:25)	13
Hour(Time())	13

InStr

Syntax

InStr(*within_string*|*substring*)

Description

Locates a substring within a string of text and returns the starting position of the substring as an integer..

Parameters

substring The text you are searching for.
The string parameter is not case sensitive.

within_string The text string you are searching within.

Returns

Returns an integer indicating the starting position of substring in within_string.

InStr returns 0 if substring does not appear in within_string. It returns 1 if substring is empty.

Example

The following table presents examples of using the InStr function.

<i>Expression</i>	<i>Result</i>
instr("ABCDEFGH" "c")	3

Parameters

<i>string</i>	A string from which to extract a substring.
<i>length</i>	A number specifying the number of characters in the substring.

Returns

Returns a substring *length* characters long from the left side of a string.

Example

The following table presents an example of using Left function.

<i>Expression</i>	<i>Result</i>
left("Hello World" 5)	Hello

Len

Syntax

Len(*string*)

Description

Returns the length of string as an integer.

Parameters

<i>string</i>	A text string.
---------------	----------------

Returns

Returns an integer indicating the length of string.

Example

The following table presents an example of using Len function.

<i>Expression</i>	<i>Result</i>
len("Hello World")	11

MakeDate

Syntax

MakeDate(*year_value* | *month_value* | *day_value*)

Description

This function returns a date value based on the year, month, and day values passed to the function as parameters.

Parameters

<i>year_value</i>	A number representing the year, such as the value returned by the Year() function.
<i>month_value</i>	A number representing the month, such as the value returned by the Month() function.
<i>day_value</i>	A number representing the day, such as the value returned by the Day() function.

Returns

Returns a date value.

Example

The following table presents examples of using the MakeDate function. In these examples, the current date was February 13, 2012:

Expression	Result
MakeDate(Add(Year(Date())) 1) Add(Month(Date())) 11) Add(Day(Date())) 1)	January 14, 2014
MakeDate(Add(Year(Date())) 1) Add(Month(Date())) -1) Add(Day(Date())) -1)	January 12, 2013

MakeTime

Syntax

MakeTime(*hour_value* | *minute_value* | *second_value* | *rollover_boolean*)

Description

This function returns a time value based on the hour, minute, and second values passed to the function as parameters.

Parameters

<i>hour_value</i>	A number representing the hour, such as the value returned by the Hour() function.
<i>minute_value</i>	A number representing the minute, such as the value returned by the Minute() function.

second_value

A number representing the second, such as the value returned by the Second() function.

rollover_boolean

(Optional) Rolls over the hour to 0 when hour_value reaches 24.

True - Returns (hour_value – 24) when hour_value is greater than 24.

False - Returns hour_value even when hour_value is greater than 24.

The default value is True.

Returns

Returns a time value.

Example

The following table presents examples of using the MakeTime function:

<i>Expression</i>	<i>Result</i>
MakeTime(Add(Hour(Time())) 12) Add(Minute(Time())) -30) Second(Time()))	7:00:00 AM
MakeTime(23 Add(60 30) 0 False)	24:30:00
MakeTime(23 Add(60 30) 0 True)	00:30:00
MakeTime(23 Add(60 -30) 0)	23:30:00

Minute**Syntax**

Minute(*time_value*)

Description

Returns the minute portion of the time value provided as a parameter.

Parameters***time_value***

A time value, such as the value returned by the Time() function.

Returns

Returns the minute portion of the time value provided as a parameter.

Parameters

<i>number1</i>	First factor.
<i>number2</i>	Second factor.
<i>dec=dec_places</i>	(Optional) The number of decimal places. The maximum is 10. The default is zero. Note that "dec=" must be included in the parameter.

Returns

The result of multiplying *number1* by *number2* rounded to *dec=dec_places* decimals.

Example

The following table presents examples of using Multiply function.

Expression	Result
Multiply(10.3 13.45)	139
Multiply(10.3 -13.45 dec=3)	-138.535

Now

Syntax

Now()

Description

Returns the current datetime, using the date format specified in the current execution option.

Returns

The current datetime.

Example

The following table presents an example of using the Now function, assuming that the function was called at 12:20 PM on July 4, 2011.

Expression	Result
Now()	07/04/2011 12:20 PM

Replace

Syntax

Replace(*source*|*find*|*replace*)

Description

Use the Replace function to replace every occurrence of a substring found in a string with a new substring.

Parameters

<i>source</i>	A string in which you want to replace substrings.
<i>find</i>	A string equal to the substring of source you want to replace.
<i>replace</i>	A string with which to replace occurrences of find in source.

Returns

Returns a string resulting from replacing every occurrence of find found in source with replace.

Example

The following table presents an example of using the Replace function.

<i>Expression</i>	<i>Result</i>
replace("original text" "i" 77)	Or77g77nal text

Right

Syntax

Right(*string*|*length*)

Description

Use the Right function to extract a substring of a specified number of characters from the right side of a string.

Parameters

<i>string</i>	A string from which to extract a substring.
<i>length</i>	A number specifying the number of characters in the substring.

Returns

Returns a substring *length* characters long from the right side of a *string*.

Example

The following table presents an example of using the Right function.

<i>Expression</i>	<i>Result</i>
right("Hello World" 5)	World

Round

Syntax

Round(*number*[[*dec=dec_places*]])

Description

Use the Round function to round a number. Decimals and negative numbers are supported. Optionally specify the number of decimal places.

Parameters

<i>number1</i>	First factor.
<i>number2</i>	Second factor.
<i>dec=dec_places</i>	(Optional) The number of decimal places. The maximum is 10. The default is zero. Note that "dec=" must be included in the parameter.

Returns

The result of rounding *number1* to *dec=dec_places* decimal places.

Example

The following table presents examples of using the Round function.

<i>Expression</i>	<i>Result</i>
Round(-130.456)	-130
Round(-130.456 dec=2)	-130.46
Round(-130.455 dec=2)	-130.45

Second

Syntax

Second(*time_value*)

Description

Returns the second portion of the time value provided as a parameter.

Parameters

time_value A time value, such as the value returned by the Time() function.

Returns

Returns the second portion of the time value provided as a parameter.

Example

The following table presents examples of using the Second function; the second example assumes that the current time is 25 seconds after the minute:

Expression	Result
Second(13:07:25)	25
Second(Time())	25

SubStr

Syntax

SubStr(*source_str*|*start_pos*[|*length*])

Description

Extracts a substring of a specified number of characters beginning at a specified location in a source string.

If *length* is not specified, SubStr returns the substring starting at the position specified in *start_pos* and continuing to the end of the string.

Parameters

source_str A string from which to extract a substring.

start_pos A number representing the character position in *source_str* where the substring starts, starting at 1.

length (Optional) A number specifying the number of characters in the substring.

Returns

Returns a string equal to a substring *length* characters long beginning at character *start_pos* of *source_str*.

Example

The following table presents examples of using SubStr function.

<i>Expression</i>	<i>Result</i>
substr("12345678" 2 3)	234
substr("12345678" 2)	2345678

Subtract

Syntax

Subtract(*number1*|*number2*[|*number3*]...)

Description

Use the Subtract function to subtract a series of numbers. Numbers can be decimal and negative.

Parameters

<i>number1</i>	Initial number.
<i>number2</i>	Number to be subtracted.
<i>number3</i> ...	(Optional) A series of additional numbers to be subtracted.

Returns

The result of subtracting *number2*, *number3*, etc., from *number1*.

Example

The following table presents examples of using the Subtract function.

<i>Expression</i>	<i>Result</i>
Subtract(10 2 3)	5
Subtract(10 -2 -3)	15

Sum

Syntax

Sum(*Index*|*Value*|*Section*|*Delimiter*)

Description

Sum works with the HTMLTable indexes.

Note: In PeopleTools releases prior to 8.53 commas were supported as a delimiter for the sum function parameters. If any tests exist in the database using that format, the commas will be converted to pipes in the upgrade.

Parameters

<i>Index</i>	The HTMLTable index string, such as 2/5/4. An index string is the return value of CellGetIndex. See CellGetIndex .
<i>Value</i>	The value that you want to add or subtract. The default action is addition.
<i>Section</i>	The section of the index that will be modified.
<i>Delimiter</i>	The character that delimits each section in the text value. The character must be enclosed in quotes.

Example

The following table presents examples of using the Sum function.

<i>Expression</i>	<i>Result</i>
sum("2/5/4" 2 1"/")	4/5/4 2 is added to the first section of the string.
sum("2/5/4" -1 3"/")	2/5/3 1 is subtracted from the third section of the string.
Sum(&index -4 3"/")	4 is subtracted from the third section of the string in the variable &index.

Time

Syntax

Time()

Description

Returns the current time, using the date format specified in the current execution option.

Description

Converts a string to uppercase.

Parameters

string The string to be converted.

Returns

Returns a string resulting from converting string to uppercase.

Example

The following table presents an example of using UCase function.

<i>Expression</i>	<i>Result</i>
ucase("Hello World 1234")	HELLO WORLD 1234

Weekday

Syntax

Weekday(*date_value*)

Description

Returns an integer value, ranging from 1 through 7, which represents the day of the week for the date value provided as a parameter, where Sunday equals 1 and Saturday equals 7.

Parameters

date_value A date value, such as the value returned by the Date() function, or "10/29/2104"

Returns

Returns an integer value, ranging from 1 through 7, which represents the day of the week for the date value provided as a parameter, where Sunday equals 1 and Saturday equals 7.

Example

The following table presents examples of using the Weekday function; the second example assumes that Tuesday is the weekday of the current date:

<i>Expression</i>	<i>Result</i>
Weekday(October 18, 2014)	6

Reserved Words Quick Reference

Reserved Words

The following table briefly explains PTF reserved words.

#CHECK#	Checks a value in an object against the expected value defined in the PTF test. Updates the value if no match exists. See <u>#CHECK#</u> .
#DIS#	Checks whether an object is display-only. See <u>#DIS#</u> .
#DTM	Enters the current date and time into the application. See <u>#DTM</u> .
#EXIST# and #NOTEXIST#	Check whether a field exists or does not exist on the page. #Exist can update the field if a value is passed following the closing # sign. See <u>#EXIST#</u> , <u>#NOTEXIST#</u> .
#FAIL# and #WARN#	Same as #CHECK# but do not update the value. If the values do not match, PTF logs a Fail or Warning. See <u>#FAIL#</u> , <u>#WARN#</u> .
#IGNORE	Place the #IGNORE reserved word in the Value field of a Test. Exec step to skip the call to the child test. See <u>#IGNORE</u> .
#LIKEF# and #LIKEW#	Match strings using LIKE. If no match exists, PTF logs a Fail or Warning. PTF does not update the value. See <u>#LIKEF#</u> , <u>#LIKEW#</u> .
#LIST#	Checks the values in a drop-down list box. Use a to separate items in the Value field. This reserved word is used only with a ComboBox object. See <u>#LIST#</u> .

#NOTHING

Deletes a value in the object or verifies that it is blank. If the object is a ComboBox and the action is Set, then PTF selects a blank item.

See #NOTHING.

#PREFIX#

Substitutes the text in the Prefix field in the Test Editor for *#PREFIX#* in the Value field.

See #PREFIX#.

#TODAY

Enters the current date into the application.

See #TODAY.