ORACLE
PEOPLESOFT ENTERPRISE

PeopleTools 8.51 PeopleBook: PeopleSoft Test Framework

August 2010

ORACLE

PeopleTools 8.51 PeopleBook: PeopleSoft Test Framework
SKU pt8.51tptf-b0810

Trademark Notice

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

License Restrictions Warranty/Consequential Damages Disclaimer

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.

Warranty Disclaimer

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.

Restricted Rights Notice

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

> *U.S. GOVERNMENT RIGHTS*
>
> Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

Hazardous Applications Notice

This software 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 which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Third Party Content, Products, and Services Disclaimer

This software and documentation may provide access to or information on 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. 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.

Contents

Chapter 3

Chapter 4

Chapter 5

Chapter 8

Chapter 9

Chapter 10

Oracle's PeopleSoft Test Framework Preface

This chapter discusses PeopleSoft Test Framework.

PeopleSoft Test Framework

PeopleSoft Test Framework (PTF) automates tasks within the PeopleSoft application, primarily functional testing.

PeopleBooks and the PeopleSoft Online Library

A companion PeopleBook called *PeopleBooks and the PeopleSoft Online Library* contains general information, including:

- Understanding the PeopleSoft online library and related documentation.
- How to send PeopleSoft documentation comments and suggestions to Oracle.
- How to access hosted PeopleBooks, downloadable HTML PeopleBooks, and downloadable PDF PeopleBooks as well as documentation updates.
- Understanding PeopleBook structure.
- Typographical conventions and visual cues used in PeopleBooks.
- ISO country codes and currency codes.
- PeopleBooks that are common across multiple applications.
- Common elements used in PeopleBooks.
- Navigating the PeopleBooks interface and searching the PeopleSoft online library.
- Displaying and printing screen shots and graphics in PeopleBooks.
- How to manage the locally installed PeopleSoft online library, including web site folders.
- Understanding documentation integration and how to integrate customized documentation into the library.
- Application abbreviations found in application fields.

You can find this companion PeopleBook in your PeopleSoft online library.

Chapter 1

Understanding PeopleSoft Test Framework

This chapter provides an overview of PeopleSoft Test Framework (PTF) and defines common PTF terms.

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:

- Test assets are PeopleTools-managed objects, which enables PTF 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 Chapter 7, "Identifying Change Impacts," page 67.

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

  See Chapter 5, "Developing and Debugging Tests," Incorporating Scroll Handling, page 57.

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

  See Chapter 10, "Test Language Reference," Process, page 116.

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

  See Chapter 10, "Test Language Reference," Query, page 119.

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. PTF supplies this information to the test by default when a test does not explicitly specify such information.
Explorer	See PTF Explorer.
Hook	Establish a connection between a 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.
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.

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.

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.

Chapter 2

Installing and Configuring PTF

This chapter presents an overview of the PTF development environment and discusses how to:

- Configure an environment for PeopleSoft Test Framework (PTF).
- Define configuration options.
- Install a PTF client.
- Configure the Web Profile.

Understanding the PTF Development Environment

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

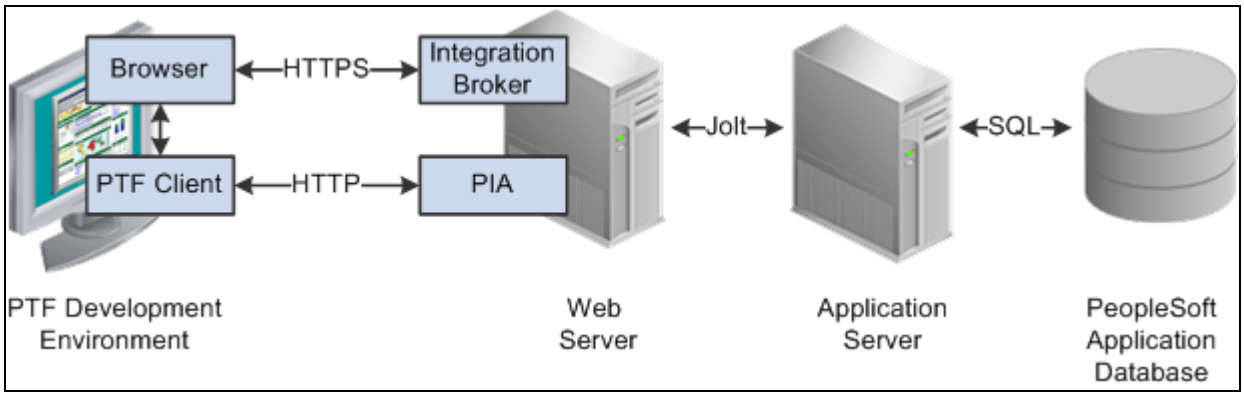


Diagram of the PTF development environment

A PTF development environment consists of the following elements:

- A PTF client instance.
- A connection to a PeopleSoft application database where test assets are stored.
- A Microsoft Internet Explorer browser instance.
- A connection to a PeopleSoft application that is to be tested.

The PTF client is a standalone program that runs on a Microsoft Windows workstation.

The PTF client connects to the PeopleSoft application database where test assets are stored using a secure HTTPS connection through Integration Broker Web Services.

The PTF client connects to the PeopleSoft application that is to be tested through a Microsoft Internet Explorer browser session. The browser connects to the PeopleSoft application using HTTP through the PeopleSoft Pure Internet Architecture (PIA).

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 certified to run on PeopleTools 8.51 can be used as a PTF environment.

This section discusses how to:

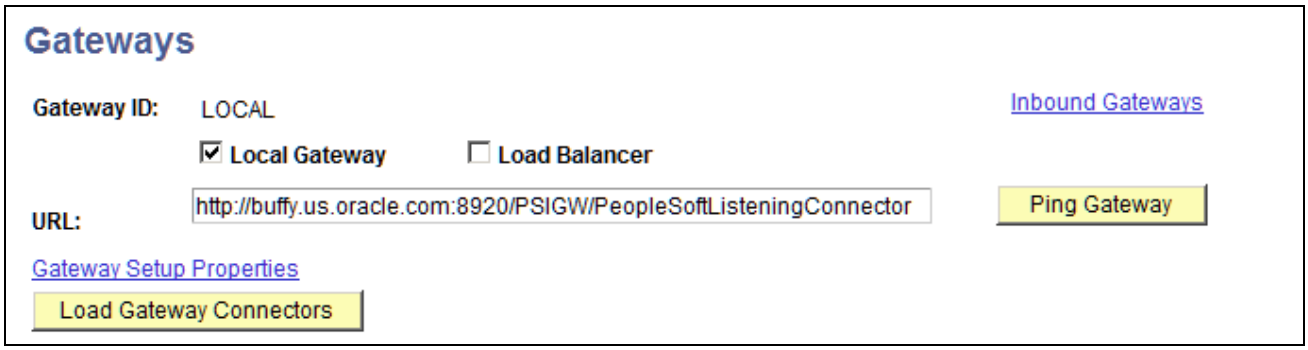
1. Verify Integration Broker setup.
2. Set up security.

Verifying Integration Broker Setup

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

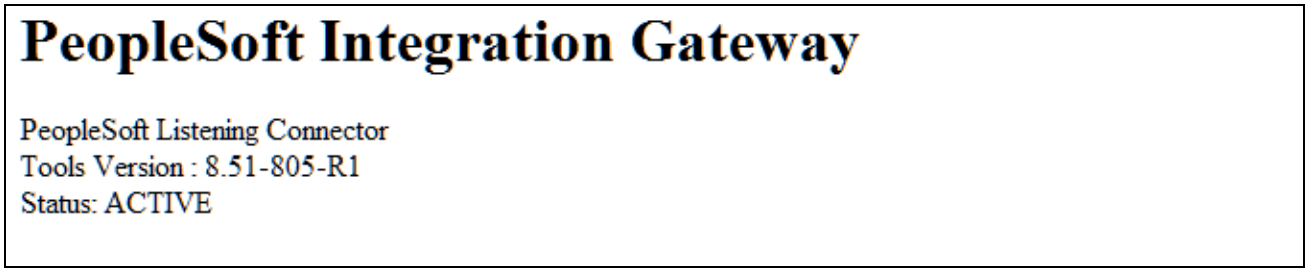
1. In your PeopleSoft application, navigate to PeopleTools, Integration Broker, Configuration, Gateways.
2. Verify that the Gateway URL field references the correct machine name.
3. Click the Ping Gateway button.
4. Verify that the message returns a status of ACTIVE.
5. Click the Gateway Setup Properties link.
6. Sign on to access integrationGateway.properties file.
7. The default user ID is *administrator,* and the default password is *password.*
8. Verify that the Gateway Default App Server URL is specified.

This is an example of the Gateways page:



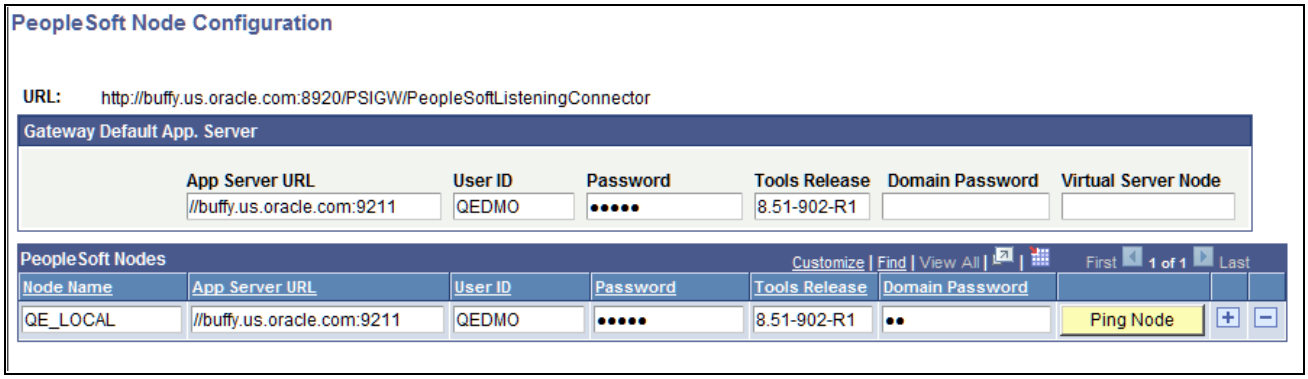
Integration Broker Gateways page

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



Integration Broker Ping message showing a status of ACTIVE

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



PeopleSoft Node Configuration page

Note. A PeopleSoft Node is required for PTF only if the web server is connected to more than one database, in which case you would enter the node name in the Node ID field of the PeopleSoft Test Framework - Signon dialog box.

See Chapter 2, "Installing and Configuring PTF," Creating a Connection to a PTF Environment, page 13.

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 Chapter 2, "Installing and Configuring PTF," Setting Up Security, page 8.

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

See Also

PeopleTools 8.51 PeopleBook: PeopleSoft Integration Broker Administration, "Getting Started with PeopleSoft Integration Broker Administration," Administering PeopleSoft Integration Broker

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 Tests	No*	Yes	Yes
Modify Tests	No*	Yes	Yes
Delete Tests	No*	Yes	Yes
Create or Modify Execution Options	No	No	Yes
Use Log Manager	No	No	Yes
Define Configuration Options	No	No	Yes

Privilege	PTF User	PTF Editor	PTF Administrator
Create Test Maintenance Reports	No	No	Yes
Create Test Coverage Reports	No	No	Yes

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

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

See Also

PeopleTools 8.51 PeopleBook: Security Administration, "Administering User Profiles"

Chapter 2, "Installing and Configuring PTF," Defining Configuration Options, page 9

Chapter 7, "Identifying Change Impacts," Creating Test Maintenance Reports, page 70

Chapter 7, "Identifying Change Impacts," Creating Test Coverage Reports, page 79

Defining PTF Configuration Options

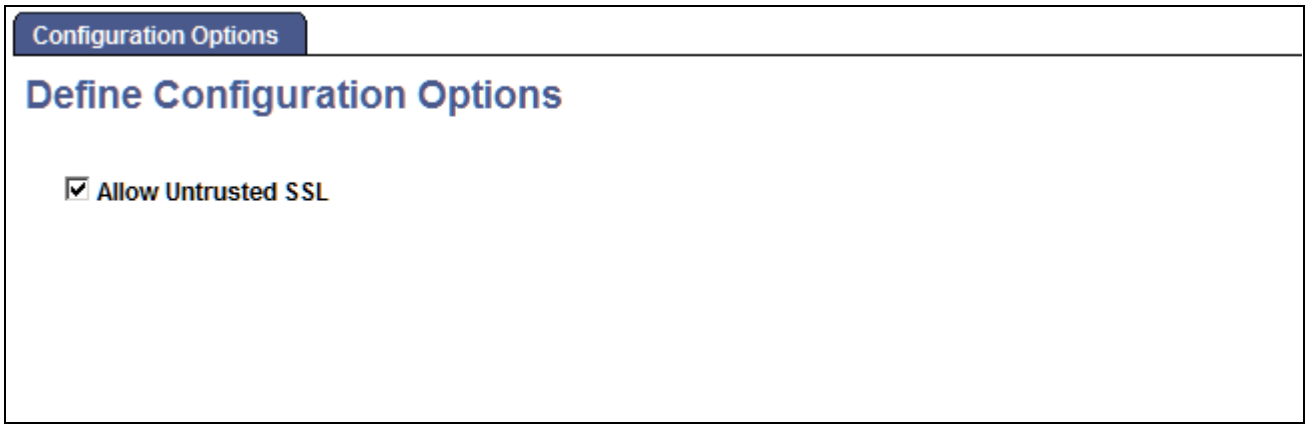
This section discusses the page used to configure PTF options.

Page Used to Define Configuration Options

Page Name	Definition Name	Navigation	Usage
Configuration Options	PSPTTSTCONFIG	PeopleTools, Lifecycle Tools, Test Framework, Define Configuration Options	Specify whether to allow untrusted SSL certificates.

Defining Configuration Options

Access the Configuration Options page (PeopleTools, Lifecycle Tools, Test Framework, Define Configuration Options).



Configuration Options page

Allow Untrusted SSL Select to allow untrusted SSL certificates.

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.

This section discusses how to:

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

Verifying Requirements

PTF client installation has the following requirements:

1. Microsoft Windows operating system.
2. Microsoft Internet Explorer.

   PTF does not support any browsers other than Microsoft Internet Explorer.

3. Microsoft .NET.

   If Microsoft .NET is not present, the PTF Installer returns an error.

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.

   PTF will need runtime access to the PTF data directory (`C:\Documents and Settings\<User>\Application Data\PeopleSoft\PeopleSoft Test Framework` by default).

Configuring the Browser Security Settings

You must configure the client browser Security settings to accept the test application URL.

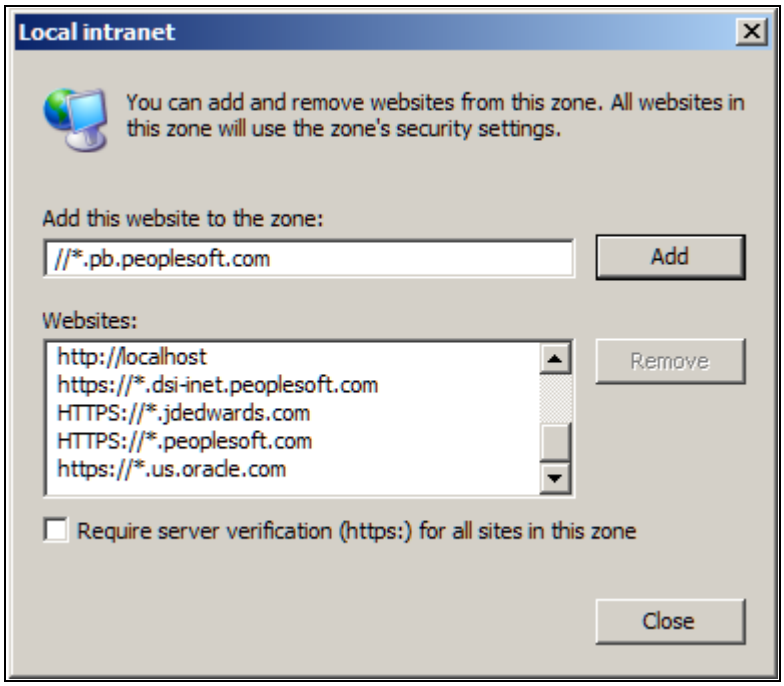
To configure the browser Security settings:

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 button.
4. Click the Sites button.
5. Click the Advanced button.
6. In the Add this website to the zone field, enter the URL for the test application.

   You may choose to use a wildcard to add all websites from a particular domain.

   For example: `http://*.pb.peoplesoft.com`.
7. Click the Add button.
8. Click the Close button.
9. Click the OK button to close each open dialog box.

This example shows the Local intranet dialog box:



Microsoft Internet Explorer Local intranet dialog box

Installing PTF Client Software

To install the PTF client software:

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

   If you are installing on a machine that has a PeopleTools 8.51 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.

   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.

4. Click the Next button.

   The Ready to Install the Program page appears.

5. Click the Install button.

   The InstallShield Wizard Complete page appears.

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

   Your PTF client software installation is complete.

7. 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 3).

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:

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

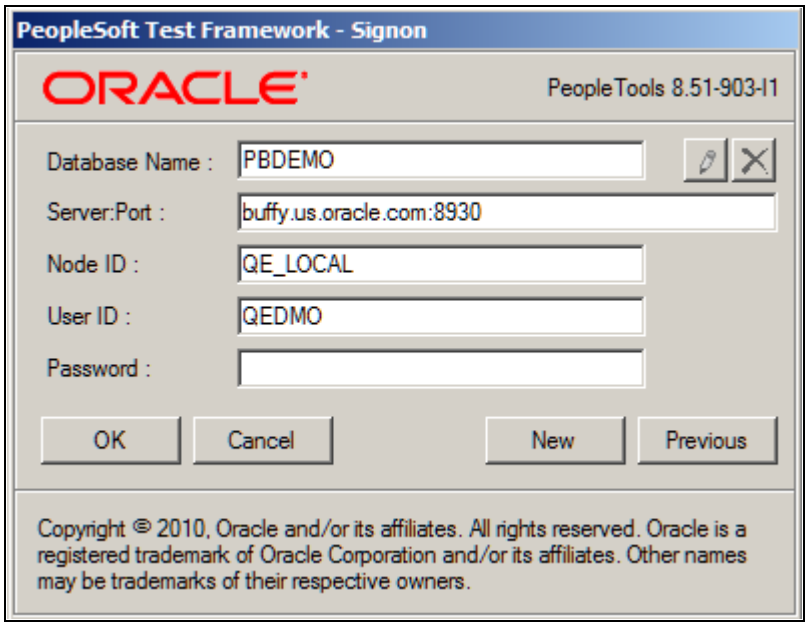
Server:Port	Enter the server name for the environment. The format for the server name is:
	`<machine_name>:<https_port>`
	For example:
	`rtdc79614.peoplesoft.com:443`
	If the https port is the default 443 the port is optional.
	You can also enter a complete https URL in this format:
	`https://<machine_name>:<https_port>/PSIGW/HttpListeningConnector`
	For example:
	`https://rtdc79614vmc.dsi-inet.peoplesoft.com:443/PSIGW/HttpListeningConnector`

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. See *PeopleTools 8.51 PeopleBook: PeopleSoft Integration Broker Administration*, "Managing Integration Gateways," Setting Oracle Jolt Connection Properties.
User	Enter a valid user name for the PeopleSoft application that contains the environment.
Password	Enter the password for this user.

4. Click the OK button.

   PTF launches with a connection to the designated environment.

This example shows a completed PeopleSoft Test Framework - Signon dialog box:

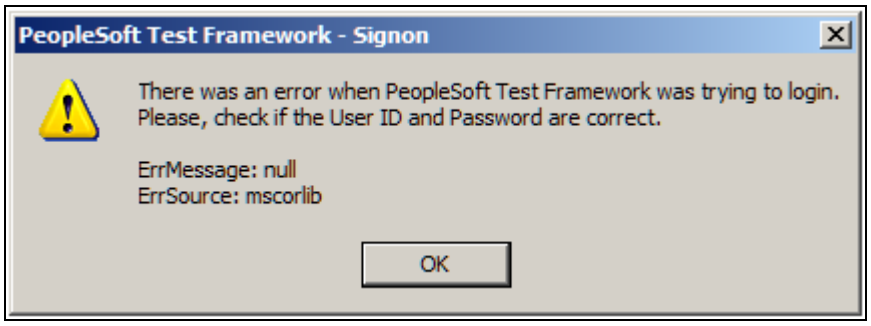


Example of a completed environment signon dialog box

Note. Node ID is required only if more than one database is connected to the server.

Troubleshooting Tips

You may receive the following signon error:

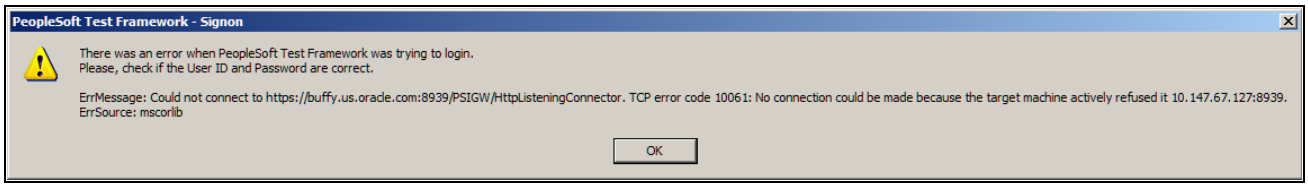


Signon error message

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.

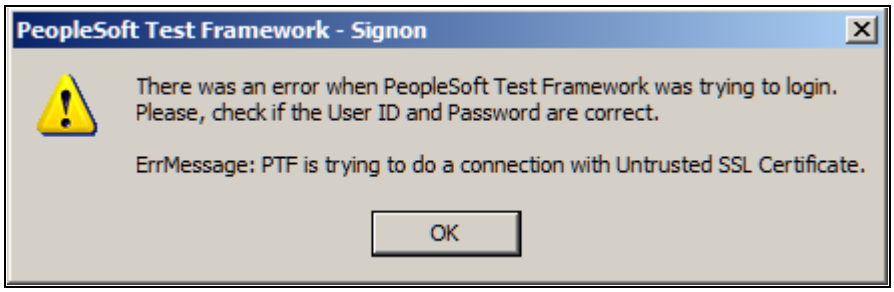
You will receive the following error message if you specified the wrong HTTPS port in the environment login URL:



HTTPS port error message

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

If you receive the following error message, select Allow Untrusted SSL on the Configuration Options page.

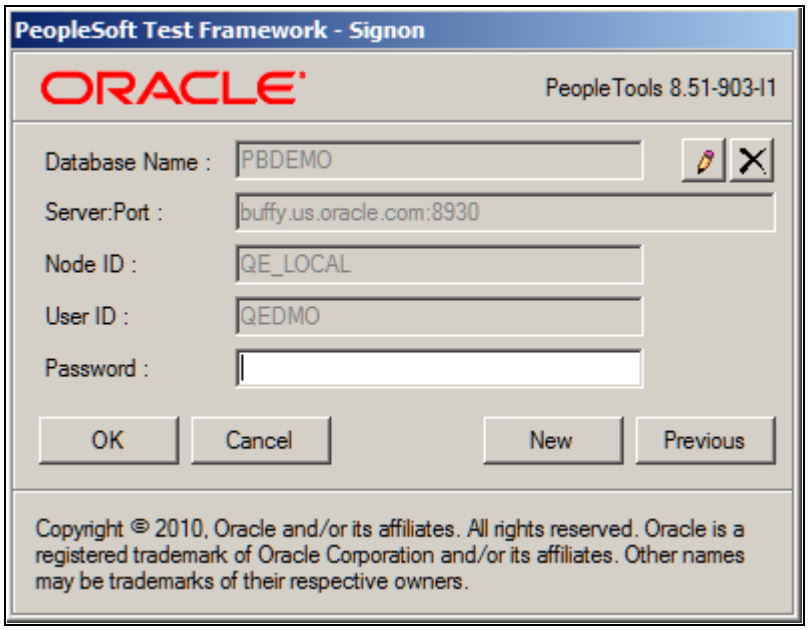


Untrusted SSL Certificate error message

See Chapter 2, "Installing and Configuring PTF," Defining Configuration Options, page 9.

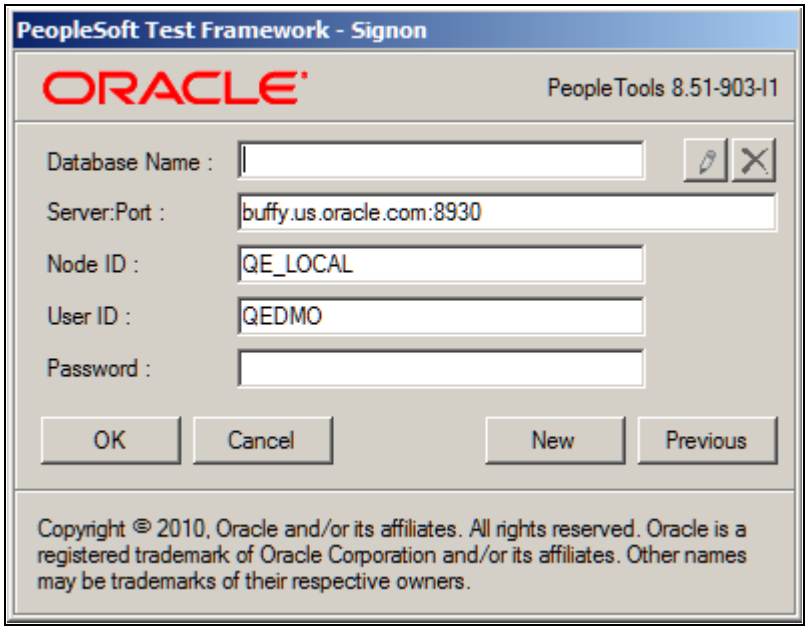
Selecting a PTF Environment

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



PeopleSoft Test Framework - Signon dialog box showing the most recently used environment

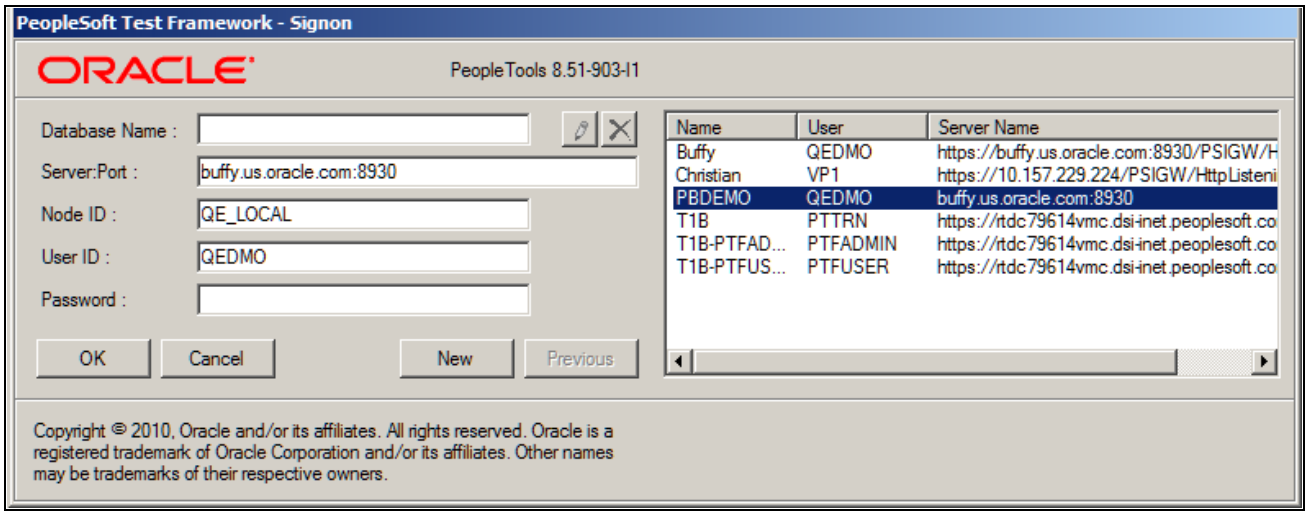
You can enter the password and click the OK button to launch PTF using that environment, or you can click the New button to create another environment login.



New PeopleSoft Test Framework - Signon dialog box

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.



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

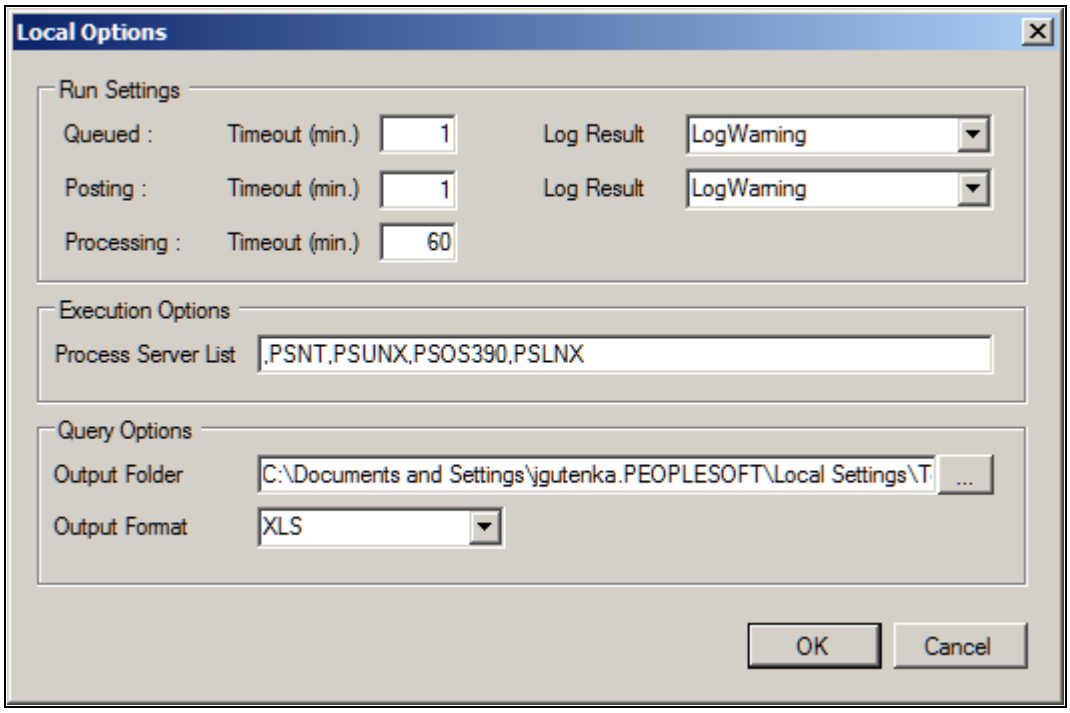
Environment Login settings are specific to the machine on which the PTF client is installed. The Environment Login settings are stored in the environments.xml file in the PTF data directory (`C:\Documents and Settings\<User>\Application Data\PeopleSoft\PeopleSoft Test Framework`) by default.

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

Configuring Local Options

Select Local Options from the PTF menu to access the Local Options dialog box. Use Local Options to configure timeouts for processes launched from a PTF test.

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.



Local Options dialog box

Run Settings

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.

Execution Options

Process Server List Enter a comma-separated list of valid process servers for this environment. Process servers you enter on this list populate the Process Servers drop-down list box in the Execution Options dialog box.

See Chapter 2, "Installing and Configuring PTF," Configuring Execution Options, page 20.

Configuring Execution Options

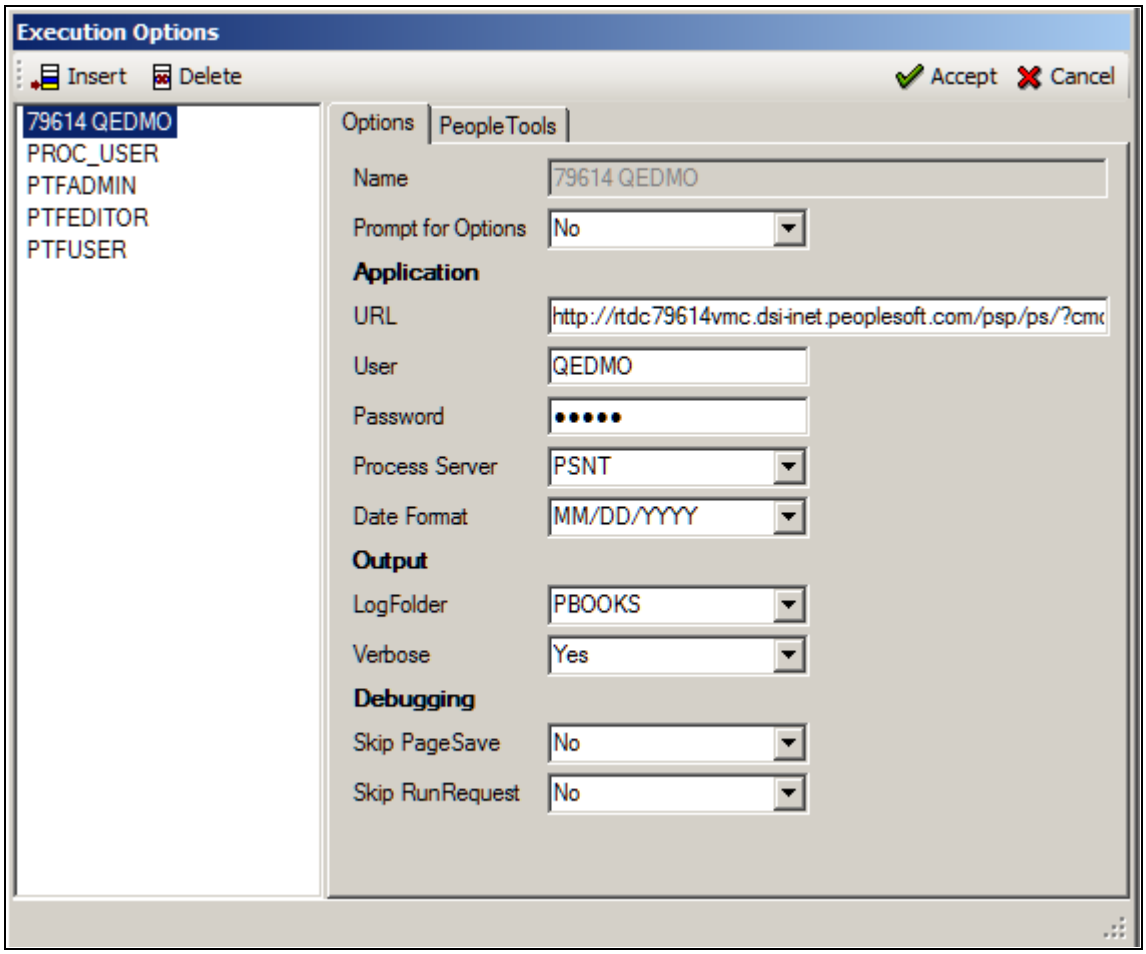
Use Execution Options to configure settings for the PeopleSoft applications you test with PTF.

Select Execution Options from the PTF menu. (The PTF menu is labeled with the name of the current PTF environment.) 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 default execution option.

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.

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.

This example shows the Execution Options page - Options tab:



Example of the Execution Options page - Options tab

Available execution options are listed in the left pane. The settings for the selected execution option are in the right pane.

These buttons are available on the toolbar:

Insert	Click to add a new execution option.
Delete	Click to remove an execution option from the list.
Accept	Click to save changes and close the dialog box.
Cancel	Click to close the dialog box without saving changes.

The following fields are on the Options tab:

Name	Enter a name for this execution option. You can use any name.
Prompt for Options	Specify whether the Execution Options dialog box should appear when a user executes a test.

Application

URL	Enter the URL for the PeopleSoft application to be tested.
User	Enter a valid user name 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 Local Options dialog box. See Chapter 2, "Installing and Configuring PTF," Configuring Local Options, page 18.
Date Format	Select a date format.

Output

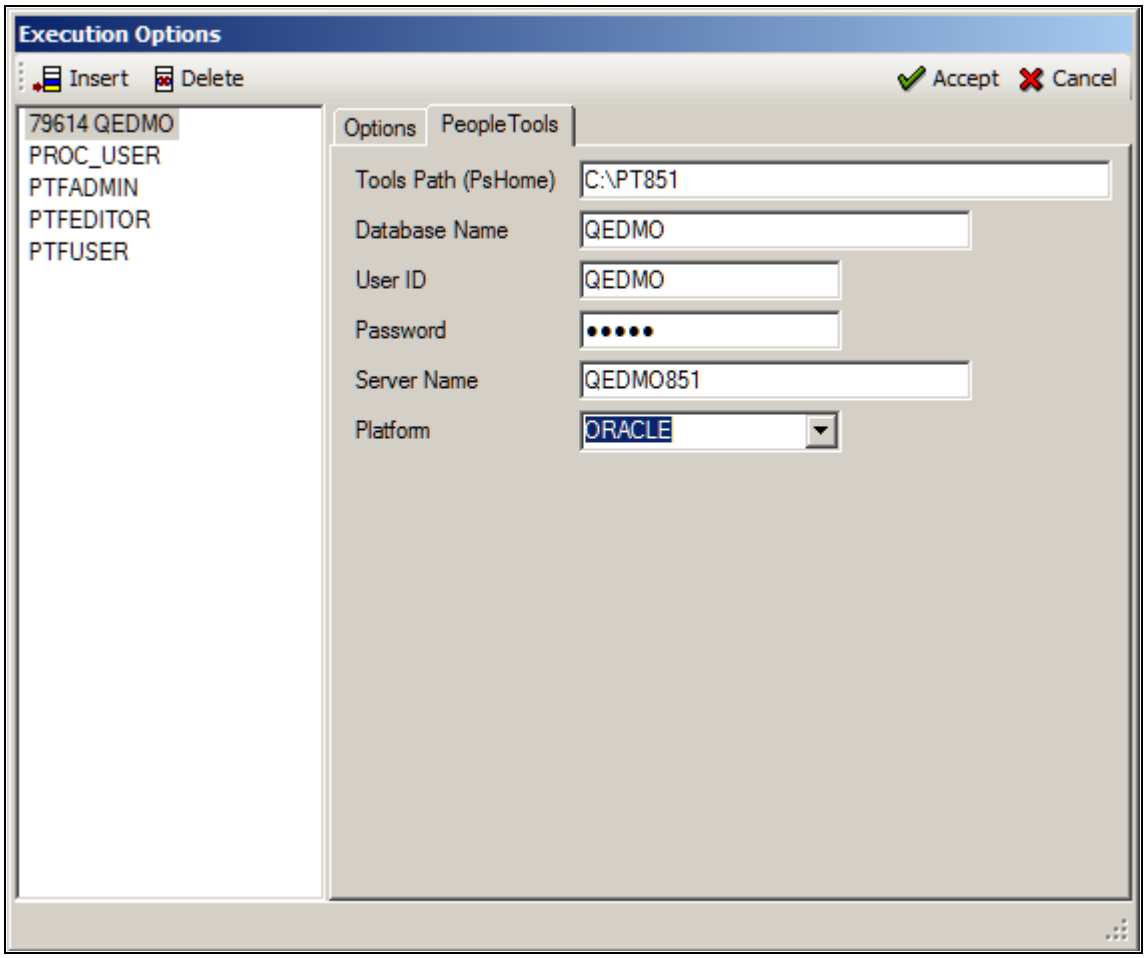
LogFolder	Select or enter the folder name to which log files will be written. If the folder does not exist it will be created.
Verbose	Specify whether to use verbose logging. Select *Yes* to record a detail line in the log for each step executed in the test. Select *No* to record only the test rollup status (Pass or Fail).

Debugging

Skip PageSave	Select *Yes* to prevent a test from executing a save. You would, for instance, select this option to avoid creating duplicate values if you plan to run a test repeatedly.
Skip RunRequest	Select *Yes* to prevent the test from executing process requests.

PeopleTools Tab

Access the PeopleTools tab:



Example of the Execution Options page - PeopleTools tab

The PeopleTools tab supplies the information required to connect to DataMover.

The following fields are on the PeopleTools tab:

Tools Path (PsHome) Enter the path to PS_HOME for this environment.

Database Name Enter the name of the database for this environment.

User ID Enter a valid database user name.

Password Enter the password for this user.

Server Name Enter the name of the database server.

Platform Select the database platform.

Default Execution Option

When a user clicks 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.

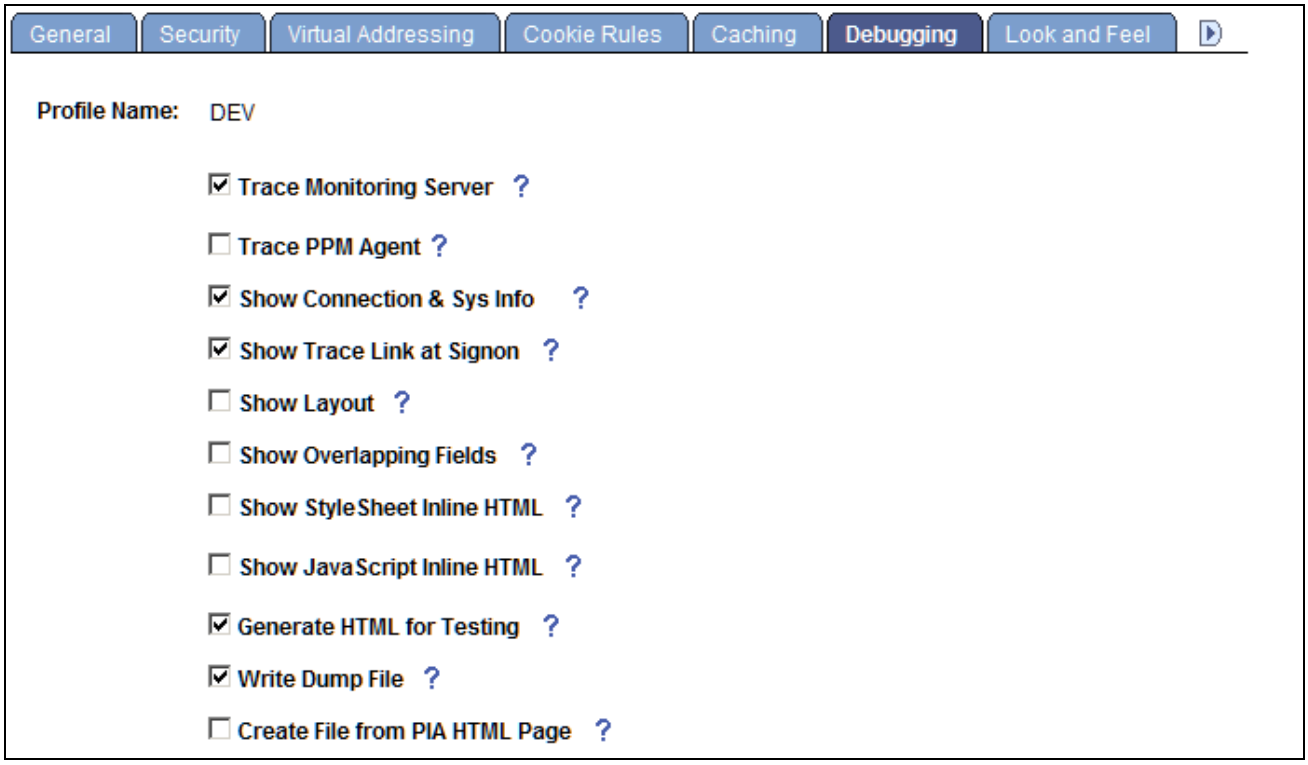
See Also

Chapter 8, "Incorporating Best Practices," Use Execution Options, page 90

Configuring the Web Profile

Configure the PeopleSoft application you are testing to generate HTML for testing.

1. Navigate to the PeopleTools, Web Profile, Web Profile Configuration, and access the Debugging tab.
2. Check the Generate HTML for Testing checkbox, as shown in the following example:



Web Profile Configuration - Debugging page

Chapter 3

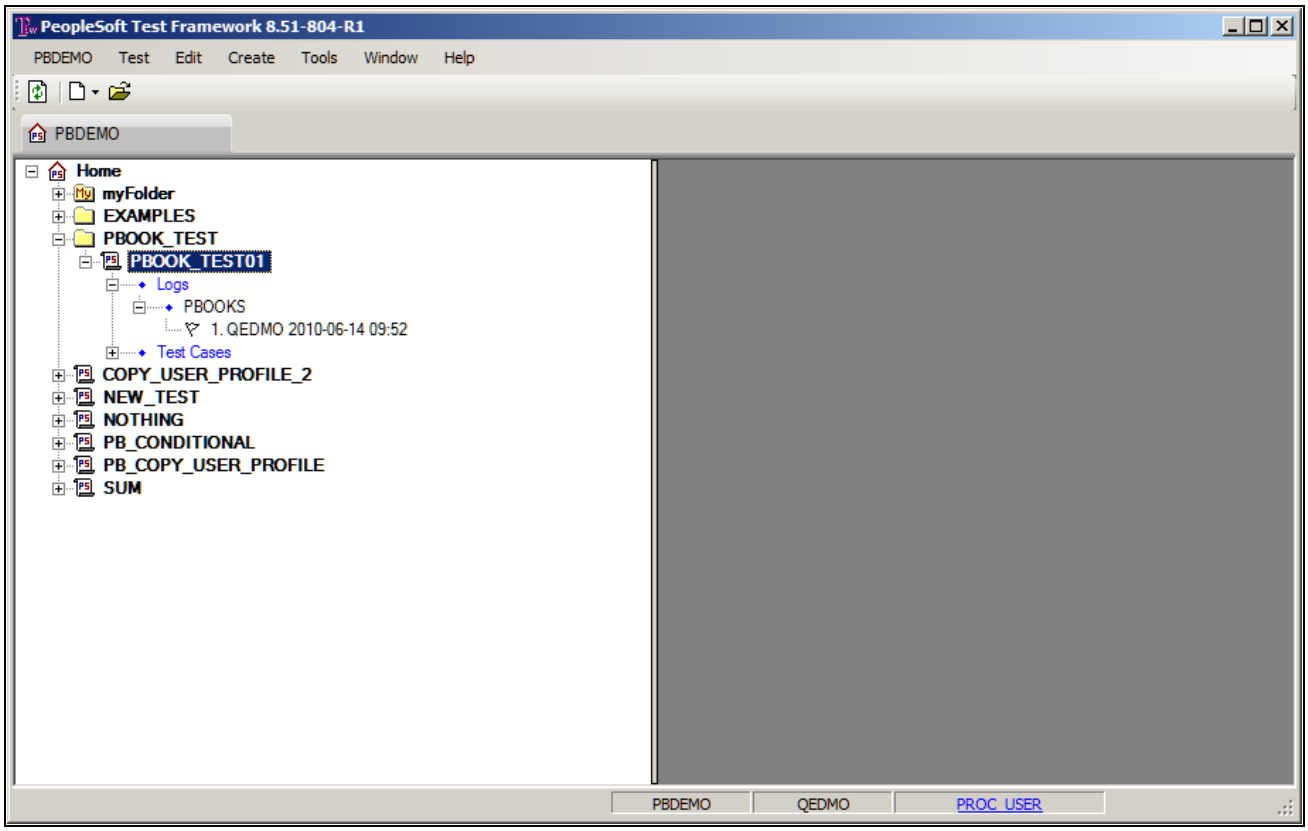
Using PeopleSoft Test Framework

This chapter discusses how to use these PeopleSoft Test Framework (PTF) tools:

- PTF Explorer
- PTF Explorer menus
- Test Editor
- PTF Recorder
- Log Viewer

Using 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. PTF Explorer is the first pane visible to the user after startup. It is labeled with the name of the PTF environment:



Example of the PTF Explorer user interface for the PBDEMO environment

You use PTF Explorer to:

- Create tests and folders.
- Delete tests and folders.
- Copy and move tests.
- Navigate to and open test assets.

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.

Using PTF Explorer Menus

The following menus appear when PTF Explorer has focus. Note that many menu commands are specific to a currently selected item. The PTF Explorer menu name corresponds to the name of the current PTF environment. In the previous example, the PTF Explorer menu name is PBDEMO.

This table describes the PTF Explorer menu commands:

PTF Explorer Menu Command	*Usage*
Refresh	Refreshes the current view.
Local Options	Opens the Local Options dialog box.
Execution Options	Opens the Execution Options dialog box.

This table describes the Test menu commands:

Test Menu Command	*Usage*
Open	Opens the selected test or test case.
Refresh Selection	Refreshes the view for the selected test.
Delete	Deletes the selected test.

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

Edit Menu Command	*Usage*
Copy Link to Clipboard	Copies the link to the selected test 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 Window menu command:

Window Menu Command	*Usage*
Quick Open	Opens a test using data that was copied to the clipboard using the Copy Link to Clipboard command. 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.

This table describes the PTF Explorer Tools menu commands:

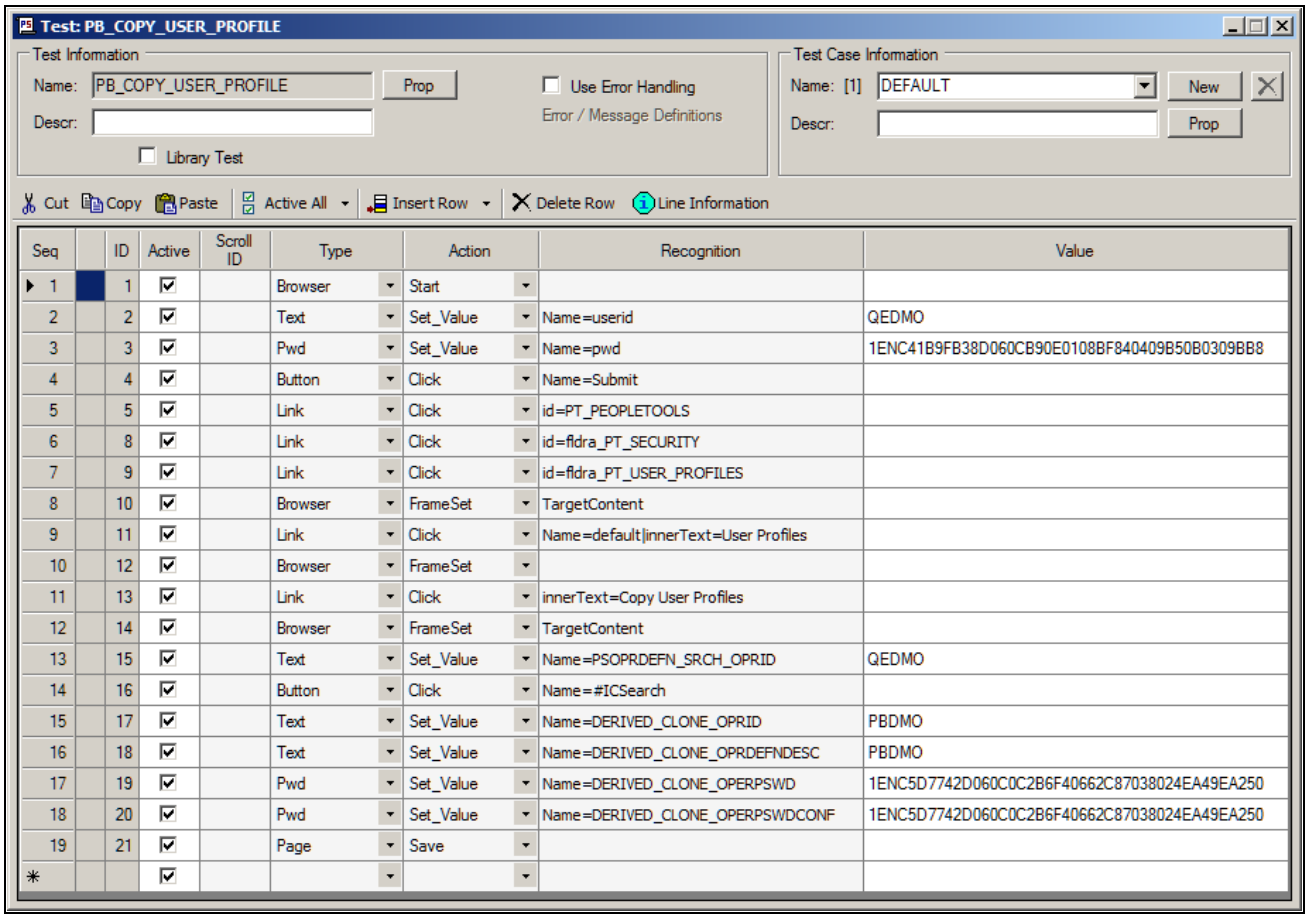
Tools Menu Command	*Usage*
Message	Opens the Message tool, which enables you to monitor test execution. The Message tool displays details about the current step, including name, object type, and value.
Log Manager	Opens the Log Manager tool, which enables you to delete logs from tests.

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 a PTF Test Editor page:



Seq		ID	Active	Scroll ID	Type	Action	Recognition	Value
1		1	☑		Browser	Start		
2		2	☑		Text	Set_Value	Name=userid	QEDMO
3		3	☑		Pwd	Set_Value	Name=pwd	1ENC41B9FB38D060CB90E0108BF840409B50B0309BB8
4		4	☑		Button	Click	Name=Submit	
5		5	☑		Link	Click	id=PT_PEOPLETOOLS	
6		8	☑		Link	Click	id=fldra_PT_SECURITY	
7		9	☑		Link	Click	id=fldra_PT_USER_PROFILES	
8		10	☑		Browser	FrameSet	TargetContent	
9		11	☑		Link	Click	Name=default|innerText=User Profiles	
10		12	☑		Browser	FrameSet		
11		13	☑		Link	Click	innerText=Copy User Profiles	
12		14	☑		Browser	FrameSet	TargetContent	
13		15	☑		Text	Set_Value	Name=PSOPRDEFN_SRCH_OPRID	QEDMO
14		16	☑		Button	Click	Name=#ICSearch	
15		17	☑		Text	Set_Value	Name=DERIVED_CLONE_OPRID	PBDMO
16		18	☑		Text	Set_Value	Name=DERIVED_CLONE_OPRDEFNDESC	PBDMO
17		19	☑		Pwd	Set_Value	Name=DERIVED_CLONE_OPERPSWD	1ENC5D7742D060C0C2B6F40662C87038024EA49EA250
18		20	☑		Pwd	Set_Value	Name=DERIVED_CLONE_OPERPSWDCONF	1ENC5D7742D060C0C2B6F40662C87038024EA49EA250
19		21	☑		Page	Save		
*			☑					

PTF Test Editor

This section discusses:

- Test Editor menus.
- Test Editor field.
- Test window.
- Test window fields.
- Test window toolbar.
- Test step fields.

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 PTF menu commands:

PTF Menu Command	Usage
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.
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.
Run	Runs the current test.
Pause	Pauses execution of the test.
End	Stops execution of the test.
Open Test Recorder	Launches the Test Recorder.

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

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

This table describes Debug menu commands:

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

Debug Menu Command	*Usage*
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.

This table discusses the Window menu commands:

Window Menu Command	*Usage*
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.
Again	Searches for the Find string again.

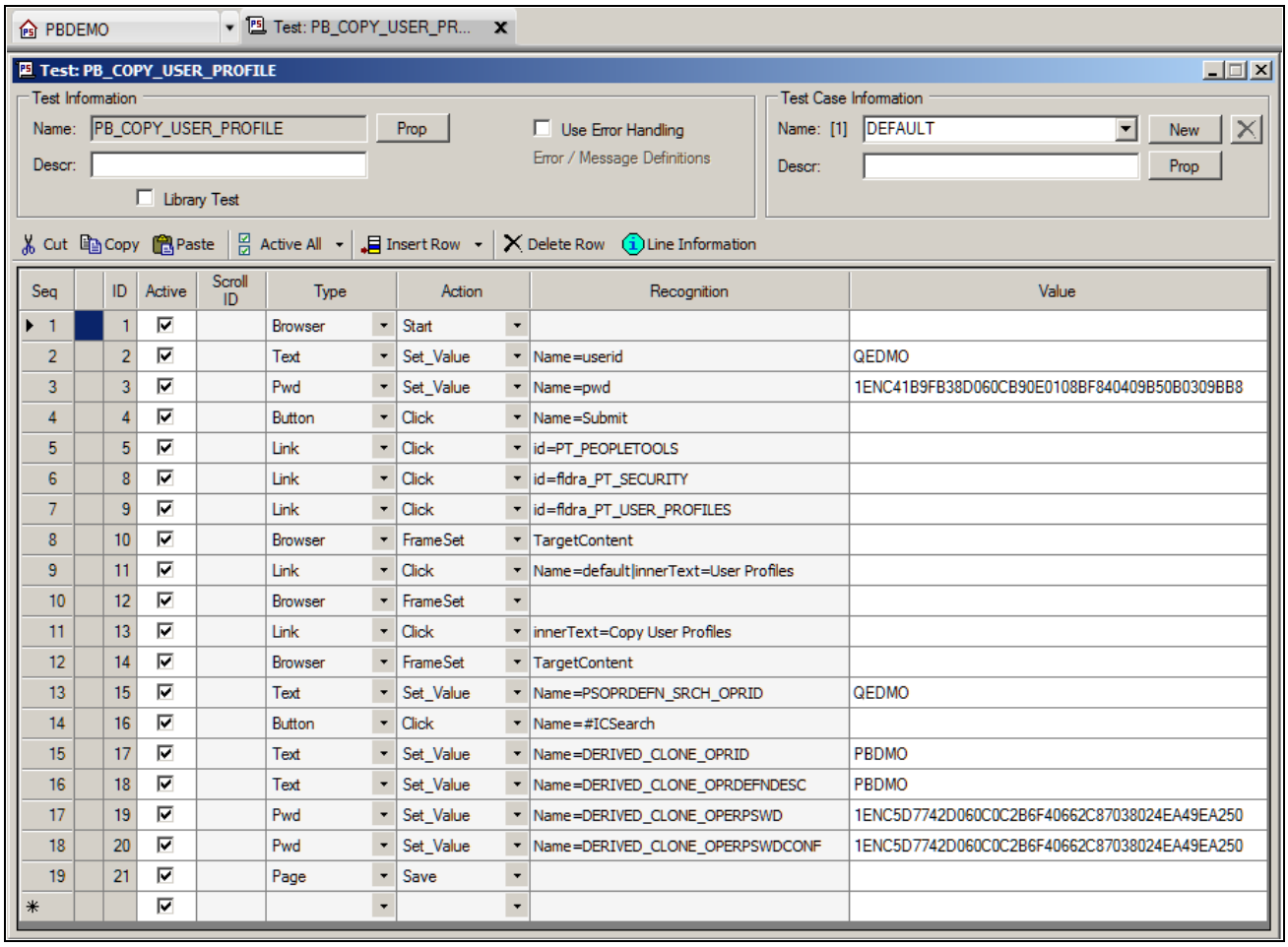
Test Editor Field

The Test Editor includes this field:

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:



Seq		ID	Active	Scroll ID	Type	Action	Recognition	Value
1		1	☑		Browser	Start		
2		2	☑		Text	Set_Value	Name=userid	QEDMO
3		3	☑		Pwd	Set_Value	Name=pwd	1ENC41B9FB38D060CB90E0108BF840409B50B0309BB8
4		4	☑		Button	Click	Name=Submit	
5		5	☑		Link	Click	id=PT_PEOPLETOOLS	
6		8	☑		Link	Click	id=fldra_PT_SECURITY	
7		9	☑		Link	Click	id=fldra_PT_USER_PROFILES	
8		10	☑		Browser	FrameSet	TargetContent	
9		11	☑		Link	Click	Name=default|innerText=User Profiles	
10		12	☑		Browser	FrameSet		
11		13	☑		Link	Click	innerText=Copy User Profiles	
12		14	☑		Browser	FrameSet	TargetContent	
13		15	☑		Text	Set_Value	Name=PSOPRDEFN_SRCH_OPRID	QEDMO
14		16	☑		Button	Click	Name=#ICSearch	
15		17	☑		Text	Set_Value	Name=DERIVED_CLONE_OPRID	PBDMO
16		18	☑		Text	Set_Value	Name=DERIVED_CLONE_OPRDEFNDESC	PBDMO
17		19	☑		Pwd	Set_Value	Name=DERIVED_CLONE_OPERPSWD	1ENC5D7742D060C0C2B6F40662C87038024EA49EA250
18		20	☑		Pwd	Set_Value	Name=DERIVED_CLONE_OPERPSWDCONF	1ENC5D7742D060C0C2B6F40662C87038024EA49EA250
19		21	☑		Page	Save		
*			☑					

Example of a Test Editor test window

Test Window Fields

The following fields appear in the test window:

Test Information

Name Displays the test name. This field is display-only.

Prop (properties) Click to access the Test Properties dialog box. You can enter a long description of the test.

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.

Library Test

If Library Test is selected, this test can only be called from another test. It cannot be run as a standalone test.

Use Error Handling

If this option is selected, PTF automatically handles all error messages defined in the Error Definitions dialog box.

If this option is deselected, the system ignores error definitions.

When the Error Handling field is selected, the Error/Message Definitions link is enabled.

Error/Message Definitions

The Error/Message Definitions link is enabled only when the Error Handling field is selected.

Click to access the Error Definitions dialog box.

Use the Error Definitions dialog box to define how PTF will respond to error messages that are encountered during execution of the test.

Test Case Information

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.

New

Click to create a new test case. The new case will have blank values in the Value column. To create a new test case populated with values from the current test case, select Test, Save Test Case As.

Delete

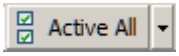
Delete the selected test case.

Prop (properties)

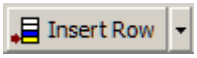
Click to access the Test Case Properties dialog box. You can enter a long description of the test case.

Test Window Toolbar

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

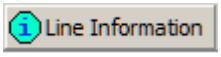


Selects or deselects the Active check box for all steps.



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

Deletes the current row.



Displays the Line Information dialog box showing the PeopleTools metadata associated with the object referenced in the step. Line Information also includes a long description where you can enter comments about the step.

Test Step Fields

A PTF test consists of a series of steps. Each step in a test is composed of eight fields, as defined in this table:

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 line (step) in a test. This value does not change when you move, add, or delete a step. Test maintenance reports use the ID value.
Active	Deselect this field to inactivate a step. PTF will skip inactive steps when the test runs. 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 type of application object the step is to take an action on or to validate. Common object 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, are *Set* and *Verify*.
Recognition	The means that PTF uses to identify the object within the application. Commonly, this is the HTML ID property.
Value	In a typical recorded step, this is the value the tester entered for an object. In a step recorded in field check mode, this would be the value that was present in the object when it was checked. Value is part of the test case, not the test itself.

See Also

Chapter 9, "Using the PTF Test Language," PTF Test Language, page 94

Using the PTF Recorder

You use the PTF 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.

This section discusses the Recorder toolbar.

Recorder Toolbar

Access the Recorder toolbar (select Test, Open Recorder or click the Show Test Recorder button).

Recorder toolbar

Note. To move the Recorder toolbar, click in the region to the left of the close icon and drag to the new location.

The Recorder toolbar provides the following functions:

Hooks 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.

Launches a PeopleSoft application in a browser window using the URL of the default execution option and hooks the Recorder to that browser.

See Chapter 2, "Installing and Configuring PTF," Configuring Execution Options, page 20.

Begins recording. The recorder adds or insert steps following the selected step.

Stops recording.

Enables Field Check Mode. Field check mode displays field values in the Recorder toolbar during recording.

You can drag and drop the Field Check Mode icon to a field in the application during recording to insert a step with a Verify or Exists action. The step is automatically populated with the object ID and value of the field.

When you drag the Field Check Mode icon from the record tool bar, your mouse image changes to a bold question mark until you select a field to check. The field that you intend to check should be highlighted when you drag the Field Check Mode icon over it. Some HTML objects, such as labels, can not be verified. If the field is not highlighted, it will not be verified by PTF.

See Chapter 10, "Test Language Reference," Verify, page 134 and Chapter 10, "Test Language Reference," Exists, page 130.

Opens a MouseOver text popu page enables you to take an action on it, such a clicking a link.

See Chapter 10, "Test Language Reference," MouseOver, page 125.

Copies the recording to the clipboard.

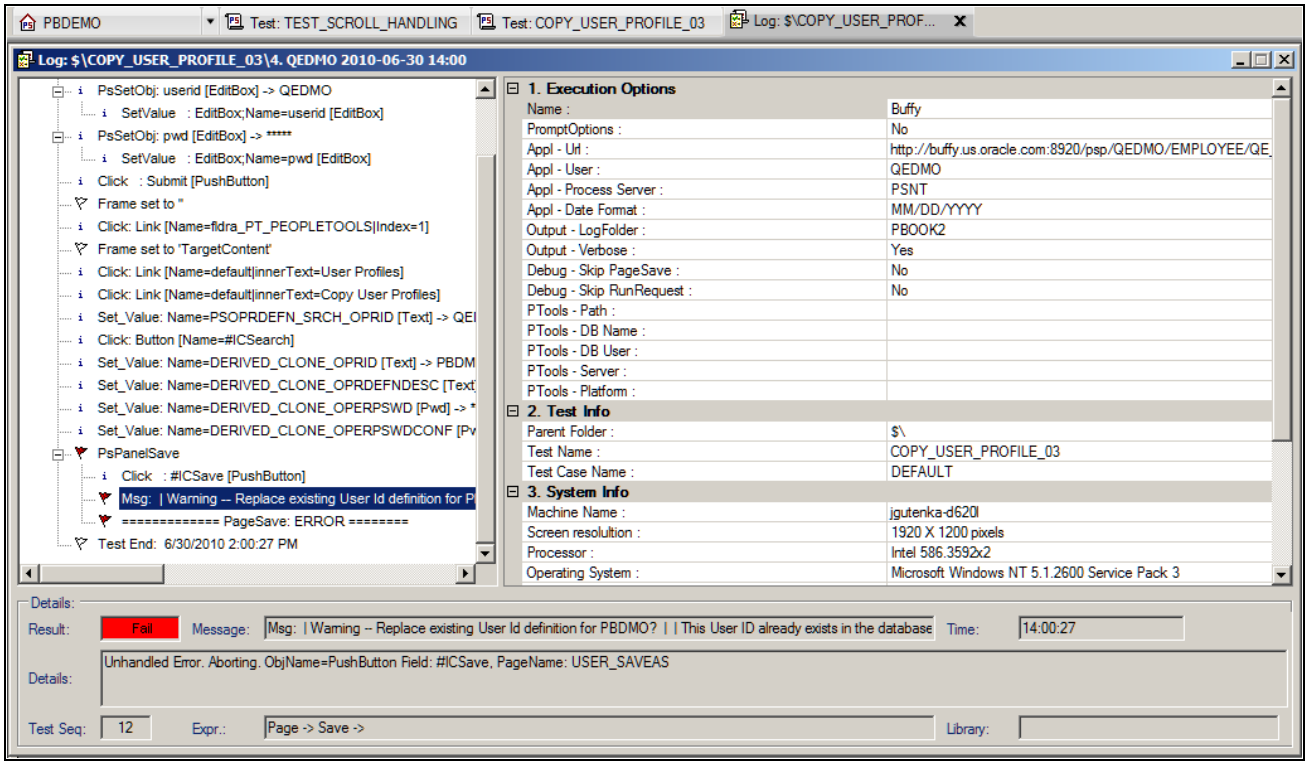
Using the Log Viewer

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:



Example of the Log Viewer

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*
Expand All	Expands all the selections in the log.
Collapse All	Collapses all the selections in the log.
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.

This table describes the Detail menu commands:

Detail Menu Command	*Usage*
Goto Script Line	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.

You can double-click a log entry or right-click and select Open to access the test with the corresponding step highlighted.

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.

See Also

Chapter 5, "Developing and Debugging Tests," Interpreting Logs, page 56

Chapter 4

Creating Tests and Test Cases

This chapter discusses how to:

- Create tests.
- Record tests.
- Create test cases.
- Execute tests.
- Review test logs.

Creating Tests

This section discusses how to:

- Create a new folder.
- Create a new test.
- Name tests.
- Copy a test.

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.

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 Prop button 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 often short and provide little information.
5. Select Test, Save or click the Save button in the toolbar.
6. Enter a name for the new test.
7. Click OK.

Naming Tests

These rules apply to test names:

- 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 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.

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
- Scroll ID is populated manually by the developer.
- Type
- Action
- Recognition
- Value

This is an example of test steps:

Seq		ID	Active	Scroll ID	Type	Action	Recognition	Value
1		19	☑		Browser	Start		
2		3	☑		Text	Set_Value	Name=userid	QEDMO
3		4	☑		Pwd	Set_Value	Name=pwd	1ENC41B9FB38D060CB90E0108BF840409B50B0309BB8
4		5	☑		Button	Click	Name=Submit	
5		6	☑		Link	Click	id=fldra_PT_PEOPLETOOLS	
6		7	☑		Link	Click	id=fldra_PT_SECURITY	
7		8	☑		Link	Click	id=fldra_PT_USER_PROFILES	
8		9	☑		Link	Click	innerText=Copy User Profiles	
9		10	☑		Browser	FrameSet	TargetContent	
10		11	☑		Text	Set_Value	Name=PSOPRDEFN_SRCH_OPRID	
11		12	☑		Text	Set_Value	Name=PSOPRDEFN_SRCH_OPRID	QEDMO
12		13	☑		Button	Click	Name=#ICSearch	
13		14	☑		Text	Set_Value	Name=DERIVED_CLONE_OPRID	QEDMO2
14		15	☑		Text	Set_Value	Name=DERIVED_CLONE_OPRDEFNDESC	QEDMO2
15		16	☑		Pwd	Set_Value	Name=DERIVED_CLONE_OPERPSWD	1ENC8BD343608000C516A4AED538C0B8D5CEE4F64500
16		17	☑		Pwd	Set_Value	Name=DERIVED_CLONE_OPERPSWDCONF	1ENC8BD343608000C516A4AED538C0B8D5CEE4F64500
17		18	☑		Page	Save		
▸*			☑					

Example of test steps

Seq and ID are system-generated fields.

Scroll ID is populated by the developer only for scroll actions.

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

For a complete description of test steps, see "Test Step Fields."

See Chapter 3, "Using PeopleSoft Test Framework," Test Step Fields, page 34.

See Also

Chapter 9, "Using the PTF Test Language," PTF Test Language, page 94

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 begins at Step 2. Step 1 is always Browser.Start or Browser.Start_Login. For an explanation of the difference between Start and Start_Login, see "PTF Test Language Reference."

   See Chapter 10, "Test Language Reference," Browser, page 97.
3. With a test open, select Test, Open Test Recorder or click the Show Test Recorder button in the toolbar.
4. The Recorder toolbar appears.

   See Chapter 3, "Using PeopleSoft Test Framework," Using the PTF Recorder, page 34.
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 button on the recorder toolbar. This will start the web browser with the default test application URL.
   - 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 originally launched by PTF.
6. Click the Start Recording button in the Recorder toolbar to begin recording.
7. Perform the test steps in the PeopleSoft application.
8. Click the Stop Recording button in the Recorder toolbar to end the recording.
9. Save the test.

Creating 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

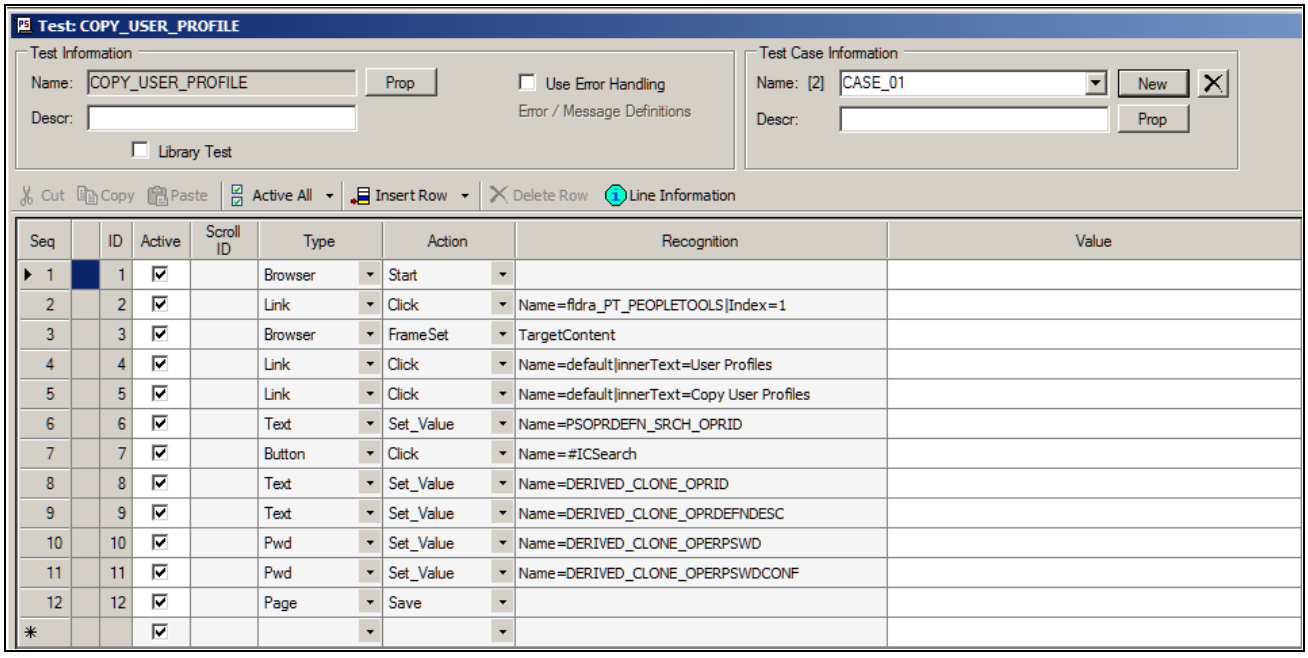
To create a new test case:

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 Prop button 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 often short and provide the user with little information.
5. 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.
6. Save the test case.

This example shows a new test case with blank values:



Seq		ID	Active	Scroll ID	Type	Action	Recognition	Value
1		1	☑		Browser	Start		
2		2	☑		Link	Click	Name=fldra_PT_PEOPLETOOLS|Index=1	
3		3	☑		Browser	FrameSet	TargetContent	
4		4	☑		Link	Click	Name=default|innerText=User Profiles	
5		5	☑		Link	Click	Name=default|innerText=Copy User Profiles	
6		6	☑		Text	Set_Value	Name=PSOPRDEFN_SRCH_OPRID	
7		7	☑		Button	Click	Name=#ICSearch	
8		8	☑		Text	Set_Value	Name=DERIVED_CLONE_OPRID	
9		9	☑		Text	Set_Value	Name=DERIVED_CLONE_OPRDEFNDESC	
10		10	☑		Pwd	Set_Value	Name=DERIVED_CLONE_OPERPSWD	
11		11	☑		Pwd	Set_Value	Name=DERIVED_CLONE_OPERPSWDCONF	
12		12	☑		Page	Save		
*			☑					

New test case with blank values

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.

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.

See Also

Chapter 4, "Creating Tests and Test Cases," Reviewing Test Logs, page 45

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.

See Also

Chapter 4, "Creating Tests and Test Cases," Reviewing Test Logs, page 45

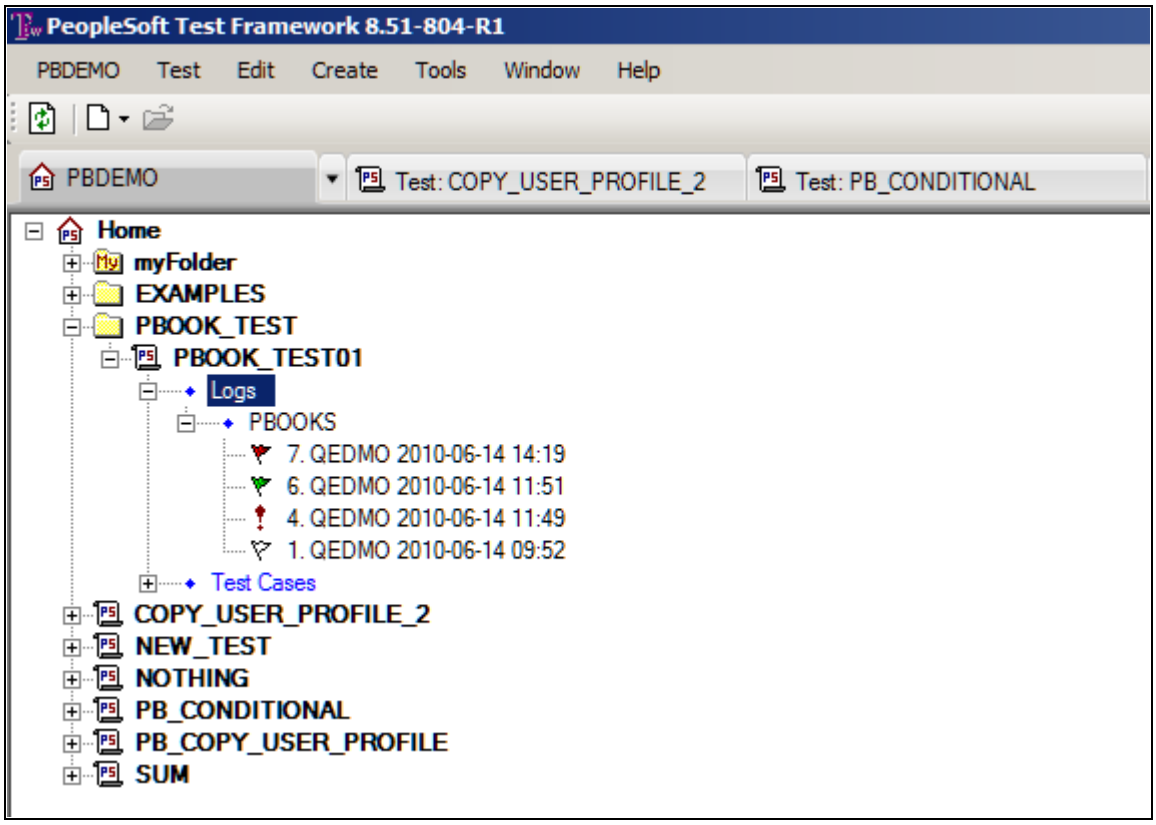
Reviewing Test Logs

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.

This example shows log entries in the Logs folder of PTF Explorer:



Example of PTF Explorer showing log entries in the Logs folder

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

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

This example shows log entries in a test log:

```
Test Start:  6/30/2010 2:00:00 PM
    PsGotoURL
    Doing login in English
    PsSetObj: userid [EditBox] -> QEDMO
        SetValue   : EditBox;Name=userid [EditBox]
    PsSetObj: pwd [EditBox] -> *****
        SetValue   : EditBox;Name=pwd [EditBox]
    Click   : Submit [PushButton]
    Frame set to ''
    Click: Link [Name=fldra_PT_PEOPLETOOLS|Index=1]
    Frame set to 'TargetContent'
    Click: Link [Name=default|innerText=User Profiles]
    Click: Link [Name=default|innerText=Copy User Profiles]
    Set_Value: Name=PSOPRDEFN_SRCH_OPRID [Text] -> QEDMO
    Click: Button [Name=#ICSearch]
    Set_Value: Name=DERIVED_CLONE_OPRID [Text] -> PBDMO
    Set_Value: Name=DERIVED_CLONE_OPRDEFNDESC [Text] -> PBDMO
    Set_Value: Name=DERIVED_CLONE_OPERPSWD [Pwd] -> *******
    Set_Value: Name=DERIVED_CLONE_OPERPSWDCONF [Pwd] -> *******
    PsPanelSave
        Click   : #ICSave [PushButton]
        Msg:  | Warning -- Replace existing User Id definition for PBDMO?  |  | This User ID already
        ============= PageSave: ERROR ========
    Test End:  6/30/2010 2:00:27 PM
```

Example of log entries in a test log

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:

i or Info	Information message only.
or Info	Information message only.
or Pass	Step was successful.
or Warning	Step was successful, but with a warning.
or Fail	Step failed.
! or Script Fail	The test encountered a condition that it was not configured to handle.

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.

See Also

Chapter 5, "Developing and Debugging Tests," Interpreting Logs, page 56

Chapter 5

Developing and Debugging Tests

After you finish recording a test, you will likely need to make some changes. You may have made an error entering a value or skipped a step. You may have recorded extra steps. You will probably need to manually add steps that cannot be recorded.

The topics in this chapter describe tools and techniques you can use to modify your tests after recording them.

This chapter discusses how to:

- Use the Message Tool.
- Use reserved words.
- Use variables.
- Use conditional logic.
- Handle errors.
- Interpret logs.
- Incorporate scroll handling.
- Call tests.

Using the Message Tool

This section discusses how to use the Message tool.

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

Select Help, 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.

This example shows the Message tool ID values:

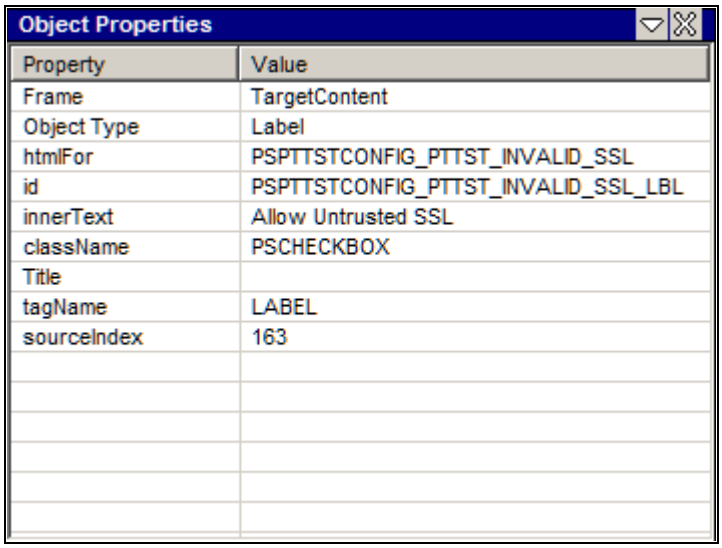
Message tool ID values

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.

HTML Browser

To view additional details about a browser object, access the Message tool and select Menu, HTML Browser, Show. This example shows the HTML Browser Object Properties window:



Object Properties

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

Example of the HTML Browser

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

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:

Text	Set_Value	Name=PA_PROP_VOUCH_CREAT_DT	06/30/2010

Example of a test step with static data

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 dynamic, replace the recorded data with the **#TODAY** reserved word, which represents the date at the moment of test execution, as shown in this example:

Text	Set_Value	Name=PA_PROP_VOUCH_CREAT_DT	#TODAY

Example of 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:

Text	Set_Value	PA_CALCULATION_DESCR	Sample pension calc

Example of a step that sets a value

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:

Text	Set_Value	PA_CALCULATION_DESCR	#CHECK#Sample pension calc

Example of a step that verifies a value

See Also

Chapter 4, "Creating Tests and Test Cases," Creating Test Cases, page 42

Chapter 10, "Test Language Reference," Reserved Words, page 135

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 prefixed by an ampersand (&).

Store a value for a variable either by placing the `ret=&varname` parameter in the ID 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 ID field of a Conditional.If_Then step or 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.

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:

Text	Get_Property	ID=GM_PROP_ID;prop=Value;ret=&propID	
Link	Click	innerText=V101	
Span	Verify	ID=GM_PROP_ID	&propID

Example of using a variable

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

See Also

Chapter 9, "Using the PTF Test Language," Parameters, page 95

Chapter 10, "Test Language Reference," Variable, page 128

Chapter 10, "Test Language Reference," Conditional, page 103

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 statement is true, then PTF executes the subsequent steps until it encounters the End_If step. If the statement is false, then flow skips to the step following the End_If step.

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

```
12. 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	Click	Name=DERIVED_IC_GBL_BRA$img	

Example of step that requires a conditional construct

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	Get_Property	Name=DERIVED_IC_GBL_BRA$img; prop=alt; ret=&flagstate	
Conditional	If_Then	&flagstate=Expand section Brazil	
Image	Click	Name=DERIVED_IC_GBL_BRA$img	
Conditional	End_If		

Example of using conditional logic

See Also

Chapter 10, "Test Language Reference," Conditional, page 103

Handling Errors

This section discusses how to handle errors.

Use the Error/Message Definitions 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 (16000,494)
```

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

CheckBox	Set_Value	Name=PA_CALCULATION_CALC_ALL_PLANS_cd$0	N
Button	Click	Name=#ICOK	

Example of steps that trigger and dismiss a warning

If some test cases belonging to this test do not deselect the Calculate all Plans check box in the first step, then the step would not trigger the warning message and PTF would fail to find the OK button in the next 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 Error/Message Definitions dialog box.

1. With a test open, select the Use Error Handling check box.

   When the Use Error Handling check box is selected, the Error/Message Definitions link is enabled.

   If the Use Error Handling check box is deselected, PTF will not check for message definitions.

2. Click the Error/Message Definitions link.

3. Enter the text of the message, or a portion of the text, in the Message field.

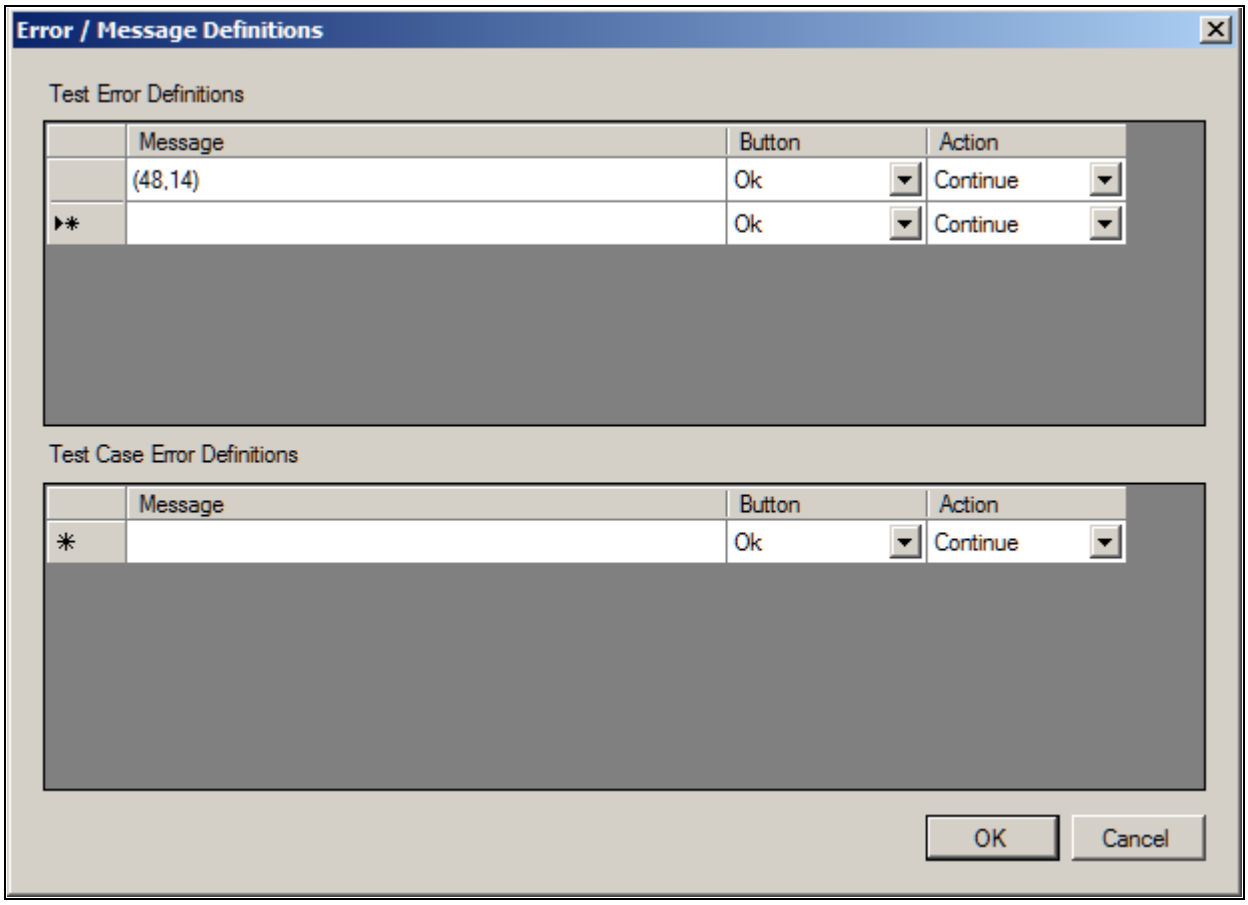
   The following example uses the message catalog number rather than the full message text.

   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.

4. Select which button the step should click.

5. Specify what action PTF should take after the button is clicked.

This example shows the Error/Message Definitions dialog box:



Error/Message Definitions dialog box

This table lists the names and definitions of the elements on the Error/Message Definitions 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 Chapter 2, "Installing and Configuring PTF," Configuring the Browser Security Settings, page 11.
Fail	Object not found in the page, or access is denied to the Frame.	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. Check browser settings to make sure your browser is configured properly. See Chapter 2, "Installing and Configuring PTF," Configuring the Browser Security Settings, page 11. If this log message appears sporadically throughout your log, it could mean that objects within your application have been removed or renamed. • 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 ID field of the step. See Chapter 5, "Developing and Debugging Tests," Using the Message Tool, page 49. • 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. See Chapter 7, "Identifying Change Impacts," page 67.

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.

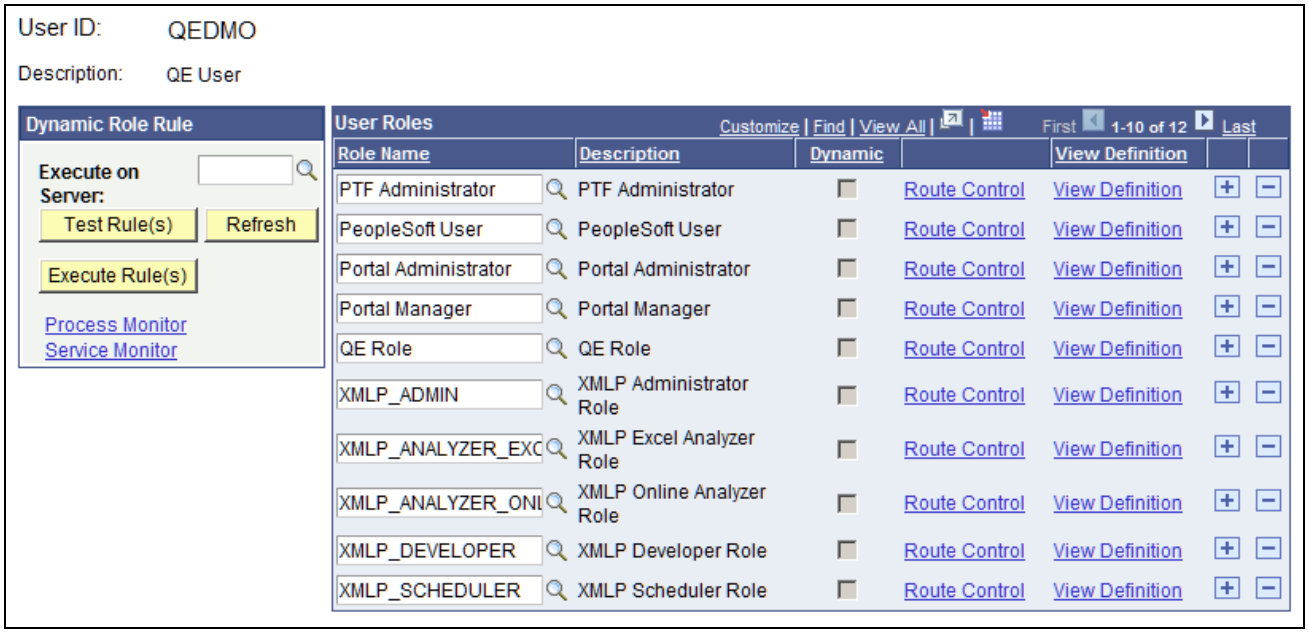
A scroll can be implemented as a scroll area or a grid. In scroll areas, the fields appear on the page in a freeform manner. In grids, fields appear as columns similar to those on a spreadsheet. Individual rows of data within a scroll or grid are uniquely identified by a set of one or more fields, or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



Example of the User Profile - Roles page

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

Seq		ID	Active	Scroll ID	Type	Action	Recognition	Value
1		22	☑		Text	Verify	Name=PSROLEUSER_VW_ROLENAME$0	PTF Administrator

Test step to verify a field on a scroll area

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 repeatably 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 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.

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.

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.

Here is an example of the Message tool with recognition data for the Role Name field:

Example of the Message tool

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:

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

Example of a step that verifies a field on a row

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 Recognition 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.

Here is an example of a Scroll.Key_Set step that sets the key to PTF Administrator:

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

Example of a Scroll.Key_Set step

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 index value.

Here is an example of an Action step:

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

Example of an Action step

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:

	CheckBox	Verify	Name=PSROLEUSER_VW_DYNAMIC_SW$0	N

Example of step using positional reference

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:

1	Scroll	Key_Set	Type=Text;Name=PSROLEUSER_VW_ROLENAME	PTF Administrator
1	Scroll	Action	ret=&Scroll1	sel
	CheckBox	Verify	Name=PSROLEUSER_VW_DYNAMIC_SW&Scroll1	N

Example of a step using an index variable

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.

Scroll ID	Type	Action	Recognition	Value
	Page	Go_To	Roles	
1	Scroll	Key_Set	Type=Text;Name=PSROLEUSER_VW_ROLENAME	PTF Administrator
1	Scroll	Action	ret=&Scroll1	not
	Log	Message	Scroll1 index variable = &Scroll1	
1	Scroll	Key_Set	Type=Text;Name=PSROLEUSER_VW_ROLENAME	PTF User
1	Scroll	Action	ret=&Scroll1	sel

Example of assigning Scroll IDs to Scroll steps

See the "PTF Language Reference" chapter for a complete description of Scroll actions and additional examples

See Also

Chapter 10, "Test Language Reference," Scroll, page 121

Calling Tests

This section provides an overview of calling tests and discusses how to:

- Use library tests.
- Use shell 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 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:

- Test.Exec - calling other tests
- DataMover.Exec - calling data mover scripts
- Query.Exec - running queries
- 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. Send the text 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

This chapter discusses how to:

- Manage PeopleSoft Test Framework (PTF) logs.
- Migrate PTF tests.

Managing PTF Logs

This section provides an overview of PTF Log Manager and discusses how to:

- Use Log Manager fields.
- Use Log Manager buttons.
- Use the Selection pane.
- Use the Trace pane.

Understanding Log Manager

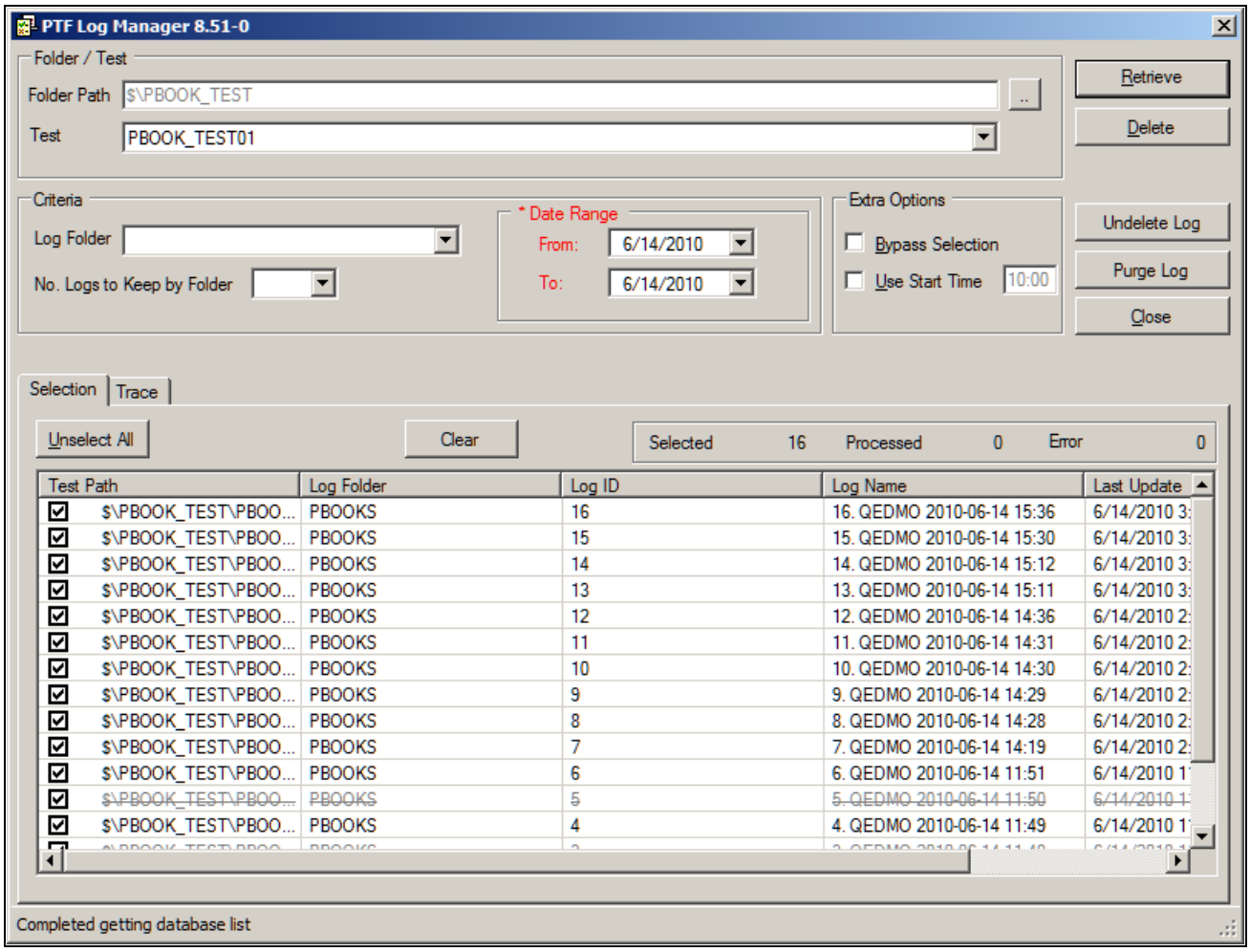
Over time, as you run tests, you will create a number of test logs. Because they reside in your application database, test logs, especially those containing many screen shots, can affect the storage demands on your database. PTF Log Manager enables you to minimize this demand and remove clutter from your database.

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 Select Help, Tools, Log Manager.

Note. Only an administrator (a user ID with the PTF Administrator role) is able to open Log Manager.

This example shows the PTF Log Manager:



Example of PTF 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 log files 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 remain in the folder. For instance, if the folder contains six logs and you specify three, then the three newest logs will be retained and the three oldest logs will be retrieved.
Date Range	The system retrieves only logs created within the date range.

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 Log Manager Buttons

Log Manager has these buttons:

Retrieve	Select to populate the selection pane based on the specified criteria.
Delete	Select to delete the selected logs.
Undelete Log	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 Log	Select to remove selected logs from the database if they are marked for deletion.

Using the Selection Pane

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 Log, or Purge Log.

Using the Trace Pane

The trace pane displays a history of processing actions for this session.

Migrating PTF Tests

Because PTF test assets 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 Application Designer that includes tests and test cases and export the project to another database using the Copy Project tool.

You can also include tests and test cases in an upgrade project.

See Also

PeopleTools 8.51 PeopleBook: PeopleSoft Application Designer Developer's Guide, "Working With Projects"

Chapter 7

Identifying Change Impacts

This chapter provides an overview of change impacts and discusses how to:

- Define analysis rules.
- Create test maintenance reports.
- Interpret test maintenance reports.
- Create test coverage reports.
- Interpret test coverage reports.
- Query PTF report tables.

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, 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 enable test developers to help in determining 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 components, menus, 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 components, menus, pages, records, and fields that are referenced in PTF tests.

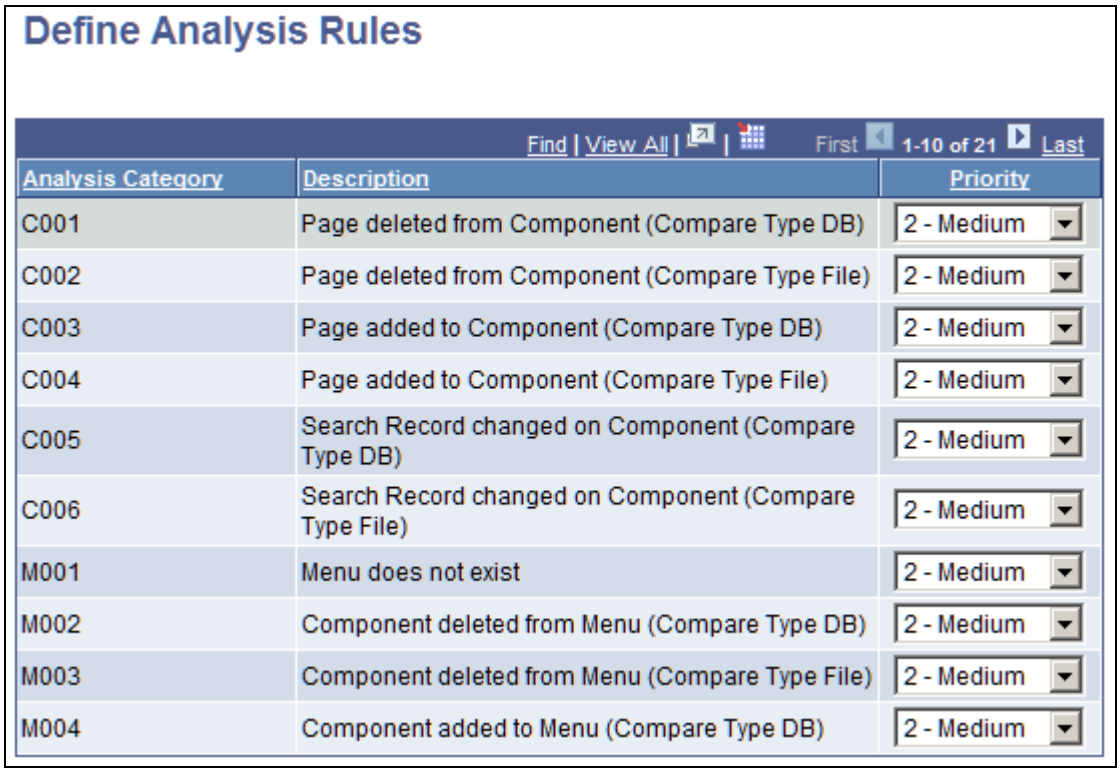
  When used in conjunction with Usage Monitor, test coverage correlations can be extended to include information on all managed objects.

  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 how to define analysis rules.

Access the Define Analysis Rules page (PeopleTools, Lifecycle Tools, Test Framework, Define Analysis Rules).



Define Analysis Rules

Find | View All | First 1-10 of 21 Last

Analysis Category	Description	Priority
C001	Page deleted from Component (Compare Type DB)	2 - Medium
C002	Page deleted from Component (Compare Type File)	2 - Medium
C003	Page added to Component (Compare Type DB)	2 - Medium
C004	Page added to Component (Compare Type File)	2 - Medium
C005	Search Record changed on Component (Compare Type DB)	2 - Medium
C006	Search Record changed on Component (Compare Type File)	2 - Medium
M001	Menu does not exist	2 - Medium
M002	Component deleted from Menu (Compare Type DB)	2 - Medium
M003	Component deleted from Menu (Compare Type File)	2 - Medium
M004	Component added to Menu (Compare Type DB)	2 - Medium

Define Analysis Rules page

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 analysed.

The following table describes the analysis categories:

Analysis Category	***Description***
C001	Page Deleted from Component (Compare Type DB)
C002	Page Deleted from Component (Compare Type File)
C003	Page Deleted from Component (Compare Type DB)
C004	Page Added to Component (Compare Type File)
C005	Search Record changed on Component (Compare Type DB)
C006	Search Record changed on Component (Compare Type File)
M001	Menu Deleted from Database
M002	Component Deleted from Menu (Compare Type DB)
M003	Component Deleted from Menu (Compare Type File)
M004	Component Added to Menu (Compare Type DB)
M005	Component Added to Menu (Compare Type File)
M01D	Menu does not exist (Compare Type DB)
P001	Field deleted on Page
P002	Field added to Page (Compare Type DB)
P003	Field added to Page (Compare Type File)
P004	Fieldname changed on Page
P005	Field Type changed on Page
P006	Field Label changed on Page
P01D	Field deleted on Page (Compare Type DB)
P03D	Fieldname changed on Page (Compare Type DB)

Analysis Category	*Description*
P04D	Field Type changed on Page (Compare Type DB)
P05D	Field Label changed on Page (Compare Type DB)
R001	RecordField now required on Page (Compare Type DB)
R002	RecordField now required on Page (Compare Type File)
R003	RecordField no longer required on Page (Compare Type DB)
R004	RecordField is now a Search Key (Compare Type DB)
R03D	RecordField no longer required on Page (Compare Type File)
R03F	RecordField is now a Search Key (Compare Type File)
R04D	RecordField no longer a Search Key (Compare Type DB)
R04F	RecordField no longer a Search Key (Compare Type File)
R05D	RecordField is now a List Box Item (Compare Type DB)
R05F	Record Field is now a List Box Item (Compare Type DB)
R06D	RecordField no longer a List Box Item (Compare Type DB)
R06F	RecordField no longer a List Box Item (Compare Type File)

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 three steps:

- Step 1: Manual Steps
- Step 2: Analyze Compare Data
- Step 3: Generate Report

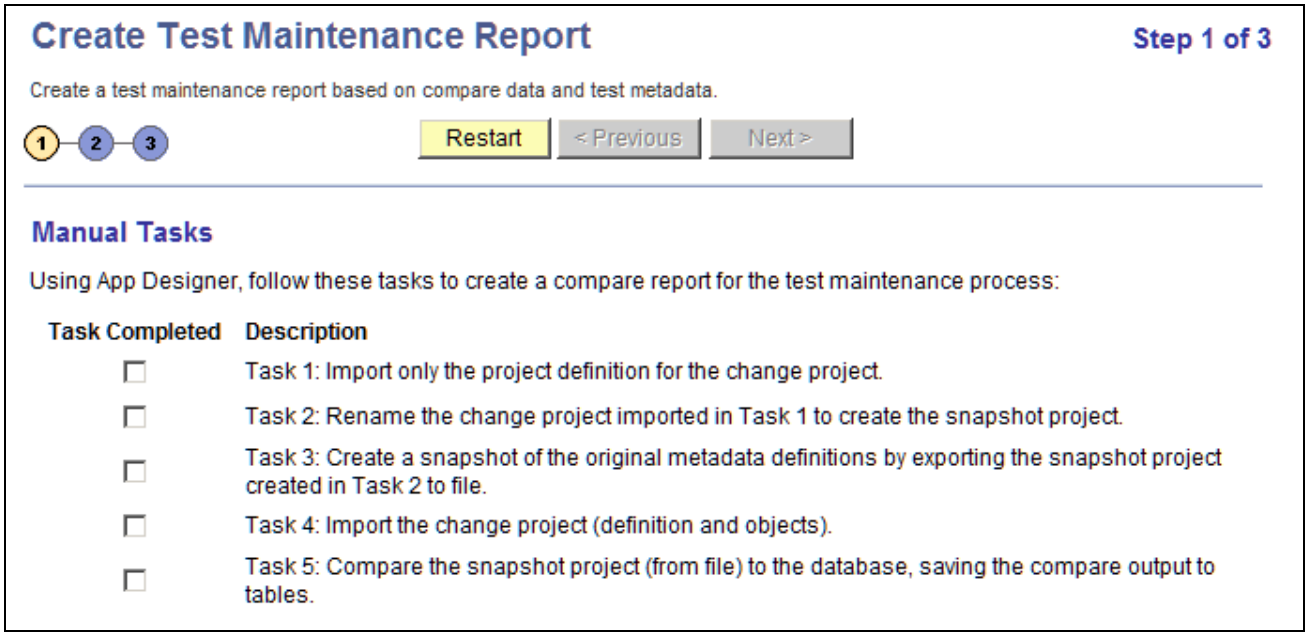
Step 1 of 3: Manual Tasks

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).



Create Maintenace Report Wizard: Step 1 of 3

As you progress through the Test Maintenance wizard the wizard tracks which tasks you have completed and which page you are on. When 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 may 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 different 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. 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.

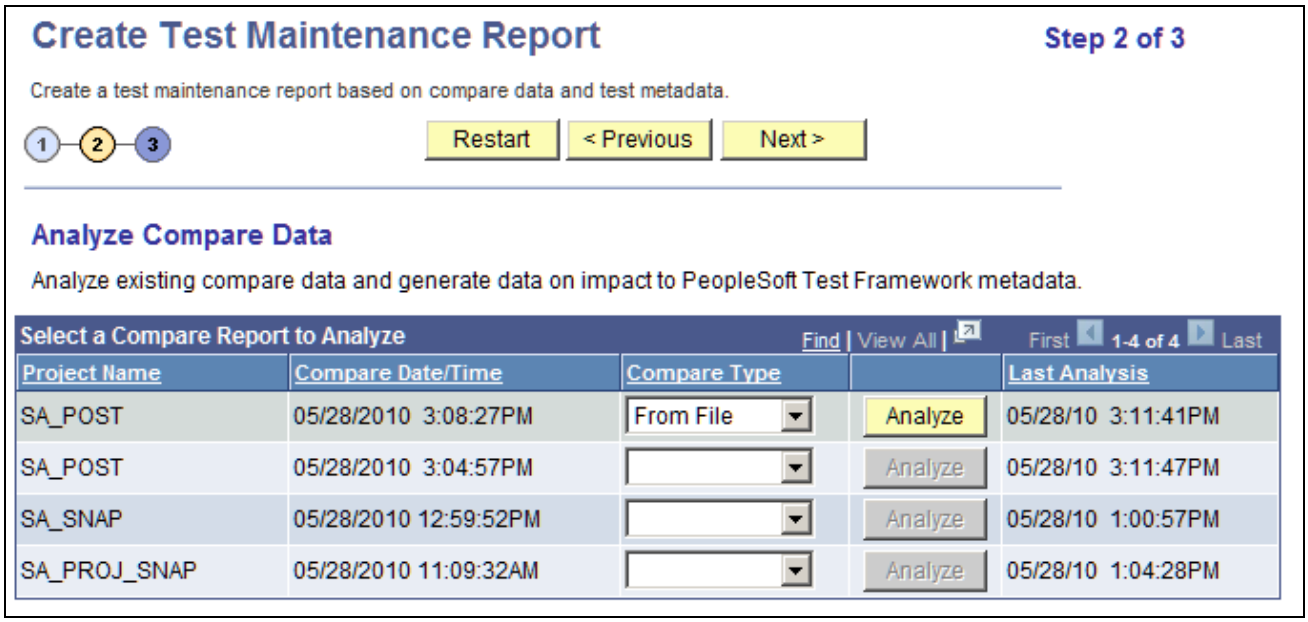
   The Replace existing SA_PROJ_SNAP? dialog box appears.
3. Click Yes.

   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 3: Analyze Compare Data

This example shows Step 2 of 3:



Create Test Maintenance Report Wizard: Step 2 of 3

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.

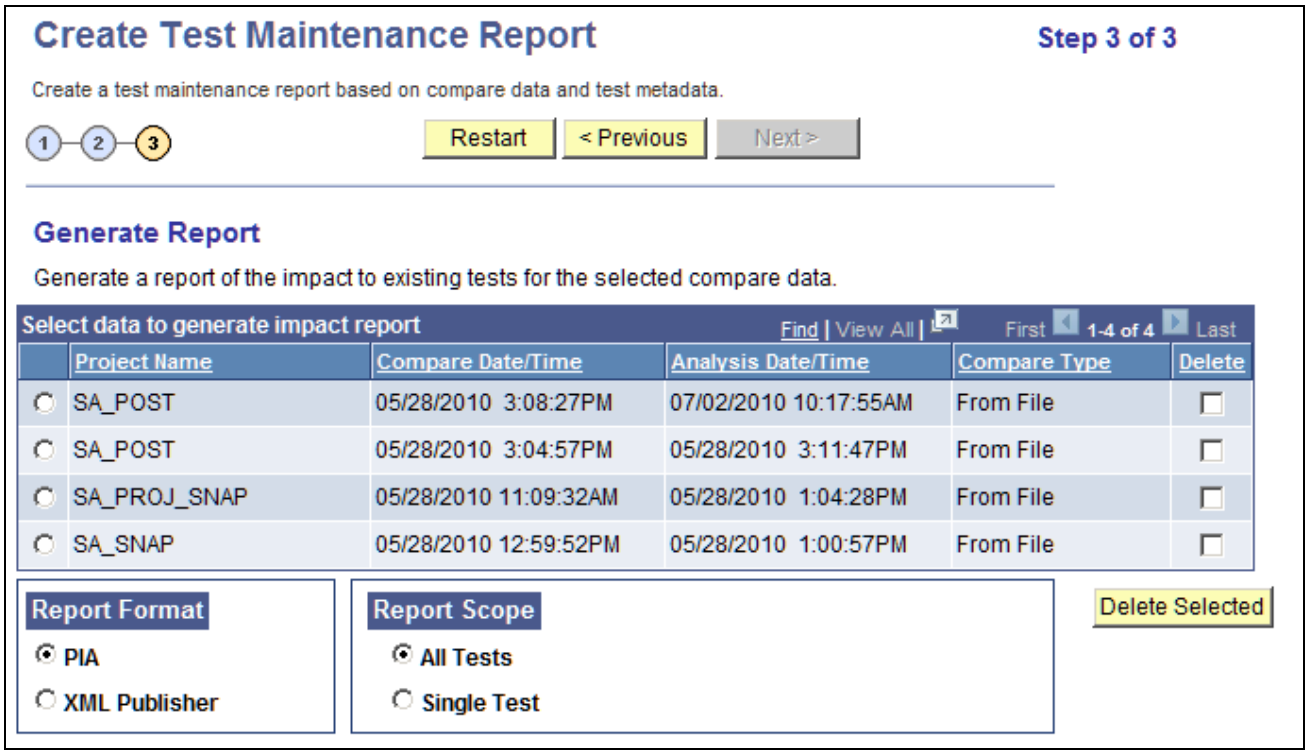
   This will launch the Application Engine program PSPTANALYSIS.

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 3: Generate Report

This example shows Step 3 of 3:



Create Test Maintenance Report Wizard: Step 3 of 3

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 Step 2.

Follow these steps to generate a report:

1. 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.

2. Select the report format.

3. Select the report scope.

4. Click Generate Report to have PTF create the test maintenance report.

Select the Delete check box and click Delete Selected to delete analyses.

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 XML Publisher (XMLP).

The following example shows a report in XMLP format:

Test Maintenance Report
Project Name SA_PROJ_SNAP

Test Name: INSTRUCTORS			
	2 - Medium	P001 Pages	PSU_INSTR — Deleted **Command ID:** 32 **Seq. Nbr:** 32 **Status:** Active **Language:** EN **Pre Object Value:** PSU_INSTR_TBL.MANAGER **Post Object Value:**
	2 - Medium	P003 Pages	PSU_INSTR — Added **Command ID:** 0 **Seq. Nbr:** 0 **Status:** Active **Language:** EN **Pre Object Value:** **Post Object Value:** PSU_INSTR_TBL.CURR_DEV_FLAG
	2 - Medium	P003 Pages	PSU_INSTR — Added **Command ID:** 0 **Seq. Nbr:** 0 **Status:** Active **Language:** **Pre Object Value:** **Post Object Value:** PSU_INSTR_TBL.CURR_DEV_FLAG
	2 - Medium	P006 Pages	PSU_INSTR — Changed **Command 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
	2 - Medium	P006 Pages	PSU_INSTR — Changed **Command ID:** 32 **Seq. Nbr:** 32 **Status:** Active **Language:** EN **Pre Object Value:** PSU_INSTR_TBL.MANAGER.Manager **Post Object Value:** PSU_INSTR_TBL.MANAGER
Test Name: STUDENTS			
	2 - Medium	P003 Pages	PSU_STUDENT_PERS — Added **Command ID:** 0 **Seq. Nbr:** 0 **Status:** Active **Language:** EN **Pre Object Value:** **Post Object Value:** PSU_STUDENT_TBL.PHONE
Test Name: TRN_STUDENT_04			
	2 - Medium	P003 Pages	PSU_STUDENT_PERS — Added **Command ID:** 0 **Seq. Nbr:** 0 **Status:** Active **Language:** EN **Pre Object Value:** **Post Object Value:** PSU_STUDENT_TBL.PHONE

Example of a Test Maintenance report in XMLP format

The following example shows a 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 — Find | View All | First 1-7 of 7

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	P006	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	P006	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)

Example of a Test Maintenance report in PIA format

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:

ps.xls [Read-Only]

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Test Name	Priority	Category	Object Type	Object Name	Market	Command ID	Seq Nbr	Status	Object Action	Pre Object Value	Post Object Value	Description
2	INSTRUCTORS	2 - Medium	P001	Pages	PSU_INSTR		32	32	Active	Deleted	MANAGER		Field deleted on Page
3	INSTRUCTORS	2 - Medium	P003	Pages	PSU_INSTR		0	0	Active	Added		RR_DEV_FLAG	Field added to Page (Compare Type File)
4	INSTRUCTORS	2 - Medium	P003	Pages	PSU_INSTR		0	0	Active	Added		RR_DEV_FLAG	Field added to Page (Compare Type File)
5	INSTRUCTORS	2 - Medium	P006	Pages	PSU_INSTR		11	11	Active	Changed	RST_NAME.First	RST_NAME.First	Field Label changed on Page
6	INSTRUCTORS	2 - Medium	P006	Pages	PSU_INSTR		32	32	Active	Changed	MANAGER.Manag	ANAGER	Field Label changed on Page
7	STUDENTS	2 - Medium	P003	Pages	PSU_STUDENT_PERS		0	0	Active	Added		.PHONE	Field added to Page (Compare Type File)
8	TRN_STUDENT_04	2 - Medium	P003	Pages	PSU_STUDENT_PERS		0	0	Active	Added		.PHONE	Field added to Page (Compare Type File)

Example of a Test Maintenance report in spreadsheet format

The following example shows a Test Maintenance report in the PTF client:

Maintenance Report

Test Name: PAGE_SAVE

Project Name: JY_TESTCOMPARE

Analysis Seq: 2

Run DateTime: 5/26/2010 3:39:43 PM

DateTime Stamp: 5/27/2010 1:59:21 PM

AnalysisCategory	Priority	CommandID	LanguageCode	SeqNbr	CommandStatus	ObjectType	ObjectDescription	ObjectNan
P006	1	0	ENG	0	A	5	Pages	USER_SA
P004	2	0	ENG	0	A	5	Pages	USER_SA
P006	1	0	ENG	0	A	5	Pages	USER_SA
P006	1	0	ENG	0	A	5	Pages	USER_SA

OK

Example of a Test Maintenance report in PTF client

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 Chapter 7, "Identifying Change Impacts," Defining Analysis Rules, page 68.
Category	Which category the change belongs to, as detailed on the Define Analysis Rules page. See Chapter 7, "Identifying Change Impacts," Defining Analysis Rules, page 68.
Object Type	The type of definition that was changed: • Menu • Component • Page • Record • Field
Market	For components, the market; for instance, GBL.
Command ID	A unique and unchanging identifier for a step in a test. Each step has a Command ID and a Sequence Number but the Sequence Number, unlike the Command 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: • Changed • Added • Deleted

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

Access the Create Test Coverage Report page by selecting PeopleTools, Lifecycle Tools, Test Framework, Create Test Coverage Report.

Create Test Coverage Report page

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 all PeopleTools 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 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 following fields are on the Generate Impact Report page:

Project Name	Select a change project to be correlated with PTF metadata.
Report Format	Select the report output format: • PIA - View the report in the application browser. • XML Publisher - Create an XML formatted text file.
Report Scope	Choose whether the report is to be run against all tests in the application's test library, or against a single test. If you choose Single Test, a field appears where you can select an existing test from a drop down list box.
Include Usage Monitor data in the test coverage report if available	Select to include Usage Monitor data in a test coverage report. You need to generate Usage Monitor data while creating your tests. See Chapter 7, "Identifying Change Impacts," Using Usage Monitor Data with PTF, page 81.

Using Usage Monitor Data with PTF

PTF used by itself tracks the use of five object types – records, menus, fields, pages, and components. You can use PTF together with Usage Monitor to track every PeopleTools managed object 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 b 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 all 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 Also

PeopleTools 8.51 PeopleBook: Performance Monitor, "Setting Up the Performance Monitor"

PeopleTools 8.51 PeopleBook: Performance Monitor, "Working With Usage Monitor," Enabling Usage Monitor

Generating Usage Monitor Data

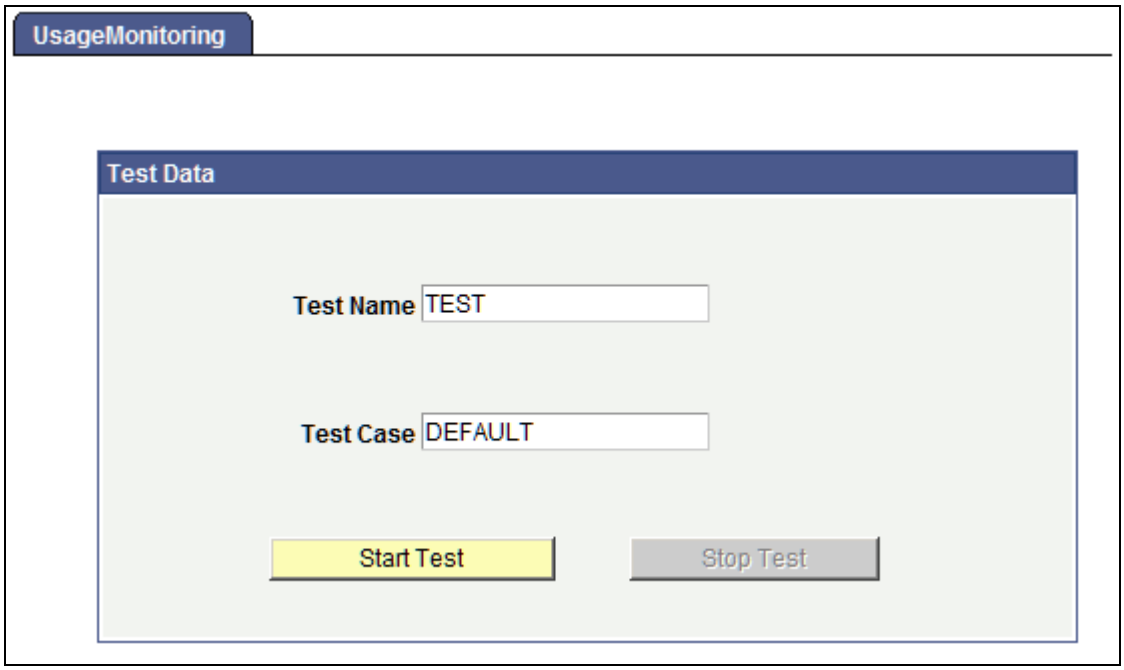
Follow these steps to generate Usage Monitor data along with a PTF test data:

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 the Usage Monitoring link 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.

This is an example of a Usage Monitoring page:



Usage Monitoring page

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).

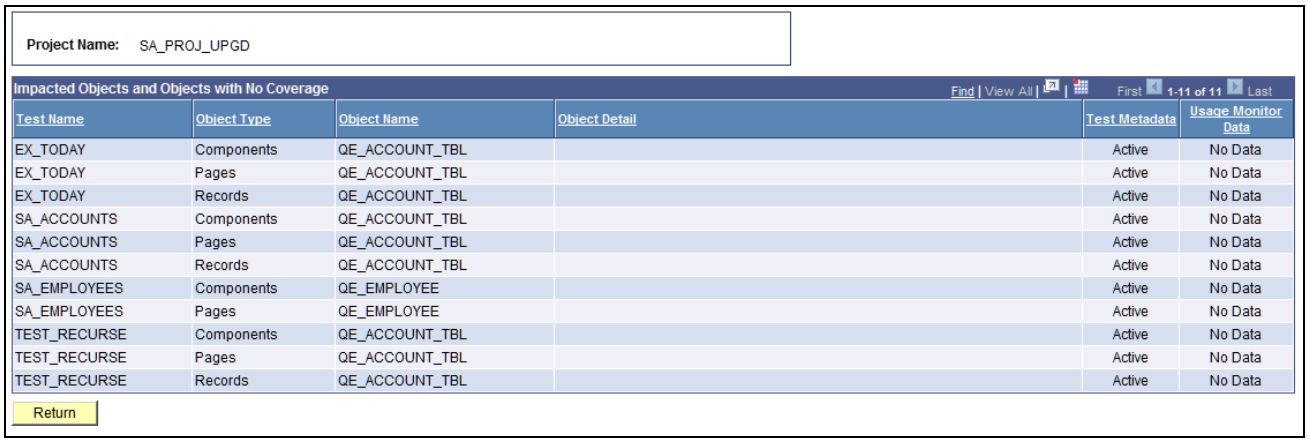
To verify that Usage Monitor Data has been aggregated correctly, run this query in your query tool:

```
SELECT * FROM PSPTUMPMTAGR;
```

Interpreting Test Coverage Reports

Use a Test Coverage report to determine which objects 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:



Project Name: SA_PROJ_UPGD

Impacted Objects and Objects with No Coverage — Find | View All | First 1-11 of 11 Last

Test Name	Object Type	Object Name	Object Detail	Test Metadata	Usage Monitor Data
EX_TODAY	Components	QE_ACCOUNT_TBL		Active	No Data
EX_TODAY	Pages	QE_ACCOUNT_TBL		Active	No Data
EX_TODAY	Records	QE_ACCOUNT_TBL		Active	No Data
SA_ACCOUNTS	Components	QE_ACCOUNT_TBL		Active	No Data
SA_ACCOUNTS	Pages	QE_ACCOUNT_TBL		Active	No Data
SA_ACCOUNTS	Records	QE_ACCOUNT_TBL		Active	No Data
SA_EMPLOYEES	Components	QE_EMPLOYEE		Active	No Data
SA_EMPLOYEES	Pages	QE_EMPLOYEE		Active	No Data
TEST_RECURSE	Components	QE_ACCOUNT_TBL		Active	No Data
TEST_RECURSE	Pages	QE_ACCOUNT_TBL		Active	No Data
TEST_RECURSE	Records	QE_ACCOUNT_TBL		Active	No Data

Return

Example of a Test Coverage report

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	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	If the object is referenced in a PTF test, this value will be *Active* or *Inactive*, reflecting whether within the test, the referenced step that is active or inactive. If the object is not referenced in a PTF test, this value will be *No Data*. This scenario can occur when the Include Usage Monitor Data checkbox is selected and the value in the Usage Monitor Data column is Yes.

Column Name	*Description*
Usage Monitor Data	When the Include Usage Monitor Data checkbox is selected this value will be *Yes* or *No.* When the Include Usage Monitor Data checkbox is deselected the value will be set to *No Data.*

Querying PTF Report Tables

You can query these tables using PS Query to produce your own maintenance and coverage reports:

- PSPTTSTCOVRGRPT

  This PTF table contains the data that is used to create the Test Coverage report.

- PSPTTSTMAINTRPT

  This PTF table contains the data that is used to create the Test Maintenance report.

Chapter 8

Incorporating Best Practices

This chapter discusses PeopleSoft Test Framework (PTF) best practices, which can help you create more robust, efficient, and maintainable tests, and will help to ensure that PTF tests the specific application functionality that you want tested.

Incorporating PTF Best Practices

This section discusses how to incorporate these PTF best practices:

- Record first.
- Document tests.
- Clean up tests.
- Use execution options.
- Use page prompting.
- Use the Process object type.
- Make tests dynamic.

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 enter or set 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.

There is a limitation in PTF when the user record a value set for Calendar objects. The Calendar object in PTF are handled as Text objects, so the user needs to enter the date value in the textbox, and not use the prompt to select the Month/Year/Day.I don't know where is the best place to comment about this limitation.

When you record a date field that includes a calendar object that enables the user to select a date, you must explicitly enter the date value in the edit box. Do not use the calendar to select a date.

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 deselect it first, and then select it. Then delete the extra step in 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.

See Also

Chapter 4, "Creating Tests and Test Cases," Recording Tests, page 41

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
- Line 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.

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.

See Also

Chapter 8, "Incorporating Best Practices," Make Tests Dynamic, page 91

Use Execution Options

Employ 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 inactivating the sequence of steps that was recorded logging in (Steps 1 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	Active	Scroll ID	Type	Action	Recognition	Value
1		1	☐		Browser	Start		
2		2	☐		Text	Set_Value	Name=userid	QEDMO
3		3	☐		Pwd	Set_Value	Name=pwd	1ENC41B9FB38D060CB90E0108BF84...
4		4	☐		Button	Click	Name=Submit	
5			☑		Browser	Start_Login		

Example of using Browser.Start_Login

See Also

Chapter 2, "Installing and Configuring PTF," Configuring Execution Options, page 20

Chapter 10, "Test Language Reference," Start_Login, page 98

Use Page Prompting

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 take the user directly to 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.

See the Page type actions Prompt and PromptOK for more details.

See Also

Chapter 10, "Test Language Reference," Prompt, page 115

Chapter 10, "Test Language Reference," PromptOK, page 115

Use the Process Object Type

The Process object type with a Run action is useful for running processes through Process Scheduler.

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.
```

Replacing a recorded sequence of steps with the Process.Run step will make your test more robust. It will be less likely to fail if a change in performance affects processing time or if a PeopleTools enhancement modifies the structure of the Process Monitor page.

See Also

Chapter 10, "Test Language Reference," Process, page 116

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 examples of test scenarios can 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 replace the recorded steps values with the appropriate dynamic construct that will ensure that the test will work at execution time.

See Also

Chapter 5, "Developing and Debugging Tests," Using Variables, page 52

Chapter 5, "Developing and Debugging Tests," Using Conditional Logic, page 53

Chapter 5, "Developing and Debugging Tests," Incorporating Scroll Handling, page 57

Chapter 5, "Developing and Debugging Tests," Using Reserved Words, page 50

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.

Take these steps 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.

See Also

Chapter 4, "Creating Tests and Test Cases," Creating Test Cases, page 42

Chapter 5, "Developing and Debugging Tests," Calling Tests, page 61

Chapter 9

Using the PTF Test Language

This chapter provides an overview of the PeopleSoft Test Framework (PTF) test structure and discusses 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.

Seq		ID	Active	Scroll ID	Type	Action	Recognition	Value
1		1	☑		Browser	Start_Login		
2		5	☑		Link	Click	id=fldra_PT_PEOPLETOOLS	
3		8	☑		Link	Click	id=fldra_PT_SECURITY	
4		9	☑		Link	Click	id=fldra_PT_USER_PROFILES	
5		10	☑		Browser	FrameSet	TargetContent	
6		11	☑		Link	Click	Name=default|innerText=User Profiles	
7		12	☑		Browser	FrameSet		
8		13	☑		Link	Click	innerText=Copy User Profiles	
9		14	☑		Browser	FrameSet	TargetContent	
10		15	☑		Text	Set_Value	Name=PSOPRDEFN_SRCH_OPRID	QEDMO
11		16	☑		Button	Click	Name=#ICSearch	
12		17	☑		Text	Set_Value	Name=DERIVED_CLONE_OPRID	PBDMO
13		18	☑		Text	Set_Value	Name=DERIVED_CLONE_OPRDEFNDESC	PBDMO
14		19	☑		Pwd	Set_Value	Name=DERIVED_CLONE_OPERPSWD	1ENC5D7742D060C0C2B6F40662C87...
15		20	☑		Pwd	Set_Value	Name=DERIVED_CLONE_OPERPSWDCONF	1ENC5D7742D060C0C2B6F40662C87...
16		21	☑		Page	Save		
▸*			☑					

Example of a PTF test

Each step in a test is composed of eight fields, as defined in the following table:

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

ID — A system-generated unique identifier for each line, or 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.

Active Deselect this field to inactivate a step. PTF will skip inactive steps when the test runs. Each step is active by default. 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 object 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 and Verify.

Recognition The means that PTF uses to identify the object within the application. Commonly, this is the HTML ID property.

Value In a typical recorded step, this is the value the tester entered for an object.

In a step recorded in field check mode, this would be the value that was present in the object when it was checked.

As a rule, 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:

Seq		ID	Active	Scroll ID	Type	Action	Recognition	Value
1		23	☑		Text	Set_Value	Name=EMPLID	KU0001

Example of a PTF test step

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.

The following chapter, "Test Language Reference," is a reference for all the PTF object types and actions.

PTF Test Language

This section discusses:

- Validation
- Parameters
- Variables
- Reserved words

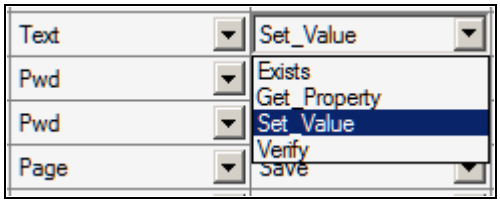
Validation

Each object type allows only certain actions. Similarly, each action can only be used with certain object types.

The PTF Test Editor automatically limits your choice of actions based on the object 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*:



Action drop-down list for Text

Conversely, if you have a step with the Type field set to *Text* and the Action field set to *Set_Value*, you can change the Type field to *Pwd* since *Set_Value* is a valid action for both Text and Pwd object types.

However, if you change the Type field to *Link*, the Test Editor automatically changes the Action field value to *Click* and limits the values in the drop-down list to the valid actions for Link.

This example shows a step using the Link type and a drop-down list with the valid actions for Link:

Actions drop-down list for Link

Parameters

Typically, you place parameters in the ID 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 Chapter 10, "Test Language Reference," Radio, page 119.

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 ID field for Browser.Start or Browser.Start_Login.

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 OPRID field to the variable *&OPRID*. The second step populates the field OPRDEFNDESC with the value in the variable *&OPRID*.

Text	Get_Property	Name=OPRID;ret=&OPRID	
Text	Set_Value	Name=OPRDEFNDESC	&OPRID

Example of using variables in test steps

See Also

Chapter 10, "Test Language Reference," Variable, page 128

Chapter 5, "Developing and Debugging Tests," page 49

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.

Text	Set_Value	Name=EFFDT_FROM	#TODAY

Example of using the #TODAY reserved word

See Also

Chapter 5, "Developing and Debugging Tests," Using Reserved Words, page 50

Chapter 10, "Test Language Reference," Reserved Words, page 135

Chapter 10

Test Language Reference

This chapter discusses:

- Object types and their associated actions.
- Common actions.
- Reserved words.
- Functions.

Object Types

This section lists each of the PTF object types and defines the actions associated with the object type. The object types are listed in alphabetical order.

Browser

These are the actions associated with the Browser object type.

Close

Description

Closes the current browser (that is, the one with the execution focus).

FrameSet

Description

Sets the focus in a browser frame.

Start

Description

Starts the browser instance where the test will be executed. Uses the URL from the selected 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, and password from the selected execution option and the language selected in the test Language field.

WaitForNew

Description

Waits for a new browser to appear, then continues execution with the new browser.

Specify a timeout value in seconds in the Value column. The default is 10.

Button

These are the actions associated with the Button object type.

Click

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Click, page 129.

Exists

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Exists, page 130.

Get_Property

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Get_Property, page 131.

Chart

These are the actions associated with the Chart object type.

ChartClick

Description

Performs a click in a chart section.

Parameters

Parameter	***Description***
chart=*value*;	The index for the chart image on the page.
idx=*value*; url=*value*; alt=*value*;	The section recognition string. It can be the section index, the section URL, or the alternative text.

GetText

Description

Performs a click in a chart section.

Parameters

Parameter	*Description*
chart=*value*;	The index for the chart image on the page.
idx=*value*; url=*value*; alt=*value*;	The section recognition string. It can be the section index, the section URL, or the alternative text.
ret=*&variable*;	The return value.

Example

This is an example of the GetText action for a Chart object type:

Chart	GetText	chart=0;ret=&chart_val;idx=3	
Log	Message	The value for index 3 is '&chart_val;'	
Chart	GetText	chart=0; ret=&chart_val; idx=2;	
Log	Message	The value for the index 2 is '&chart_val;'	

Example of GetText action for a Chart object type

CheckBox

These are the actions associated with the CheckBox object type.

Exists

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Exists, page 130.

Get_Property

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Get_Property, page 131.

Set_Value

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Set_Value, page 133.

Verify

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Verify, page 134.

ComboBox

These are the actions associated with the ComboBox object type.

Exists

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Exists, page 130.

Get_Property

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Get_Property, page 131.

Set_Value

Description

Sets the field value in the ComboBox to the value specified in Value. You can also use this action with the #LIST# reserved word to verify whether items exist in the list.

Example

This example sets the field value to One and verifies that the values Two and Three exist:

ComboBox	Set_Value	Name=DUMMY_FIELD	#LIST#[One]|Two|Three

Example of using the Set_Value action with the #LIST# reserved word

See Also

Chapter 10, "Test Language Reference," #LIST#, page 139

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 the field value is set to Two and verifies that the values One and Three exist:

ComboBox	Verify	Name=DUMMY_FIELD	One|[Two]|Three

Example of Verify action

Conditional

These are the actions associated with the Conditional object type.

If_Then

Description

Validates the given condition. When it is True, the system executes the lines between the If_Then step and the End_If step.

If_Then supports these logical operators:

<>, >=, <=, >, <, =

Example

This example shows the use of the Conditional construct:

Image	Get_Property	Name=DERIVED_IC_GBL_BRA$img; prop=alt; ret=&flagstate	
Conditional	If_Then	&flagstate=Expand section Brazil	
Image	Click	Name=DERIVED_IC_GBL_BRA$img	
Conditional	End_If		

Example of the If_Then conditional construct

End_If

Description

The close statement of the If_Then construct.

DataMover

This is the action associated with the DataMover object type.

Exec

Description

Executes a DataMover script. The output folder will be the one selected for the PeopleSoft Data Mover output folder. If the output folder is not specified, it uses the system temp folder.

Example

This example shows the use of the Exec action:

DataMover	Exec		C:\temp\emp_imp.dms

Example of the DataMover Exec action

File

This is the action associated with the File object type.

Upload

Description

Uploads a file.

In the Recognition column specify the string to get the object from the page. In the Value column specify a full file pathname.

Example

This example shows the use of the Upload action:

File	Upload	name=#OrigFileName	C:\Temp\MyFile.txt

Example of the File Upload action

HTMLTable

These are the actions associated with the HTMLTable object type.

CellClick

Description

Clicks on a specific HTMLTableCell based on the index parameter.

Parameters

Parameter	*Description*
index=*n*/*n*/*n*;	The table, row, column index string. For example: `index = 1/2/3;` This function 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

Parameter	*Description*
index=*n*/*n*/*n*;	The table, row, column index string. For example: `index = 1/2/3;` This function clicks on 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

Parameter	***Description***
index=*n*/*n*/*n*;	The table, row, column index string. For example: `index = 1/2/3;` This function clicks on the third column of the second row of the first table.
name=*value*;	The HTMLImage's NameProp value.

CellClickOnLink

Description

Clicks the link specified in the table cell location based on the index parameter.

Parameters

Parameter	***Description***
index=*n*/*n*/*n*;	The table, row, column index string. For example: `index = 1/2/3;` This function clicks on the third column of the second row of the first table.
link=*value*;	The link text value.

CellExists

Description

Determines whether a cell exists.

Parameters

Parameter	*Description*
index=*n*/*n*/*n*;	The table, row, column index string. For example: `index = 1/2/3;` This function clicks on 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

Returns a string containing the HTMLTable index, row, and column of the value specified in the text parameter.

Parameters

Parameter	*Description*
text=*value*;	The text to look for on the component.
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.
ret=*&variable*	The return value.

CellGetValue

Description

Returns the contents of an HTMLTableCell.

Parameters

Parameter	*Description*
index=*n*/*n*/*n*;	The table, row, column index string. For example: `index = 1/2/3;` This function clicks on 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

Parameter	*Description*
table=*value*;	The table index.
row=*value*;	The row index.
ret=*&variable*	The return value.

RowCount

Description

Returns the number of rows for the HTMLTable.

Parameters

Parameter	*Description*
table=*value*;	The table index.
ret=*&variable*	The return value.

Example

This is an example of several of the HTMLTable actions:

HTMLTable	CellGetIndex	text=Calendar; equal=true; ret=&index;	
HTMLTable	CellClick	index=sum(&index, -1, 3, /);	
HTMLTable	CellGetValue	index=&index; ret=&cell_value;	
HTMLTable	CellGetIndex	text=Main Menu; equal=true; ret=&index;	
HTMLTable	CellClick	index=sum(&index, -1, 3, /);	
HTMLTable	ColCount	table=10; row=1; ret=&col_count;	
HTMLTable	RowCount	table=1; ret=&row_count;	
HTMLTable	CellClick	index=1/2/3; ret=&exists;	

Example of actions for the HTMLTable object type

Image

These are the actions associated with the Image object type.

Click

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Click, page 129.

Exists

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Exists, page 130.

Get_Property

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Get_Property, page 131.

Link

These are the actions associated with the Link object type.

Click

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Click, page 129.

Exists

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Exists, page 130.

Get_Property

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Get_Property, page 131.

Log

Use the Message, Pass, Warning, and Fail log actions to write entries to the 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.

Use the Snapshot action to capture a screen image.

These are the actions to add entries to the execution log.

Fail

Description

Logs an entry with a status of Fail.

Message

Description

Logs a message with a status of Info.

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.

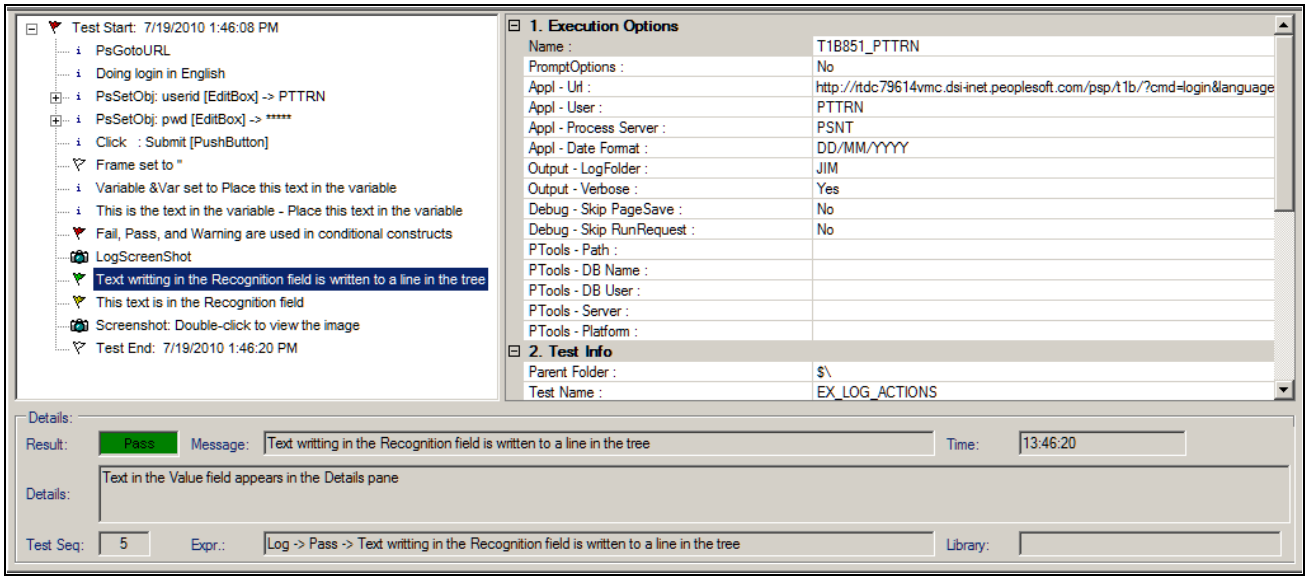
Example

This example shows how the Log actions log text:

Type	Action	Recognition	Value
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 Details 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	

Example of Log action steps

This log example shows how text from the Log actions appears in the Log Viewer:



Example of a PTF log

LongText

These are the actions associated with the LongText object type.

Exists

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Exists, page 130.

Get_Property

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Get_Property, page 131.

Set_Value

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Set_Value, page 133.

Verify

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Verify, page 134.

MultiSelect

These are the actions associated with the MultiSelect object type.

Exists

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Exists, page 130.

Get_Property

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Get_Property, page 131.

Set_Value

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Set_Value, page 133.

Verify

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Verify, page 134.

Page

These are the actions associated with the Page object 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 page name in the Recognition field.

Example

This example shows the use of the Go_To action:

Page	Go_To	Roles	

Example of the Go_To action

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

PromptOK

Description

Closes the search page. If the specified action is update, this action selects the first returned value.

Example

This example shows the use of the Prompt and Prompt OK actions:

Browser	Start_Login		
Page	Prompt	SQA_MENU.SQA_SIMPLECOMP.GBL	add update
Text	Set_Value	name=SQA_SIMPLEREC_SQA_DATAID	US001
Page	PromptOk		

Example of the Prompt and PromptOK actions

Save

Description

Attempts to save the current page. This action checks for the SkipSavePage flag in the execution options.

Process

The Process actions run processes through Process Scheduler.

Run

Description

Runs a Process Scheduler process.

Parameters

Parameter	***Description***
prcname=*value*;	The process name.
wait=*value*;	True - the test waits for the process to finish. False - the test does not wait for the process to finish. The default is False.
outtype=*value*;	The process output type.
outformat=*value*;	The process output format.
outfile=*value*;	The process output file.

Parameter	Description

expected=*value*;	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:
	`expected=Success;`
	`expected=No Success;`
	If the final status equals the expected status, then a Pass is logged; if not, a Fail is logged.

ret=*&variable*	The return value. True - the process completed successfully. False - the process did not complete successfully.

Example

This example shows the use of the Run action to run a process:

Browser	Start_Login		
Page	Prompt	RC_MANAGE_TELEMARKETERS.RUN_RCOM01.GBL	add update
Text	Set_Value	name=PRCSRUNCNTL_RUN_CNTL_ID	test
Page	PromptOk		
Text	Set_Value	name=RUN_RCOM01_BUSINESS_UNIT	US001
Process	Run	prcname=RCOM01; wait=true;	

Example of the Run action

Run_Def

Description

Changes the default behavior of the run process. This action only supports one parameter per step. For multiple parameters, you will need multiple steps.

Parameters

Parameter	Description
RunButton=*value*;	The run button name.
ProcessMonitorLink=*value*;	The Process Monitor link name.

Parameter	*Description*
ProcessInstanceField=*value*;	The field where the process instance ID will appear.
QueuedTimeout=*value*;	Overwrites the local option value (in minutes).
PostingTimeout=*value*;	Overwrites the local option value (in minutes).
ProcessingTimeout=*value*;	Overwrites the local option value (in minutes).
ExceptionTimeout=*value*;	General timeout, in minutes, for all the states that are not in the local option.
QueuedResult=*value*;	Overwrites the local option value. Valid values are *FAIL* and *WARN*.
PostingResult=*value*;	Overwrites the local option value. Valid values are *FAIL* and *WARN*.

See Also

Chapter 2, "Installing and Configuring PTF," Configuring Local Options, page 18

Pwd

These are the actions associated with the Pwd object type.

Exists

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Exists, page 130.

Set_Value

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Set_Value, page 133.

Query

This is the action associated with the Query object type.

Exec

Description

Runs a query in PeopleSoft Query and downloads the results.

Parameters

Parameter	***Description***
outFolder=*value*;	The folder where the result will be saved. If this parameter is missing, the system will use the value in the local option.
outFormat=*value*;	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 option.
param=*value*;	The list of comma delimited values for all the query parameters.

Example

This example shows the use of the Query Exec action:

Query	Exec	Name=MESSAGES_FOR_MSGSET	outfolder=c:\temp;outformat=XML;param...
Query	Exec	Name=MESSAGES_FOR_MSGSET	param=3;
Query	Exec	Name=MESSAGES_FOR_MSGSET	outFormat=CSV;param=2;

Example of the Query Exec action

Radio

These are the actions associated with the Radio object type.

Exists

Description

The value property is required to validate whether a radio button exists on the page. It can be defined in the ID field using `value=`*`value`* or in the Value field.

See Chapter 10, "Test Language Reference," Exists, page 130.

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 ID field using the `Value=` parameter.

The following example shows how to use either the Value field or the `value=` parameter to identify a radio button:

Radio	Exists	name=SQA_SIMPLEREC_SQAOPTION; ret=&RadioEx1	2
Radio	Exists	name=SQA_SIMPLEREC_SQAOPTION; ret=&RadioEx2; value=3	

Example of setting the value for a Radio object type

Get_Property

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Get_Property, page 131.

Set_Value

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Set_Value, page 133.

Verify

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Verify, page 134.

Scroll

These are the actions associated with the Scroll object type.

The Scroll ID field is required for all Scroll action types.

See Also

Chapter 5, "Developing and Debugging Tests," Incorporating Scroll Handling, page 57

Action

Description

Takes an action based on the row specified by Key_Set.

Parameters

Parameter	***Description***
ret=*&variable*;	The return value. It returns the position index for the field acted upon, based on the row identified using Key_Set.

Specify the action in the Value column.

This table lists the valid values for the 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.

Value	*Description*
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 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=*text_to_find*
not	Check that the row is not in the scroll. Add a Fail to the log if the row is found.

Definition

Description

Use the Definition action to overwrite the default name object of scroll buttons and the scroll parent.

Parameters

Parameter	*Description*
def=*value*;	One of the definition values. Example: `def=parent;` The following table gives valid values for the def parameter.

type=*value*;	The object type for the new action.
	Example 1:
	`type=Image;`
	Example 2:
	`type=PushButton;`

value=*value*;	The scroll parent variable or the new object recognition string.
	Example 1:
	`value=&Scr1;`
	Example 2:
	`value=Name=$ICField3$newm$0$$img$0;`

This table lists the valid values for the def parameter:

Value	*Description*
add	Overwrite the Add new row button.
del	Overwrite the Delete row button.
next	Overwrite the Next button.
prev	Overwrite the Previous button.
first	Overwrite the First button.
last	Overwrite the Last button.
parent	Reassign the parent for a specific scroll area.

Key_Set

Description

Defines the set of keys that define a scroll.

Specify the object type and field name as parameters. Specify the key value in the Value column.

Parameters

Parameter	*Description*
type=value;	The object type for the key field. Example: `type=Text;`
name=value;	The name of the key field.

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

These examples show the use of scroll actions:

Scroll ID	Type	Action	Recognition	Value
1	Scroll	Key_Set	type=Text; Name=PSE_SCR_REC01_PSE_KEY_LVL1	ROW1
1	Scroll	Action	RET=&sCR1	upd
	Text	Verify	Name=PSE_SCR_REC01_PSE_KEY_LVL1&Scr1	ROW1
	ComboBox	Set_Value	Name=PSE_SCR_REC01_PSE_COMBO&sCR1	second
	Text	Set_Value	Name=PSE_SCR_REC01_CON_CHAR_01&Scr1	updated

Example of the Key_Set and Action actions (1 of 2)

Scroll ID	Type	Action	Recognition	Value
1	Scroll	Key_Set	type=Text; Name=PSE_SCR_REC01_PSE_KEY_LVL1	000A
1	Scroll	Action	ret=Scr&1	fnd=0a
	ComboBox	Set_Value	Name=PSE_SCR_REC01_PSE_COMBO&Scr1	first
	LongText	Set_Value	Name=PSE_SCR_REC01_CON_LONG_01&Scr1	changed
	Text	Verify	Name=PSE_SCR_REC01_PSE_KEY_LVL1&Scr1	000A
1	Scroll	Key_Set	type=EditBox; Name=SQA_SIMPLER_L1_SQA_DUMMY1	2
1	Scroll	Reset		

Example of scroll actions, including RowCount and Reset (2 of 2)

This is an example of the use of the Definition action with the def parameter to insert a new row and to reassign the parent definition:

Scroll ID	Type	Action	Recognition	Value
1	Scroll	Definition	def=parent; value=&scr1	
1	Scroll	Action	ret&Scr1	upd
	Text	Set_Value	Name=SQA_SIMPLER_L1_SQA_DUMMY1&Scr1	2
2	Scroll	Definition	def=parent; value=Scr&1	
2	Scroll	Key_Set	type=Text; Name=SQA_SIMPLER_L2_SQA_DUMMY2	B
2	Scroll	Action	ret&Scr2	upd
2	Text	Set_Value	Name=SQA_SIMPLER_L2_SQA_DUMMY1&Scr2	B

Example of the Definition action

Span

These are the actions associated with the Span object type.

Exists

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Verify, page 134.

Get_Property

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Verify, page 134.

MouseOver

Description

Fires the mouseover event to show a popup page. Enter the page name in the Recognition field.

See Also

PeopleTools 8.51 PeopleBook: PeopleSoft Application Designer Developer's Guide, "Using Page Controls," Using Pop-up Pages

MouseOverClose

Description

Fires the mouseout event to close a popup page. Enter the page name in the Recognition field.

See Also

PeopleTools 8.51 PeopleBook: PeopleSoft Application Designer Developer's Guide, "Using Page Controls," Using Pop-up Pages

Example

The following examples show how MouseOver and MouseOverClose can be used with popup pages.

To validate a field value in a mouseover popup window:

1. Click and drag the Field Check button 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:

Browser	Start		
Span	MouseOver	id=QE_EMPLOYEE_EMPLID	en
Span	Verify	Comment=QE_EMPL2_DEPTID	22000
Span	MouseOverClose	id=QE_EMPLOYEE_EMPLID	

Example of MouseOver with Span.Verify

To record an action for an object inside a mouseover popup window:

1. Click and drag the Field Check button 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:

Browser	WaitForNew		
Span	MouseOver	id=QE_EMPLOYEE_EMPLID	
Link	Click	Name=QE_EMPL_PHOTO2_URL|innerText=UR...	
Span	MouseOverClose	id=QE_EMPLOYEE_EMPLID	

Example of MouseOver with Link.Click

Verify

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Verify, page 134.

Example

This step validates a Span object that contains informational text:

Span	Verify	ClassName=PSTEXT	(Includes values assigned for types spec...

Example of the Span.Verify action

Test

This is the action associated with the Test object type.

Exec

Description

Calls another test or library.

Specify the test name in the ID field and the test case name in the Value field.

Example

This step calls the test 851_USER_PROFILE_01:

Test	Exec	851_USER_PROFILE_01	CASE01

Example of the Test Exec action

Text

These are the actions associated with the Text object type.

Exists

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Exists, page 130.

Get_Property

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Get_Property, page 131.

Set_Value

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Set_Value, page 133.

Verify

Description

The description for this action is in the Common Actions section.

See Chapter 10, "Test Language Reference," Verify, page 134.

Variable

This is the action associated with the Variable object type.

Set_Value

Description

Assigns a value to the given variable.

Example

This example shows how to set a variable and how to use variables in other steps:

Log	Message	This is a test of variables	
Variable	Set_Value	&Var1=This is the value for Var1	
Log	Message	The value in Var1 is '&Var1'	
Variable	Set_Value	&Var2	This is Var2
Log	Message	The value in Var2 is '&Var2'	
Variable	Set_Value	&Var3=&Var1	
Log	Message	Var3 is a clone of Var1. The value is &Var3	

Example of the Set_Value action

See Also

Chapter 9, "Using the PTF Test Language," Variables, page 96

Common Actions

The actions in this section can be used with multiple object types. The object types that support the action are listed with each action.

Click

Description

Performs a mouse click on the specified object.

This is the list of objects that support this action:

- Button

  See Chapter 10, "Test Language Reference," Button, page 98.

- Image

  See Chapter 10, "Test Language Reference," Image, page 109.

- Link

  See Chapter 10, "Test Language Reference," Link, page 110.

Example

This example shows the use of the Click action with a Button object and a Link object:

Button	Click	Name=PB_FILTER	
Link	Click	Name=LAST_NAME$0	

Example of the Click action

Exists

Description

Checks whether the object exists on the page.

This is the list of objects that support this action:

- Button

  See Chapter 10, "Test Language Reference," Button, page 98.

- CheckBox

  See Chapter 10, "Test Language Reference," CheckBox, page 100.

- ComboBox

  See Chapter 10, "Test Language Reference," ComboBox, page 101.

- Image

  See Chapter 10, "Test Language Reference," Image, page 109.

- Link

  See Chapter 10, "Test Language Reference," Link, page 110.

- LongText

  See Chapter 10, "Test Language Reference," LongText, page 112.

- MultiSelect

  See Chapter 10, "Test Language Reference," MultiSelect, page 113.

- Pwd

  See Chapter 10, "Test Language Reference," Pwd, page 118.

- Radio

  Refer to the Radio object type entry for details.

  See Chapter 10, "Test Language Reference," Radio, page 119.

- Text

  See Chapter 10, "Test Language Reference," Text, page 127.

Parameters

Parameter	*Description*
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:

Text	Exists	name=USERID;ret&exists	

Example of the Exists action

Get_Property

Description

Gets the property value 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 Chapter 5, "Developing and Debugging Tests," Using the Message Tool, page 49.

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. PTF does not get label properties.

This is the list of objects that support this action:

- Button

  See Chapter 10, "Test Language Reference," Button, page 98.

- CheckBox

  See Chapter 10, "Test Language Reference," CheckBox, page 100.

- ComboBox

  See Chapter 10, "Test Language Reference," ComboBox, page 101.

- Image

  See Chapter 10, "Test Language Reference," Image, page 109.

- Link

  See Chapter 10, "Test Language Reference," Link, page 110.

- LongText

  See Chapter 10, "Test Language Reference," LongText, page 112.

- MultiSelect

  See Chapter 10, "Test Language Reference," MultiSelect, page 113.

- Radio

  Refer to the Radio object type entry for details.

  See Chapter 10, "Test Language Reference," Radio, page 119.

- Text

  See Chapter 10, "Test Language Reference," Text, page 127.

Parameters

Parameter	***Description***
prop=*value*;	The property name.
ret=*&variable*;	The return value.

Example

This example shows the use of the Get_Property action:

Text	Get_Property	Name=userid; ret=&TxtProp; prop=Status;	
Log	Message	This is the Status: &TxtProp;	
Button	Get_Property	Name=Submit; ret=&BtnProp; prop=tagName;	
Log	Message	This is the TagName for the button: &BtnProp;	
Button	Get_Property	Name=Submit; ret=&BtnProp; prop=Value;	
Log	Message	This is the Value for the button: &BtnProp;	

Example of the Get_Property action

Set_Value

Description

Sets the field value in the browser object.

This is the list of objects that support this action:

- CheckBox

  See Chapter 10, "Test Language Reference," CheckBox, page 100.

- ComboBox

  See Chapter 10, "Test Language Reference," ComboBox, page 101.

- LongText

  See Chapter 10, "Test Language Reference," LongText, page 112.

- MultiSelect

  See Chapter 10, "Test Language Reference," MultiSelect, page 113.

- Pwd

  See Chapter 10, "Test Language Reference," Pwd, page 118.

- Radio

  Refer to the Radio object type entry for details.

  See Chapter 10, "Test Language Reference," Radio, page 119.

- Text

  See Chapter 10, "Test Language Reference," Text, page 127.

Example

This is an example of the use of the Set_Value action:

Type	Action	Recognition	Value
LongText	Set_Value	name=SQA_SIMPLEREC_SQA_DESCRIPTION	long
Text	Set_Value	name=SQA_SIMPLEREC_PSE_CHECKBOX	ebox
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

Example of the Set_Value action

Verify

Description

Compares the value in the browser to the expected value, and adds a pass or fail log entry for the validation.

This is the list of objects that support this action:

- CheckBox

  See Chapter 10, "Test Language Reference," CheckBox, page 100.

- ComboBox

  See Chapter 10, "Test Language Reference," ComboBox, page 101.

- LongText

  See Chapter 10, "Test Language Reference," LongText, page 112.

- MultiSelect

  See Chapter 10, "Test Language Reference," MultiSelect, page 113.

- Radio

  Refer to the Radio object type entry for details.

  See Chapter 10, "Test Language Reference," Radio, page 119.

- Span

  See Chapter 10, "Test Language Reference," Span, page 125.

- Text

  See Chapter 10, "Test Language Reference," Text, page 127.

Example

This is an example of the Verify action:

Text	Verify	ID=PSE_DATES_SQA_DATE	#TODAY

Example of the Verify action

Reserved Words

This section defines PTF reserved words. The reserved words are listed alphabetically.

#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:

Text	Set_Value	ID=PA_CLC_SUMMARY_CALC_NAME	KUSPTEST
Text	Verify	ID=PA_CLC_SUMMARY_CALC_NAME	KUSPTEST

Example of setting a value and verifying a value in two steps

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 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:

Text	Set_Value	ID=PA_CLC_SUMMARY_CALC_NAME	#CHECK#KUSPTEST

Example of setting a value and verifying a value using #CHECK#

#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:

Text	Set_Value	Name=PA_CALCULATION_BEN_CMDT_DATE	#DIS#07/12/2000

Example of using #DIS#

#DTTM

Description

Similar to #TODAY, the #DTTM reserved word inserts the current date and time into a field in the application.

See Also

Chapter 10, "Test Language Reference," #TODAY, page 141

#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:

Text	Set_Value	Name=PA_CLC_PLN_INPT_BENEFIT_PLAN	#EXIST#
Text	Set_Value	Name=PA_CLC_PLN_INPT_BENEFIT_PLAN	#EXIST#KUHP

Example of using #EXIST#

See Also

Chapter 10, "Test Language Reference," #NOTEXIST#, page 140

#FAIL#

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 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:

Text	Set_Value	ID=PA_CLC_SUMMARY_CALC_NAME	#FAIL#KUSPTEST

Example of #FAIL#

See Also

Chapter 10, "Test Language Reference," #WARN#, page 142

#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:

Type of Match	***Pattern***	***Match (Log a Pass)***	***No Match (Log a Fail or Warn)***
Multiple characters	a*a	aa, aBa, aBBBa	ABC
Multiple characters	*ab*	abc, AABB, 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:

Text	Set_Value	Name=PA_CLC_PLN_INPT_BENEFIT_PLAN	#LIKEF#US*
Text	Set_Value	Name=PA_CLC_PLN_INPT_BENEFIT_PLAN	#LIKEF#US*1

Example of using #LIKEF#

#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:

Text	Set_Value	Name=PA_PROP_VOUCH_CREAT_DTTM	#LIKEF##TODAY*

Example of using #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#.

See Also

Chapter 10, "Test Language Reference," #LIKEF#, page 137

#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.

To check one or more values and then set an item in a drop-down list box, list the items separated by a a vertical pipe (|) and place brackets ([]) around the item that you want to select.

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.

ComboBox	Set_Value	Name=PA_CALCULATION_CALCULATION_TYPE	#LIST#Individual|Pre-Defined Group|Pre-Defined List
ComboBox	Set_Value	Name=PA_CALCULATION_CALCULATION_TYPE	#LIST#[Individua]|Pre-Defined Group|Pre-Defined List

Example of using #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:

ComboBox	Set_Value	Name=PA_CALCULATION_CALCULATION_TYPE	#LIST#I|G|L
ComboBox	Set_Value	Name=PA_CALCULATION_CALCULATION_TYPE	#LIST#[I]|G|L

Example of using #LIST# with translate values

#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.

See Also

Chapter 10, "Test Language Reference," #EXIST#, page 136

#NOTHING

Description

This reserved word deletes a value from a text box, selects a blank value from a drop-down list box, or verifies that a text box or drop-down list box 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 Step 13 #NOTHING selects a blank value in the Calculation Reason field and then, in Step 14, it verifies that the field is blank:

ComboBox	Set_Value	Name=PA_CALCULATION_CALC_REASON	#NOTHING
ComboBox	Verify	Name=PA_CALCULATION_CALC_REASON	#NOTHING

Example of using #NOTHING

#PREFIX#

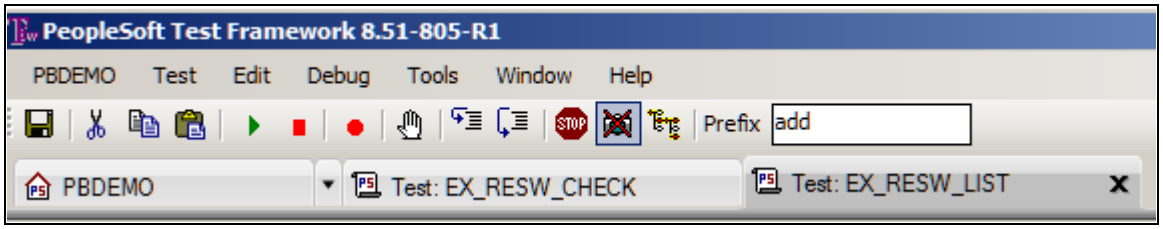
Description

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:



Example of Prefix field

The following test step would enter the value *addUSER* into the User ID field:

Text	Set_Value	Name=userid	#PREFIX#USER

Example of using #PREFIX#

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:

Text	Set_Value	ID=PA_CLC_EMP_VW_EVENT_DT	#TODAY

Example of using #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:

Text	Set_Value	ID=PA_CLC_EMP_VW_EVENT_DT	#TODAY+10

Example of using #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.

See Also

Chapter 10, "Test Language Reference," #FAIL#, page 137

Functions

This section lists the PTF functions.

Sum

Syntax

Sum(text, sum_value, index_part, delimiter)

Description

Sum works with the HTMLTable indexes. You can also use Sum to perform a simple numeric sum.

Parameters

Parameter	*Description*
Text	The HTMLTable index string, such as 2/5/4. This string is the return value of CellGetIndex. See Chapter 10, "Test Language Reference," CellGetIndex, page 107.
Sum_Value	The value that you want to add or subtract. The default action is addition.
Index_part	The index on the section that will be modified.
Delimiter	The character that delimits each section in the text value.

Example

The follow table presents examples of using Sum.

Expression	*Result*
Sum(2/5/4, 2, 1, /)	4/5/4 2 is added to the first part of the string.
Sum(2/5/4, -1, 3, /)	2/5/3
Sum(&index, -4, 3, /)	4 is subtracted from the third part of the string in the variable &index.
Sum(&Dumvar, 2, 1,)	This is an example of a simple numeric sum. 2 is added to the value in &Dumvar. Assuming the value in &Dumvar is 3, the result would be 5. Note that the trailing comma is required.

Appendix A

Reserved Words Quick Reference

This quick reference lists the PeopleSoft Test Framework (PTF) reserved words.

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 Chapter 10, "Test Language Reference," #CHECK#, page 135.
#DIS#	Checks whether an object is display-only. See Chapter 10, "Test Language Reference," #DIS#, page 136.
#DTTM	Enters the current date and time into the application. See Chapter 10, "Test Language Reference," #DTTM, page 136.
#EXIST# and **#NOTEXIST#**	Check whether a field exists or does not exist on the page. See Chapter 10, "Test Language Reference," #EXIST#, page 136 and Chapter 10, "Test Language Reference," #NOTEXIST#, page 140.
#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 Chapter 10, "Test Language Reference," #FAIL#, page 137 and Chapter 10, "Test Language Reference," #WARN#, page 142.
#LIKEF# and **#LIKEW#**	Match strings using LIKE. If no match exists, PTF logs a Fail or Warning. PTF does not update the value. See Chapter 10, "Test Language Reference," #LIKEF#, page 137 and Chapter 10, "Test Language Reference," #LIKEW#, page 139.

#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 Chapter 10, "Test Language Reference," #LIST#, page 139.

#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 Chapter 10, "Test Language Reference," #NOTHING, page 140.

#PREFIX#

Substitutes the text in the Prefix field in the Test Editor for *#PREFIX#* in the Value field.

See Chapter 10, "Test Language Reference," #PREFIX#, page 140.

#TODAY

Enters the current date into the application.

See Chapter 10, "Test Language Reference," #TODAY, page 141.

Index

Numerics/Symbols

A

B

C

D

E

F

I

L

M

N

P

Q

R

S

T

U

V