ORACLE
PEOPLESOFT

Oracle's PeopleTools PeopleBook

PeopleTools 8.52: PeopleSoft Test Framework

June 2013

ORACLE

PeopleTools 8.52: PeopleSoft Test Framework
SKU pt8.52tptf-b0613

Trademark Notice

License Restrictions Warranty/Consequential Damages Disclaimer

Warranty Disclaimer

Restricted Rights Notice

Hazardous Applications Notice

Third Party Content, Products, and Services Disclaimer

Contents

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Chapter 7

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 *PeopleBooks and the PeopleSoft Online Library* in the online PeopleBooks Library for your PeopleTools release.

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 83.

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

  See Chapter 5, "Developing and Debugging Tests," Incorporating Scroll Handling, page 71.

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

  See Chapter 10, "Test Language Reference," Process, page 149.

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

  See Chapter 10, "Test Language Reference," Query, page 152.

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

You should be aware that PTF is not designed to:

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

Terminology

This table defines some PTF terms:

Asset	See Test Asset.
Execution Options	A list of application environments available to the tester. Execution options store application environment information such as URL, user ID, password, and Process Scheduler server, and information needed to run DataMover. PTF supplies this information to the test by default when a test does not explicitly specify such information.
Explorer	See PTF Explorer.
Grid	See Scroll
Hook	Establish a connection between a PTF test and a PeopleSoft application browser.
Library	Similar to a test, a library contains one or more steps that together automate some discrete amount of test functionality. Unlike a test, a library is never executed by itself. Rather, libraries are meant to be called (sometimes repetitively) by tests.
Log	An object that saves the experience of a single test execution event. Logs report the success or failure of the test execution and include messages and screen shots to indicate where errors occurred.
Log Manager	A tool that enables PTF administrators to purge unneeded logs

Maintenance	The process of updating PTF tests and test cases to reflect object modifications present in upgrades or changes to the PeopleSoft application. This is done by way of a direct connection to the PeopleSoft metadata, not by executing the test. For example, if a field is renamed in an upgrade, the PTF maintenance process can warn the user that a test containing a reference to the old field name will likely fail to find the object by the old identification method. The maintenance process can help the user find the obsolete field reference and replace it with the valid (renamed) field reference before executing the test.
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.
Scroll	A scroll represents a rowset, which is a set of rows of data uniquely identified by one or more key fields. Rows in a scroll can contain child rowsets. Scrolls are rendered on PeopleSoft pages as grids of data or as a grouping of fields in a scroll area.
Scroll Area	See Scroll.
Step	The smallest unit of test functionality in PTF. A test will contain a number of steps. A step typically corresponds to a single manual test step or test instruction.

Test Asset

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

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

Test

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

Test Case

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

Test Editor

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

Variable

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

Chapter 2

Installing and Configuring PTF

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:

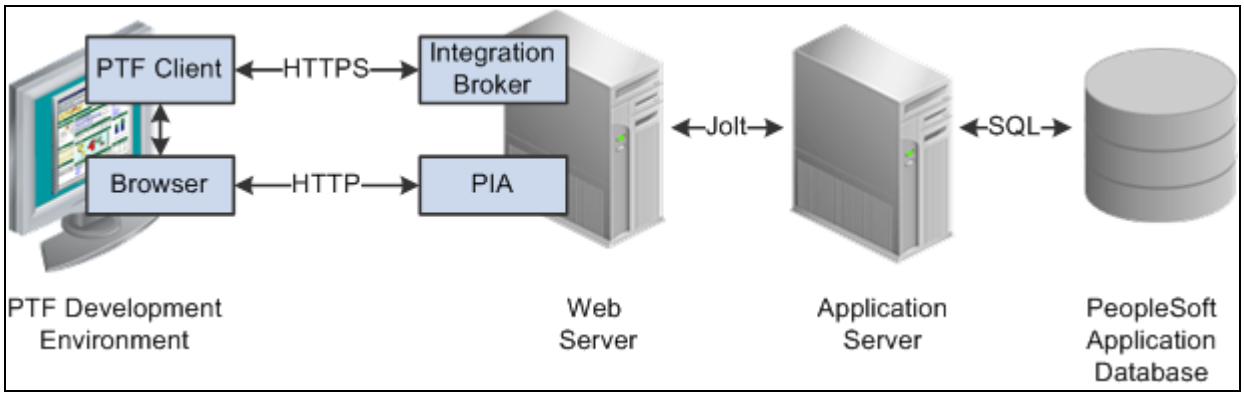


Diagram of the PTF development environment

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 or greater can be used as a PTF environment.

This section discusses how to:

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

Verifying Integration Broker Setup

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

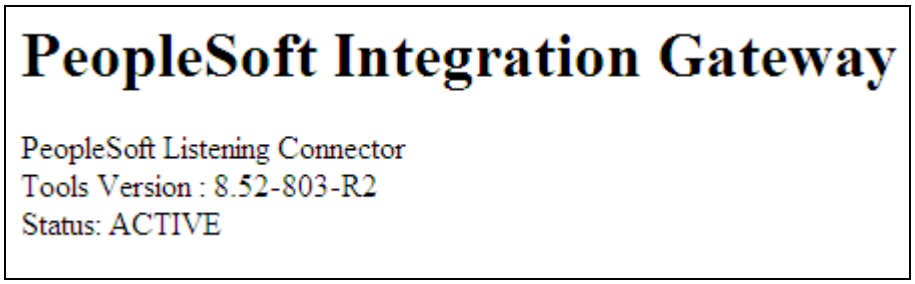
1. In your PeopleSoft application, navigate to PeopleTools, Integration Broker, Configuration, Gateways.
2. Verify that the Gateway URL field references the correct machine name.
3. Click the Ping Gateway button.
4. Verify that the message returns a status of ACTIVE.
5. Click the Gateway Setup Properties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

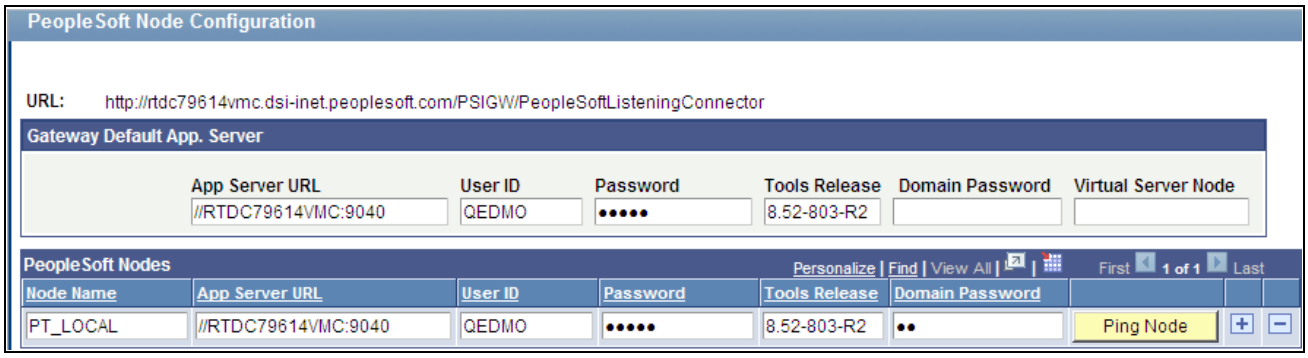
The port number in the URL (80 in this example) is the http port of the web server. The default is 80.

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



Integration Gateway Ping message

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



PeopleSoft Node Configuration page

The port number in the App Server URL (9040 in this example) generally corresponds with the JSL Port Number as defined in the Application Server configuration. The default port number is 9000.

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

See Chapter 2, "Installing and Configuring PTF," Creating a Connection to a PTF Environment, page 16.

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

See Chapter 2, "Installing and Configuring PTF," Creating a Connection to a PTF Environment, page 16.

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 8.52: 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, Modify, and Delete Tests	No*	Yes	Yes
Create, Modify, and Delete Test Cases	Yes	Yes	Yes

Privilege	*PTF User*	*PTF Editor*	*PTF Administrator*
Create or Modify Execution Options	No	No	Yes
Use Log Manager	No	No	Yes
Define Configuration Options	No	No	Yes
Create Test Maintenance Reports	No	No	Yes
Create Test Coverage Reports	No	No	Yes

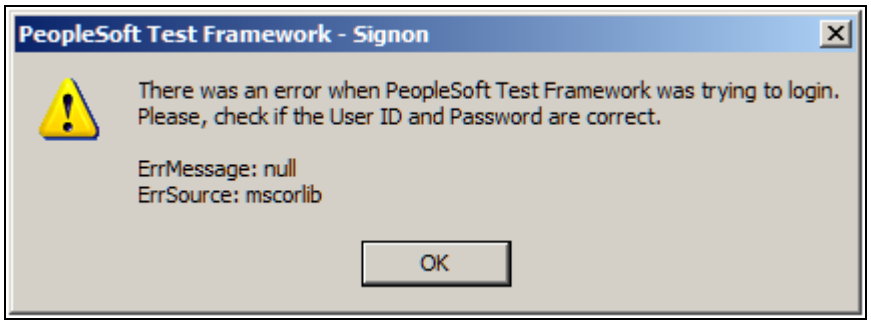
*PTF User can create, modify, and delete tests only in myFolder.

Modify Test includes these actions:

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

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

If PTF security is not configured properly ou may receive the following error when signing on the the PTF client:



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.

See Also

PeopleTools 8.52: Security Administration, "Administering User Profiles"

Chapter 2, "Installing and Configuring PTF," Defining PTF Configuration Options, page 11

Chapter 7, "Identifying Change Impacts," Creating Test Maintenance Reports, page 86

Chapter 7, "Identifying Change Impacts," Creating Test Coverage Reports, page 96

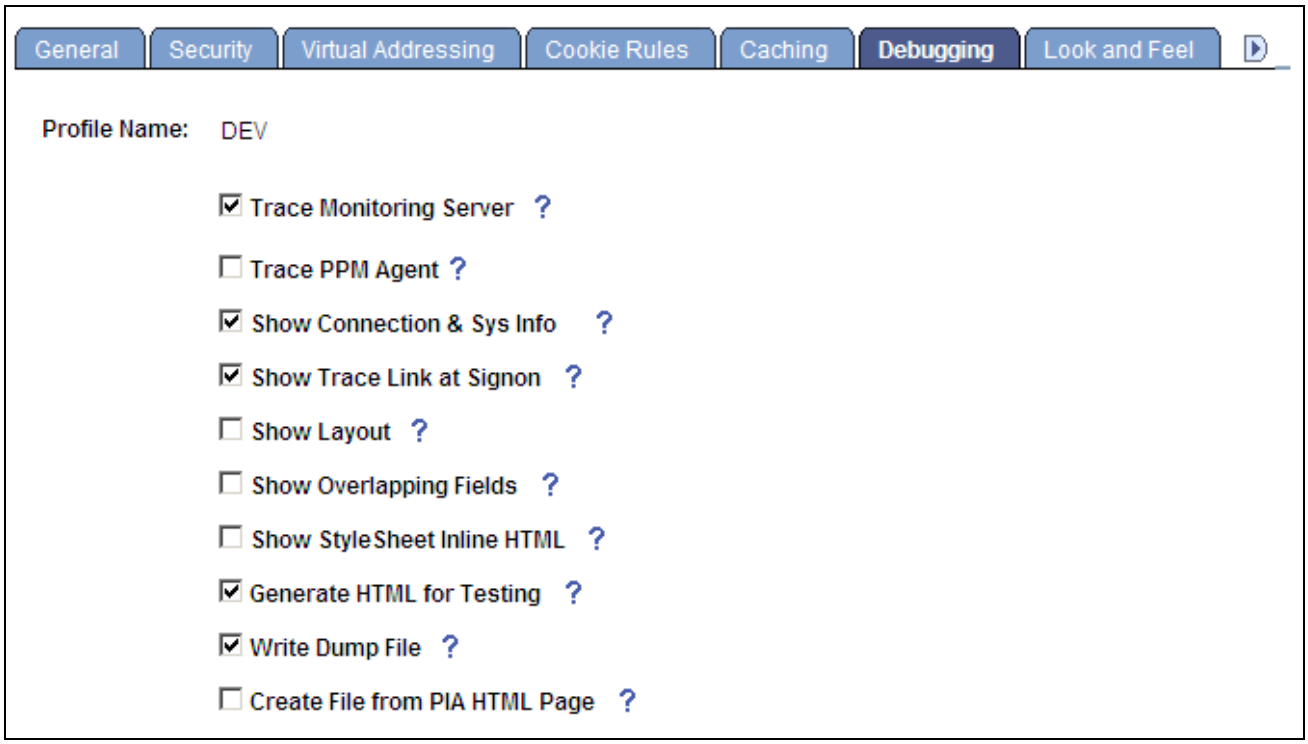
Configuring the Web Profile

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

1. Navigate to the PeopleTools, Web Profile, Web Profile Configuration.
2. Select the profile name for your environment. (This is the web profile that was selected during web server installation.)
3. Click the Debugging tab.
4. Check the Generate HTML for Testing checkbox.

   If this option is not selected PTF will not record HTML objects correctly.
5. Check the Show Connection & Sys Info checkbox.

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



Web Profile Configuration - Debugging page

Defining PTF Configuration Options

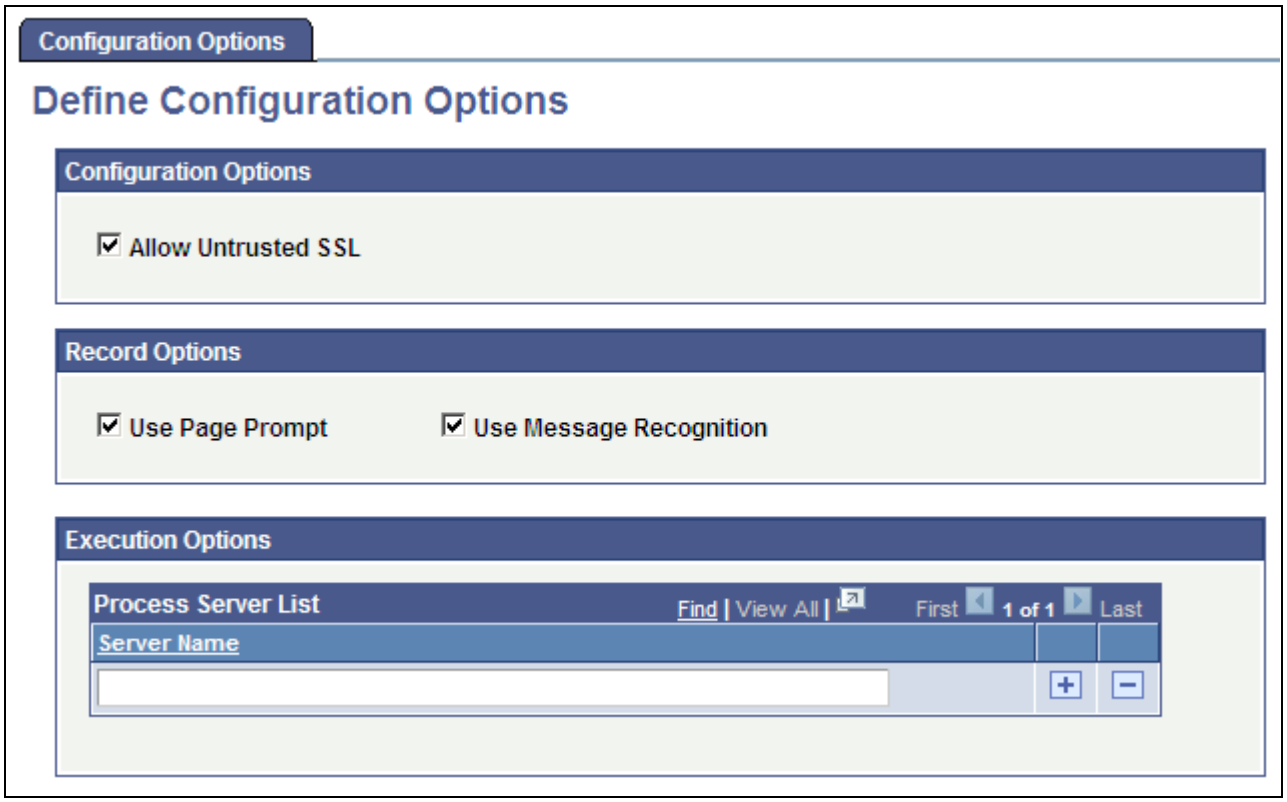
This section discusses the page used to configure PTF options.

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.

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

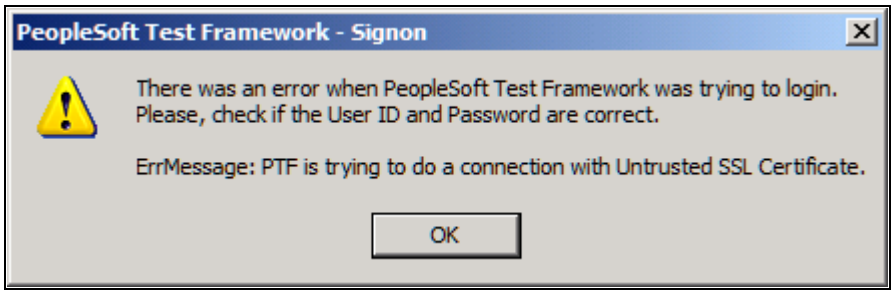
See Chapter 10, "Test Language Reference," Page, page 147.

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

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

See Chapter 2, "Installing and Configuring PTF," Configuring Execution Options, page 23.

If you receive the following error message, select the Allow Unstrusted SSL checkbox if you want to enable PTF to connect without a trusted SSL certificate:



Error Message: PTF is trying to do a connection with Untrusted SSL Certificate.

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

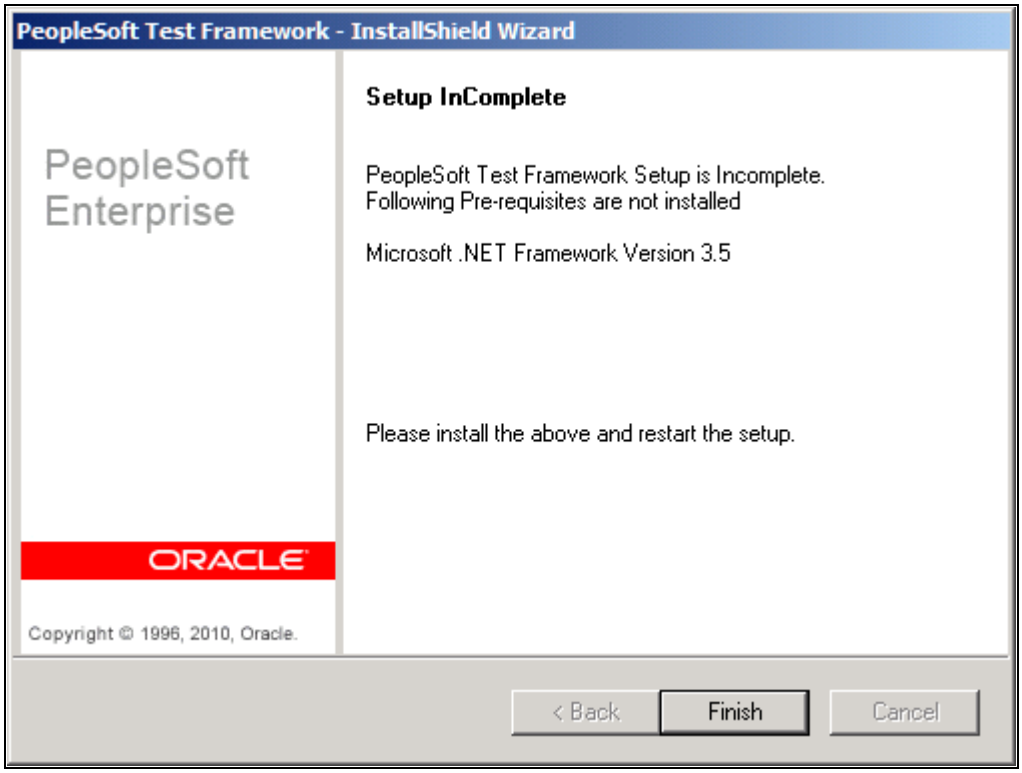
PTF client installation has the following requirements:

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

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

3. Microsoft .NET Framework v3.5.

   If Microsoft .NET Framework v3.5 is not present, the PTF Installer returns the following error during installation:



Microsoft .NET Framework error message

4. In order to install PTF, you will need read and write access to the PTF home directory (`C:\Program Files\PeopleSoft\PeopleSoft Test Framework`) by default.

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

Configuring the Browser Security Settings

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

If browser security settings are not properly configured you may encounter problems with PTF test playback.

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 domains for the test applications.
7. Add entries for both http and https.

   For example:

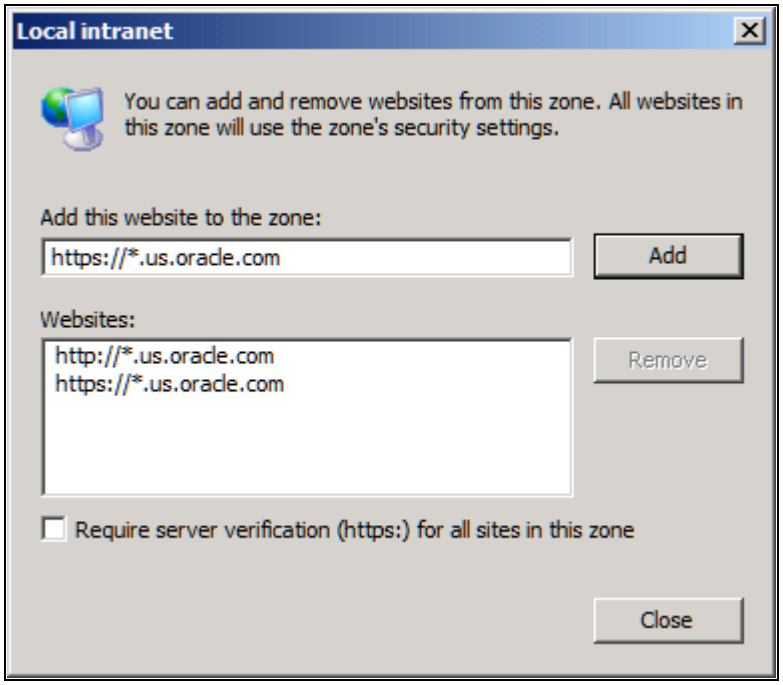
   ```
   http://*.<domain_name>
   https://*.<domain_name>
   ```

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

   ```
   http://rtdc79614vmc.us.oracle.com:80/PTTRN/signon.html
   ```

   then the domain name is `us.oracle.com`
8. Click the Add button.
9. Click the Close button.
10. Click the OK button to close each open dialog box.

This example shows the Local intranet dialog box:



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 or later installation, setup.exe is located in the `<PS_HOME>\setup\PsTestFramework` directory.

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

2. Run setup.exe.

   The installation wizard appears.

3. Click the Next button.

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

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

5. Click the Next button.

   The Ready to Install the Program page appears.

6. Click the Install button.

   The InstallShield Wizard Complete page appears.

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

   Your PTF client software installation is complete.

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

   - Locate the PTF shortcut on your desktop.
   - Navigate to Start, All Programs, PeopleSoft Test Framework.
   - In Windows Explorer, navigate to `C:\Program Files\PeopleSoft\PeopleSoft Test Framework` (or the installation directory you specified in Step 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 and port for the environment. Contact your Integration Broker administrator or system administrator for the correct values.

The format for the Server:Port field is:

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

For example:

```
rtdc79614.us.oracle.com:443
```

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

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

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

For example:

```
https://rtdc79614vmc.dsi-inet.peoplesoft.com:443/PSIGW/HttpLis
```

Use Proxy Select this field if using a proxy server.

When you select this checkbox the Proxy Information link is enabled.

Proxy Information Click this link to enter details for the proxy server.

Enter the following information for the proxy server:

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

  If you use network authentication, use the DOMAIN\USER format.
- Password: Enter the password.

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

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

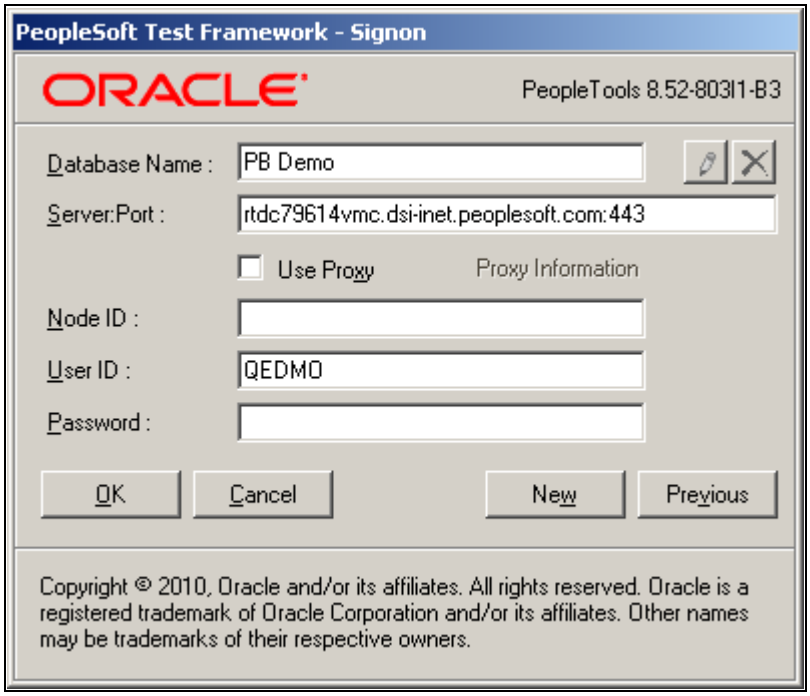
See Chapter 2, "Installing and Configuring PTF," Verifying Integration Broker Setup, page 6.

User	Enter a valid user ID for the PeopleSoft application that contains the environment. The user ID must have one of the PTF security roles assigned. Contact your security administrator to add the role if required. If this user ID does not have PTF access you will receive a signon error: See Chapter 2, "Installing and Configuring PTF," Setting Up Security, page 8.
Password	Enter the password for this user.

4. Click the OK button.

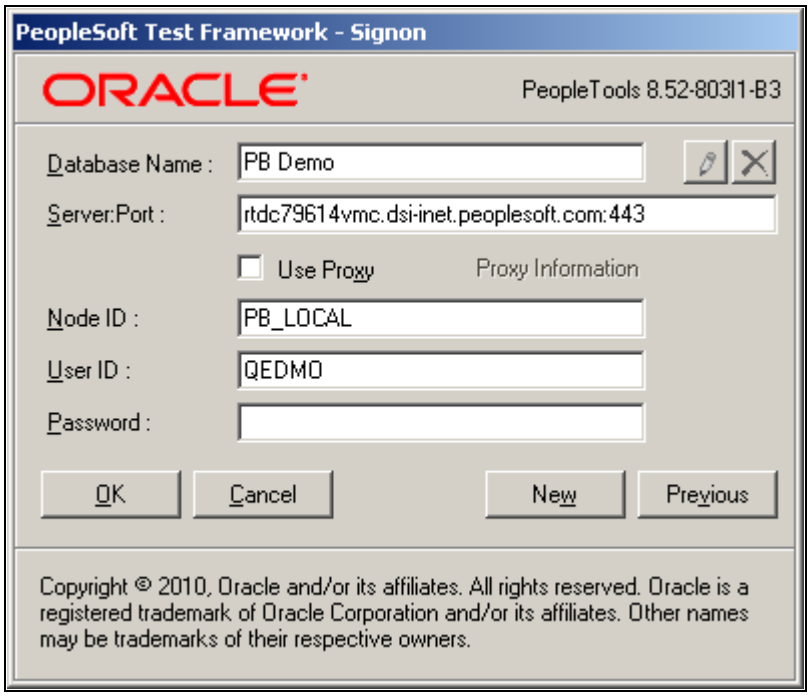
   PTF launches with a connection to the designated environment.

The following example shows a completed PeopleSoft Test Framework - Signon dialog box. In this example the Node ID field is left blank because the default gateway is used.



Example of a completed environment signon dialog box

This example shows a completed PeopleSoft Test Framework - Signon dialog box where the default gateway is not used, which requires a Node ID to be specified:



Example of a completed environment signon dialog box with Node ID specified

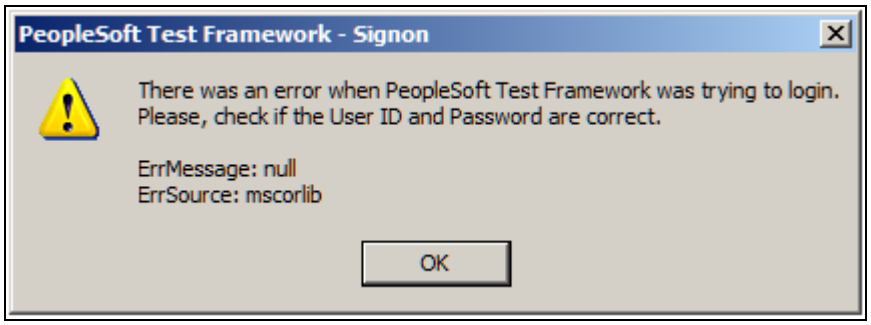
---

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

Troubleshooting Tips

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

You will receive the following signon error if PTF security has not been configured correctly:



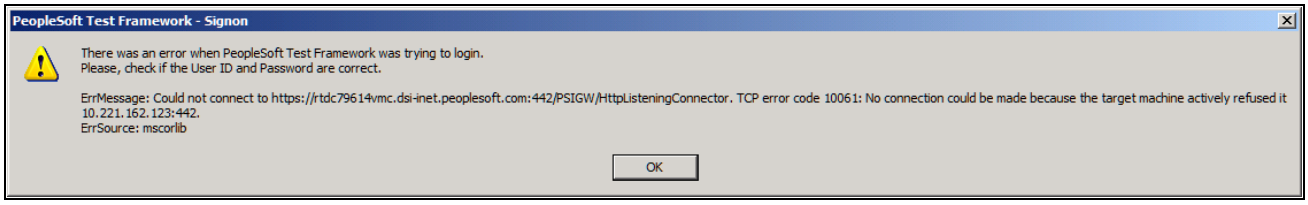
Signon error message

Possible causes and solutions for this error are:

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

- The user ID for the ANONYMOUS node does not have PTF privileges. Add at least the PTF User role to the user profile.

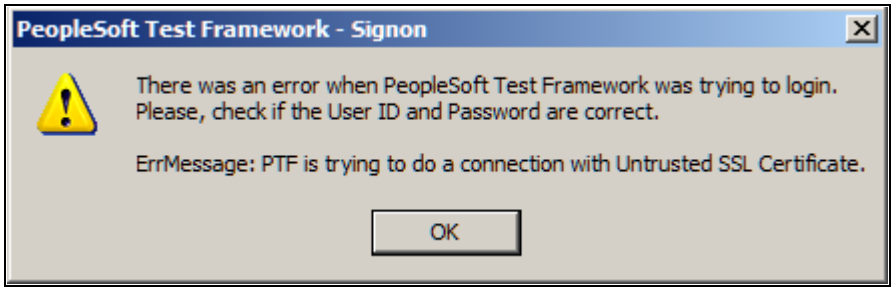
You will receive the following error message if you specified the wrong HTTPS port in the environment login URL:



Wrong HTTPS error message

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

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



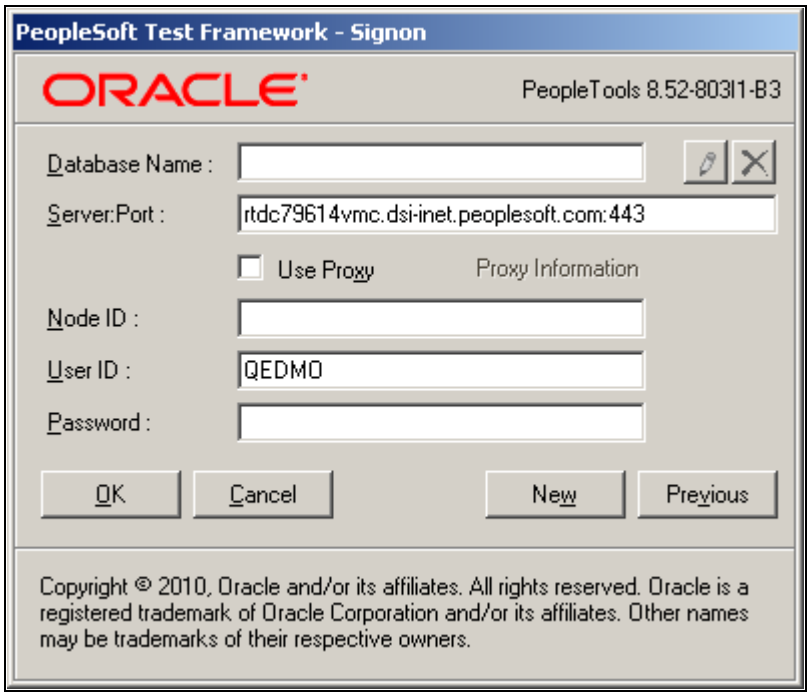
Untrusted SSL Certificate error message

See Chapter 2, "Installing and Configuring PTF," Defining PTF Configuration Options, page 11.

Selecting a PTF Environment

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

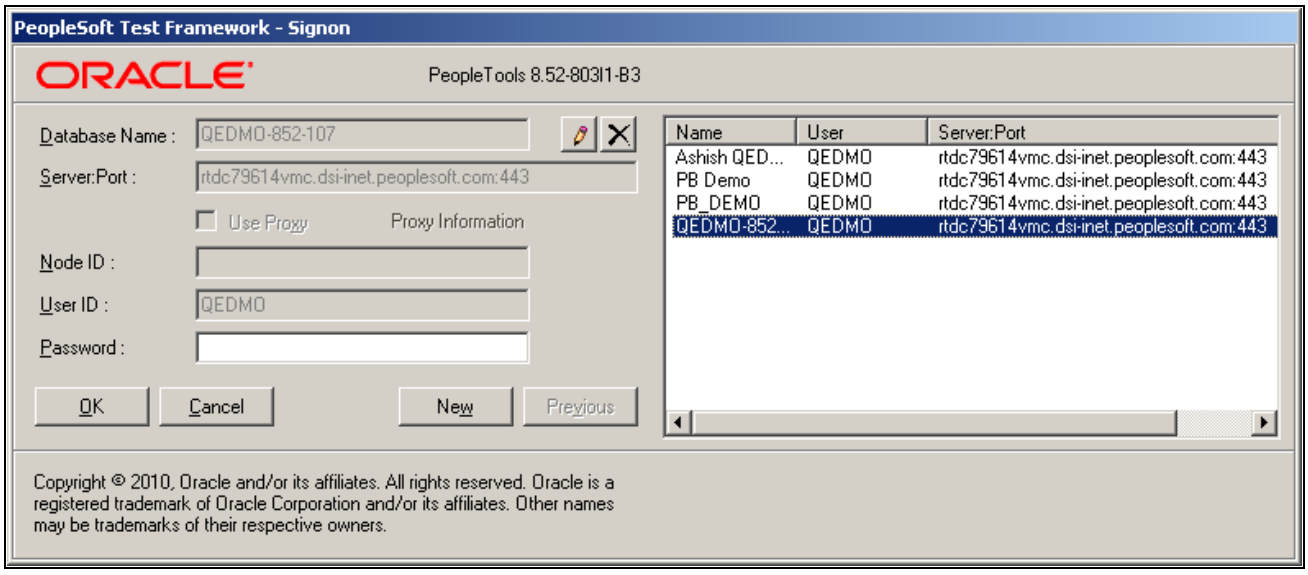
You can enter the password and click the OK button to launch PTF using that environment, or you can click the New button to create another environment login.



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

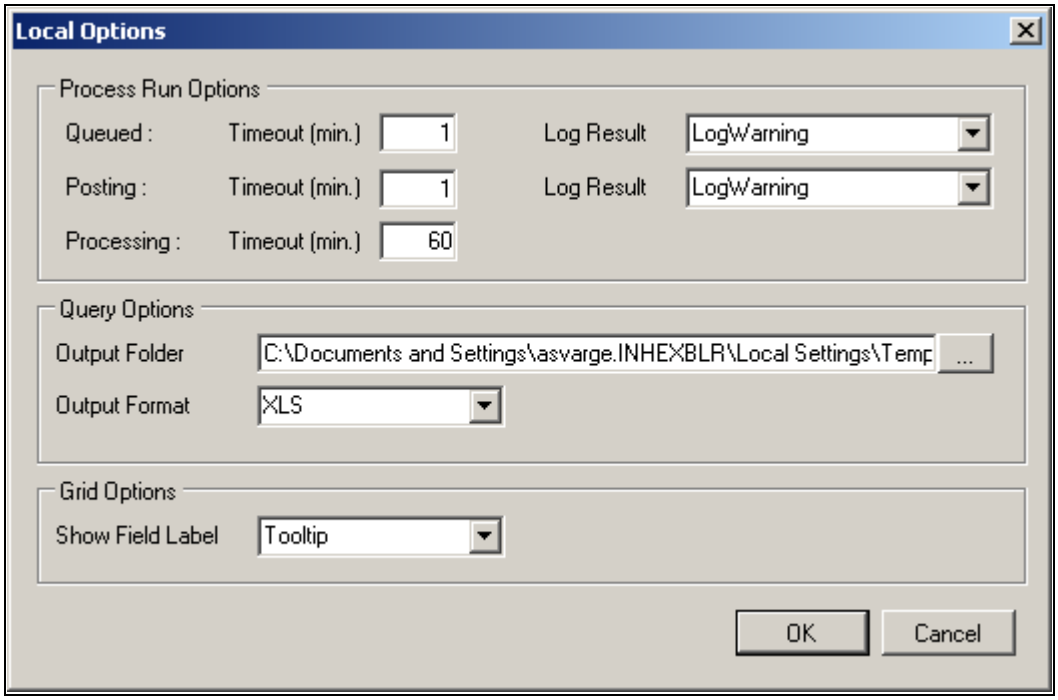
Environment signon settings are specific to the machine on which the PTF client is installed. The environment signon settings are stored in the environments.xml file in the PTF data directory (`C:\Documents and Settings\<User>\Application Data\PeopleSoft\PeopleSoft Test Framework`) by default.

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.

Grid Options

Show Field Label	Select Tooltip to show field labels as tooltips (hover text). Select Column to show field labels in a column in the test window.

Configuring Execution Options

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

You can access Execution Options either through a PeopleSoft application browser session, or through the PTF client.

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.

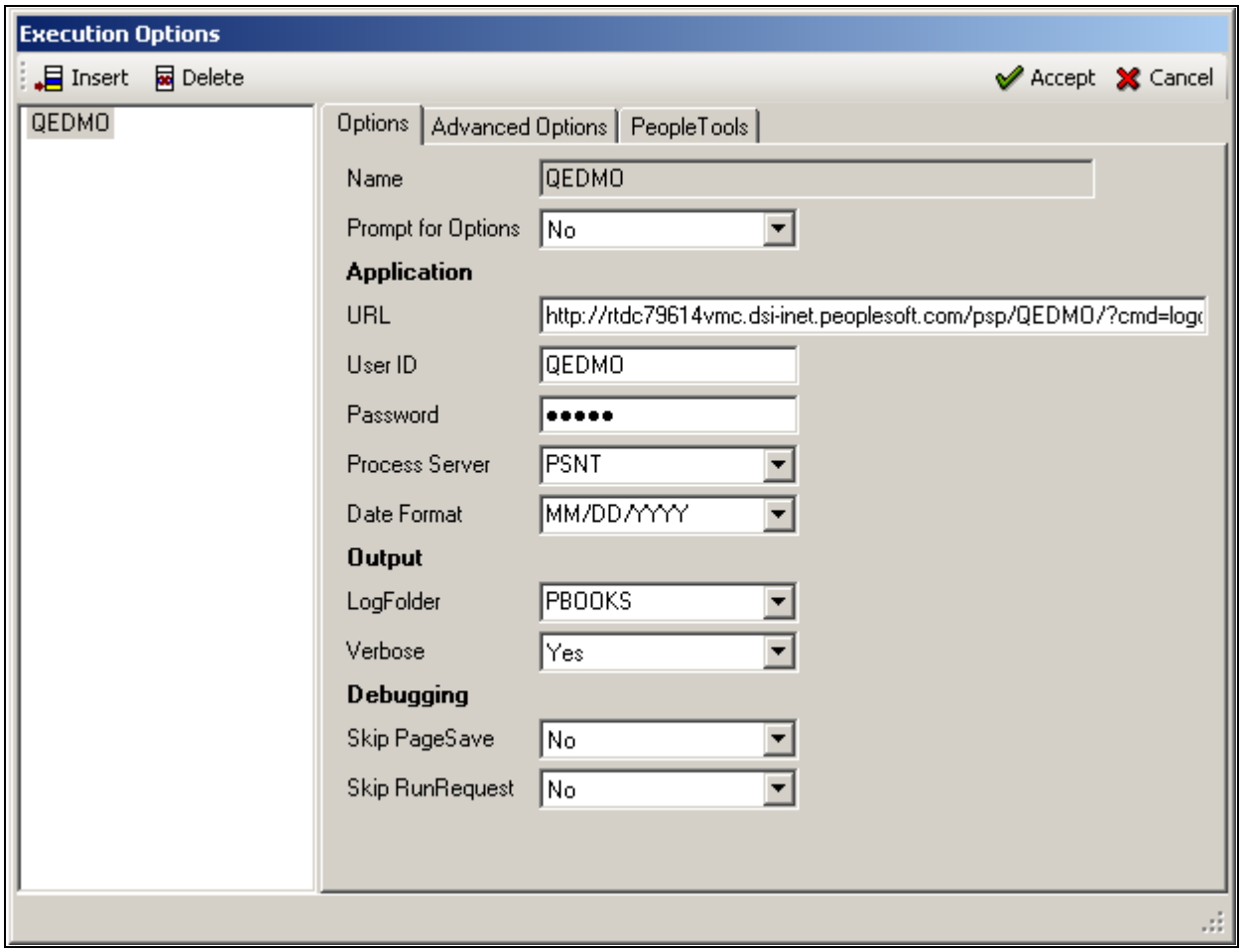
Execution options are stored as part of the metadata for a PTF environment and can be viewed and selected in the PTF client by all users of that environment. Only a PTF administrator (a user with the PTF Administrator role) is able to insert, delete, or modify execution options.

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.

Configuring Execution Options in PTF Client

You can define execution options either in the PTF client or using the Define Execution Options page in the PeopleSoft Internet Architecture.

This section describes how to define execution options in the PTF client.



Execution Options dialog – Options tab

In the PTF client, available execution options are listed in the left pane of the Execution Options dialog. 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 Configuration Options page. See Chapter 2, "Installing and Configuring PTF," Defining Configuration Options, page 11.
Date Format	Select a date format.

Output

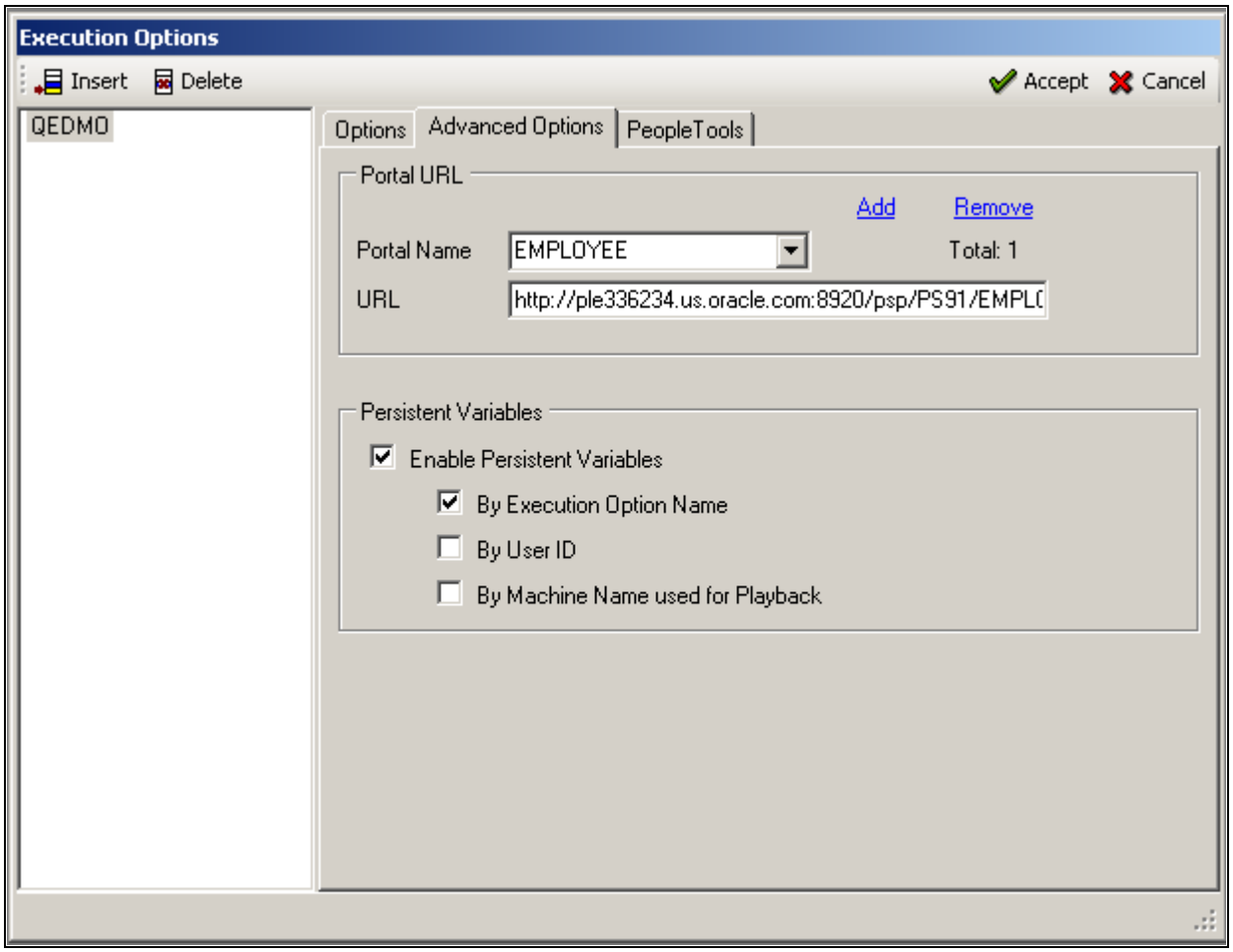
LogFolder	Select or enter the folder name to which logs will be written. If the folder does not exist it will be created.
Verbose	Specify whether to use verbose logging. Select *Yes* to log a detail line in the log for each step executed in the test. Select *No* to log only the test rollup status (Pass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

Debugging

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

Advanced Options Tab

This example shows the Execution Options dialog - Advanced Options tab in the PTF client:



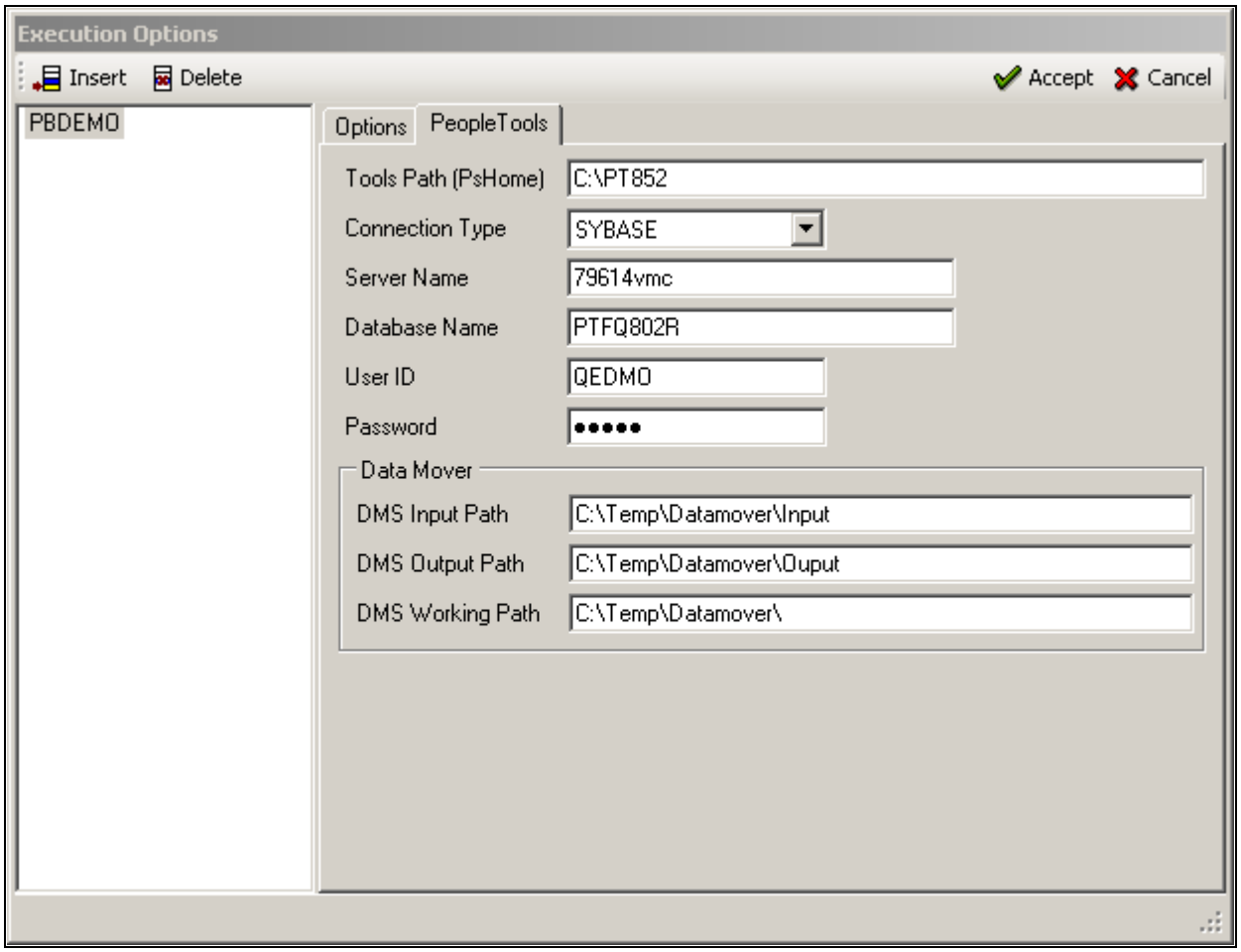
Execution Options dialog – Advanced Options tab

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

The following fields are on the Advanced Options tab:

PeopleTools Tab

This example shows the Execution Options dialog - PeopleTools tab in the PTF client:



Execution Options dialog – 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.

Connection Type Select the connection type.

Server Name Enter the name of the database server.
The Server Name field is enabled only for Sybase and Informix platforms.

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.

DMS Input Path Enter the Datamover input path.

DMS Output Path Enter the Datamover output path.

DMS Working Path Enter the Datamover working path.

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.

Overriding the Default Execution Option

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

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

See Also

Chapter 8, "Incorporating Best Practices," Use Execution Options, page 112

Configuring Execution Options in PeopleSoft Internet Architecture

This section describes how to define execution options using the Define Execution Options page.

Page Used to Define Execution Options

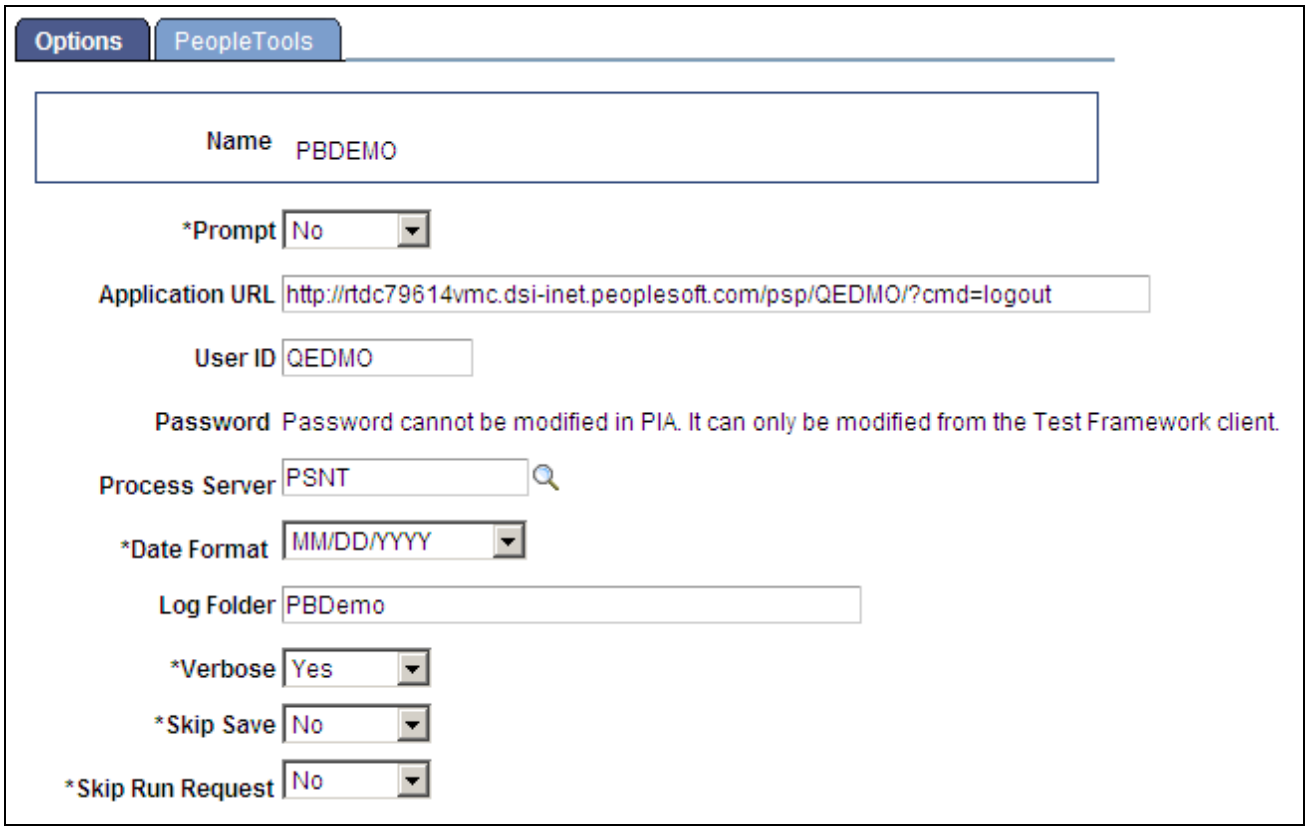
Page Name	*Definition Name*	*Navigation*	*Usage*
Execution Options	PSPTTSTEXEOPTIONS	PeopleTools, Lifecycle Tools, Test Framework, Define Execution Options	Define PTF configuration options.

Configuring Execution Options

You can also define and modify execution options through a PeopleSoft application component. Only a user with the PTF administrator role is able to access the component.

To access Execution Options, select PeopleTools, Lifecyle Tools, Test Framework, Define Execution Options.

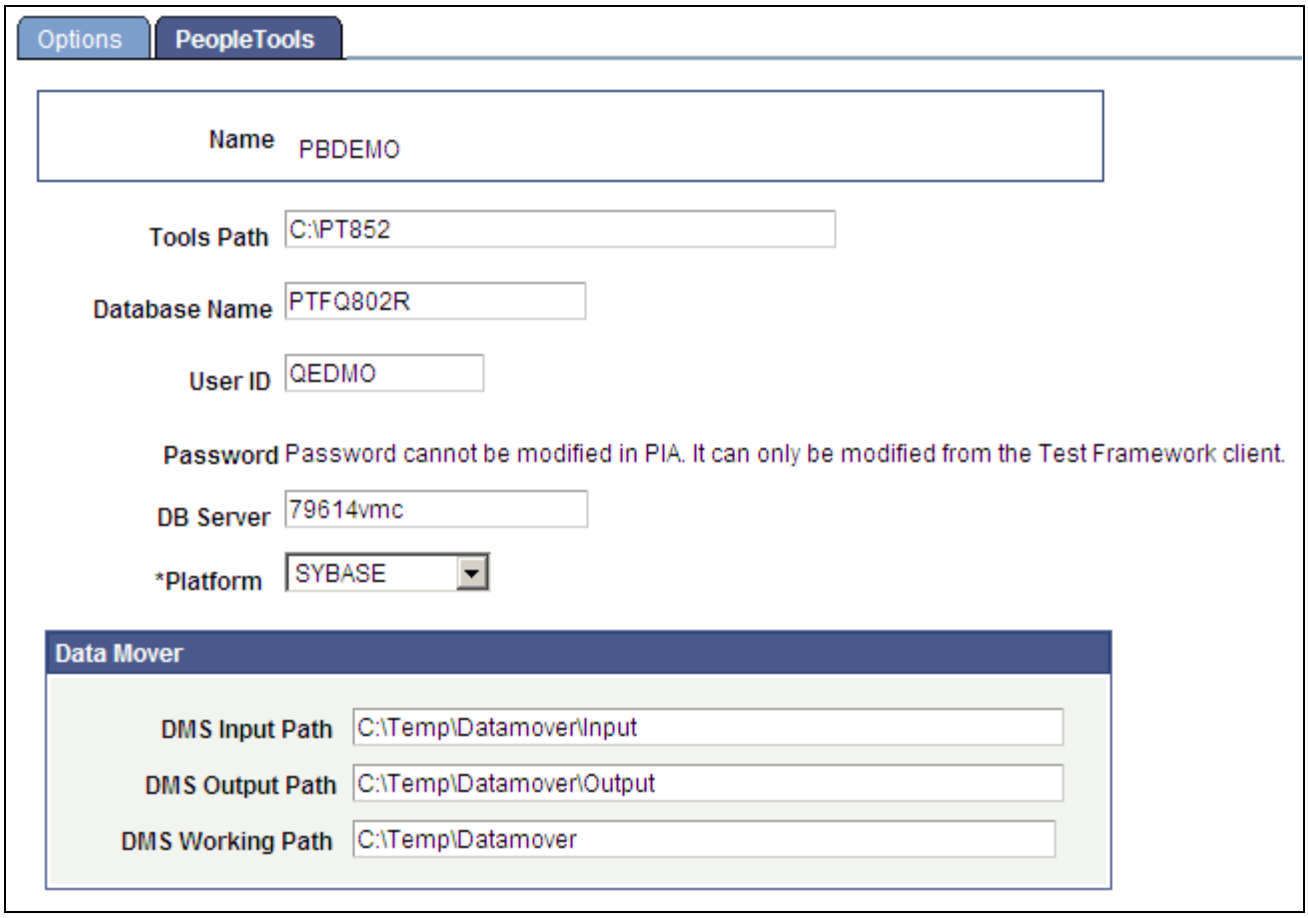
This example shows the Execution Options component - Options page in PIA:



Execution Options component - Options page

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

This example shows the Execution Options component – PeopleTools page:



Execution Options component - PeopleTools page

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

Chapter 3

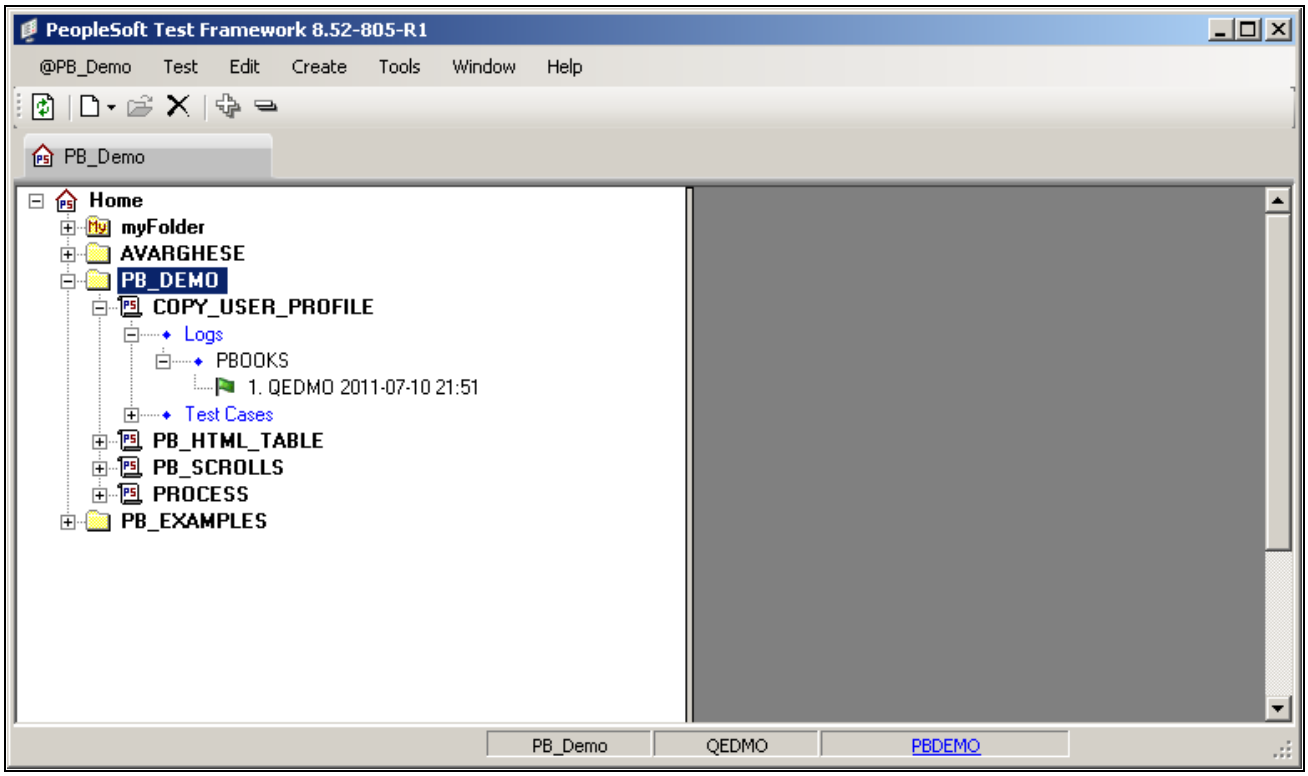
Using PeopleSoft Test Framework

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

- PTF Explorer
- 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 PB_Demo 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.

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 PTF menu commands:

PTF Menu Command	*Usage*
Refresh	Refreshes the current view.

PTF Menu Command	*Usage*
Local Options	Opens the Local Options dialog box.
Execution Options	Opens the Execution Options dialog box.

This table describes the PTF Explorer Test menu commands:

Test Menu Command	*Usage*
Open	Opens the selected test or test case.
Delete	Deletes the selected test.
Refresh Selection	Refreshes the view for the selected test.
Expand Selection	Expands all the branches in the selected node. If Home is selected, the entire tree is expanded
Collapse Selection	Collapses all the branches in the selected node. If Home is selected, the entire tree is collapsed.

The PTF Explorer 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 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.

Tools Menu Command	*Usage*
Log Manager	Opens the Log Manager tool, which enables you to delete logs from tests.

This table describes the PTF Explorer Window menu command:

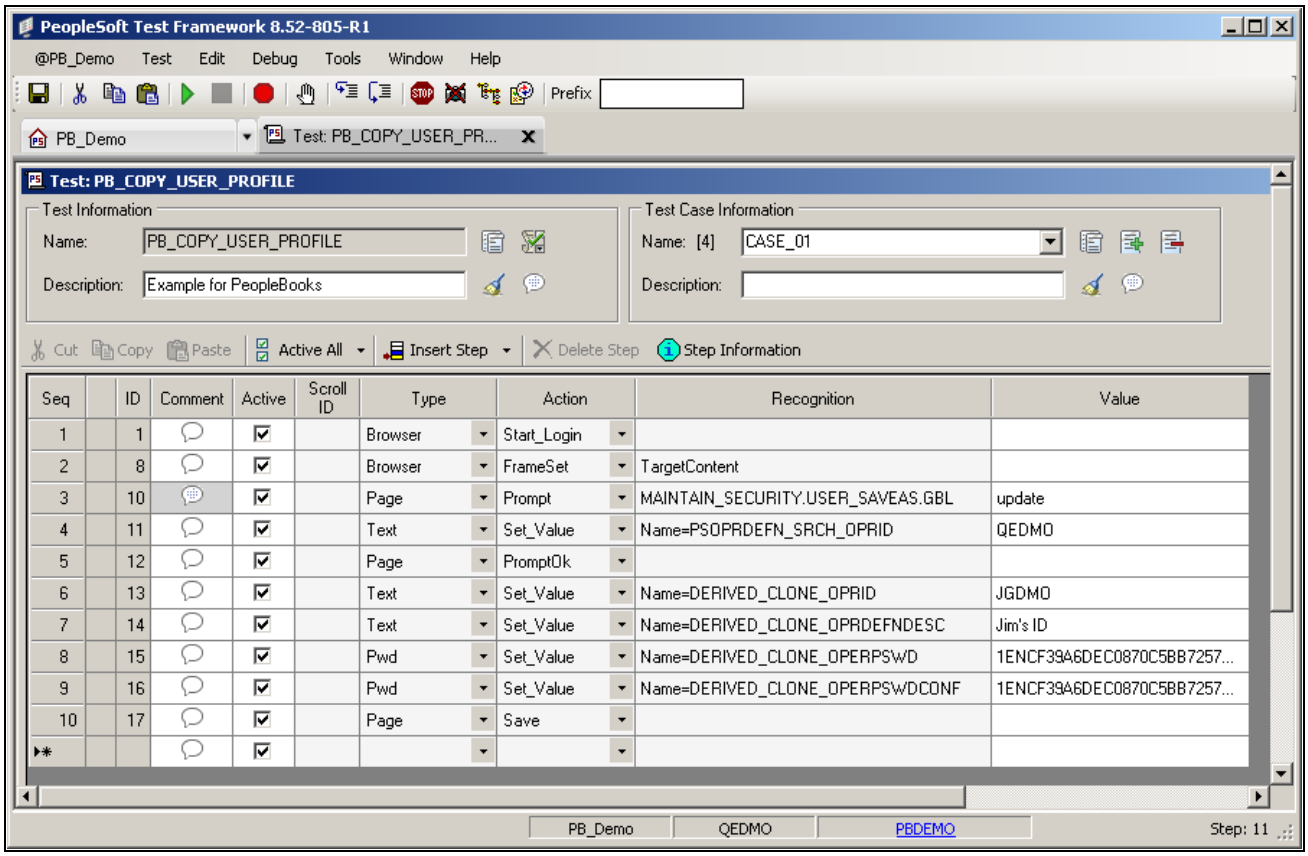
Window Menu Command	*Usage*
Quick Open	Opens a test using data that was copied to the clipboard using the Copy Link to Clipboard command. Using the Copy Link to Clipboard command and the Quick Open command together enables users to easily share tests without having to navigate to the test in PTF Explorer. 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.

Using the Test Editor

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

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

This example shows the PTF Test Editor:



PTF Test Editor

This section discusses:

- Test Editor menus.
- Test Editor field.
- Test Editor toolbar.
- 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 Test Editor PTF menu commands:

PTF Menu Commands	*Usage*
Local Options	Open the Local Options dialog.
Execution Options	Open the Execution Options dialog.

This table describes the Test Editor Test menu commands:

Test 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.
Export	Export test cases and test case values to a character-delimited text file, such as comma-separated values (csv) file.
Import	Import test cases and test case values from a character-delimited text file, such as comma-separated values (csv) file.
Copy Link to Clipboard	Copies a link to the test to the clipboard. You can send this information to another user, who can use the Window, Quick Open feature to open the test without having to navigate to it in PTF Explorer.
Validate Test	Validate the test against the application metadata to identify changes in the metadata since the test was last modified. As you modify a test based on the results of a test maintenance report, you can validate the test by running a test maintenance report. The report is generated for a single test, using analyses of compare reports generated in Step 2 of the Test Maintenance Wizard. See Chapter 7, "Identifying Change Impacts," Creating Test Maintenance Reports, page 86.
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 Test Editor 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 MatchingSteps check box to create a list of matches.
Again	Searches for the Find string again.

This table describes Test Editor Debug menu commands:

Debug Menu Command	*Usage*
Step Into	Executes the current step of the test and advances to the next step.
Step Over	Executes the current step of the test and advances to the next step, unless the next step calls another test. Steps over called tests.
Toggle Break	Sets a break point at the selected step or removes an existing breakpoint.
Clear All Breaks	Removes all break points.
Stop on Error	Stops execution if the test encounters an error.
Disable Screen Shots	By default, the recorder creates a screen shot with each error. Select this option to save space in the log by not creating screen shots.
Highlight Errors	Highlights errors in the log in yellow.
Overwrite Log	If a log window is open after the most recent execution of a test, you can choose whether to append to the open log or overwrite the open log. Select this option to overwrite the open execution log.

This table discusses the Test Editor 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.

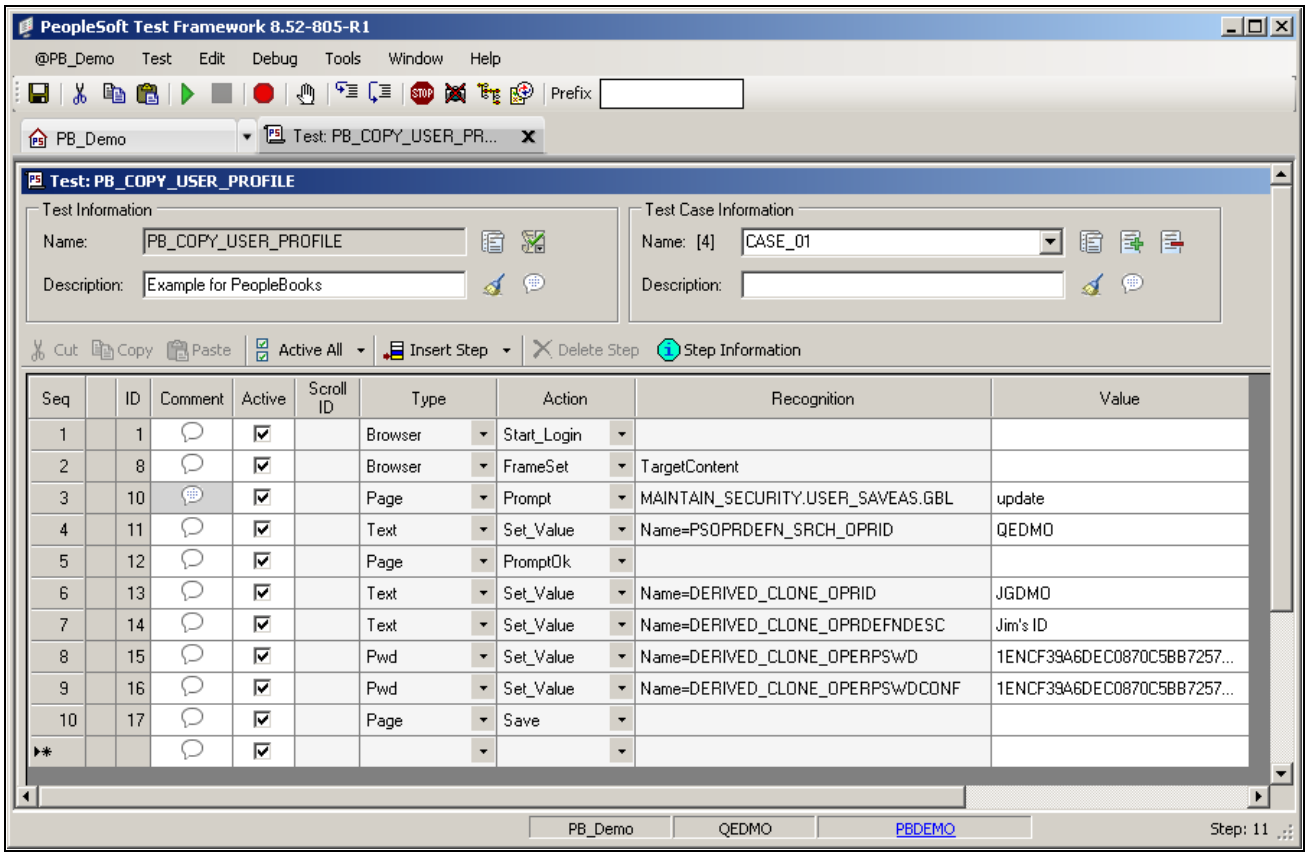
Test Editor Toolbar

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

	Run test.
	End test.
	Show Test Recorder.
	Debug break.
	Step into.
	Step over.
	Stop on error.
	Disable screen shots.
	Highlight errors in the execution log.
	Overwrite the open execution log.
	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:



Example of a Test Editor test window

Test Window Fields

The following fields appear in the test window:

Test Information Fields

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

Description Brief description of the test. You can enter a detailed description in the Comments.

Test Information Functions

 Test Save As.

Click to access the Message Recognition dialog box.

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

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

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

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

See Chapter 5, "Developing and Debugging Tests," Using Library Tests, page 75.

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

Test Case Information Fields

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

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

Test Case Information Functions

Test Case Save As.

Create a new test case.

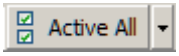
Delete the selected test case.

Open the Test Case Properties dialog.

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

Test Window Toolbar

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

	Selects or deselects the Active check box for all steps.
	Inserts a new, blank step below the current step. Click the drop-down arrow to insert the new step above the current step.
	Deletes the current step.
	Displays the Step Information dialog box showing the PeopleTools metadata associated with the object referenced in the step. Step 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 step in a test. This value does not change when you move, add, or delete a step. Test maintenance reports use the ID value.
Comment	When you click on the comment icon in the comment field a rich text editor opens that enables you to add detailed comments for each step. The comment field is gray when it contains a comment.
Active	Deselect this field to inactivate a step. PTF will skip inactive steps when the test runs. Each step is active by default. This field is grayed for inactive steps.
Scroll ID	This field is only required for scroll handling.
Type	The step type. Often, the step type corresponds to the type of application object the step is to take an action on or to validate, such as Text, Checkbox, Browser, and so on. Other step types perform certain functions, such as executing a test or query, setting a variable, or performing conditional processing.
Action	The action the test is to take . Each step type has a set of associated actions. The two most common actions used with a Text step type, for example, are *Set* and *Verify*.

Recognition	The means that PTF uses to identify the HTML object within the application. Commonly, this is the HTML ID property.
Field Label	Contains the label text of the page control referenced in the Recognition field. The Field Label column appears when the Show Field Label checkbox is selected in Local Options.
Value	In a typical recorded step, this is the value the tester entered for an object. In a step recorded in Verify mode, this would be the value that was present in the object when it was 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 118

Chapter 4, "Creating Tests and Test Cases," Creating Tests, page 49

Using the PTF Test Recorder

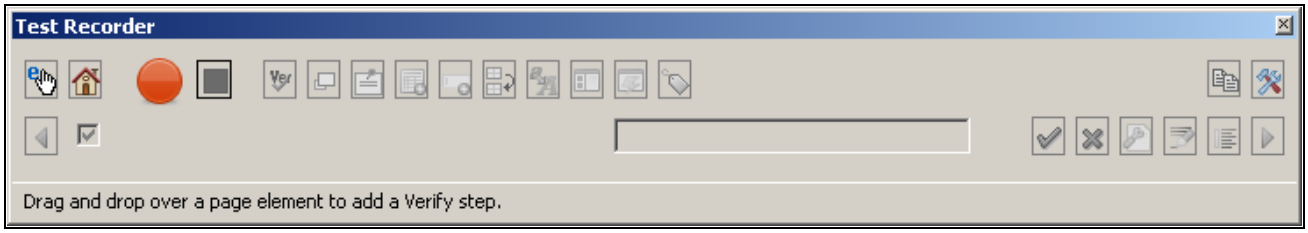
You use the PTF Test Recorder to record the steps in a test. When you record a test, PTF monitors each action you perform in the target application and creates a corresponding step in the test. As you are recording, you can add additional steps, such as Verify, Log, and Conditional, that do not directly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

This section discusses:

- The Test Recorder toolbar
- Recording action tools
- Step modification tools

Test Recorder Toolbar

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



Test Recorder toolbar

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

The Test Recorder toolbar is divided into four areas:

- Recording action tools.
- Step modification tools.
- Tool hints bar.
- Status bar.

Recording Action Tools

This section describes the recording action tools.

Using Drag and Drop

Some recording action tools use drag and drop to get page element recognition data.

During recording, when you drag the icon from the Test Recorder tool bar, your mouse image changes to a bold question mark. As you hover over a page element, the page element highlights. The element that you intend to reference must be highlighted when you drop the icon over it. If the field is not highlighted, PTF will not recognize it. Drag and drop actions are not applicable to all HTML objects.

The Test Recorder recording action tools provide the following functions:

Hook a browser. Drag and drop this icon onto an active PeopleSoft browser session to hook the recorder to that browser. PTF will only hook a browser that was originally launched by PTF. You can drag the hook icon to any portion of the PIA application. No page objects will be highlighted.

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

See Chapter 2, "Installing and Configuring PTF," Configuring Execution Options, page 23.

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

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

Stop recording and write recorded steps to the test.

Pause recording.

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

See Chapter 10, "Test Language Reference," Verify, page 172.

Open field pop ups or organization chart menu items.

See Chapter 10, "Test Language Reference," MouseOver, page 161.

Add a Log step.

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

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Add a Conditional If_Then construct. You are prompted for an expression, such as &Exists=True. The Recorder inserts a Conditional.If_Then step. Record additional steps, then click the icon again to add a Conditional.End_If step.

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

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

See Chapter 10, "Test Language Reference," Variable, page 164 and Chapter 9, "Using the PTF Test Language," System Variables, page 120.

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

See Chapter 5, "Developing and Debugging Tests," Incorporating Scroll Handling, page 71.

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

 Add a Field.GetLabel step. You are prompted for Return Variable.

Step Modification Tools

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

The Test Recorder toolbar step modification icons provide the following functions:

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

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

 Specify whether the step is active or inactive.

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

 Cancel modifications.

 Modify the step.

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

 Edit test step comments.

Recorder Utility Tools

The Test Recorder toolbar utility icons provide the following functions:

 Copy the recording to the clipboard.

Configure recording settings.

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

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

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

Use explicit menu navigation.

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

See Chapter 10, "Test Language Reference," Page, page 147.

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

See Chapter 5, "Developing and Debugging Tests," Handling Application Messages, page 68.

See Also

Chapter 4, "Creating Tests and Test Cases," Recording Tests, page 51

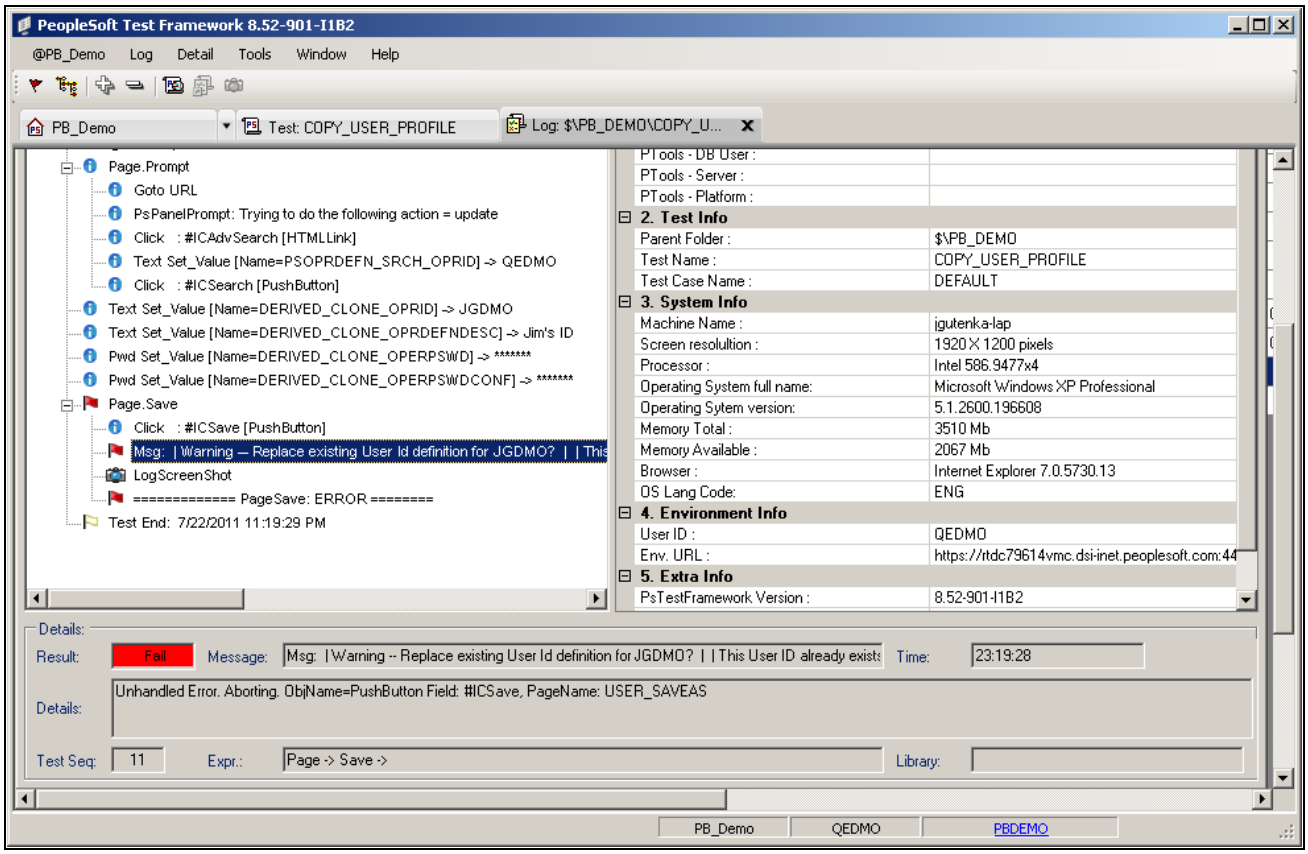
Using the Log Viewer

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:



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***
Find	Find text within a log.
Expand All	Expands all the selections in the log.
Collapse All	Collapses all the selections in the log.

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

This table describes the Detail menu commands:

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

This table describes the Window menu commands:

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

See Also

Chapter 4, "Creating Tests and Test Cases," Reviewing Test Logs, page 60

Chapter 5, "Developing and Debugging Tests," Interpreting Logs, page 70

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.

Note. Folder names must not contain the following characters:

space & ? / \ * < > ' "

Creating a New Test

To create a new test:

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

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

   It is a good practice to provide a good description for a test, such as a description of the product, feature, or business process being tested. Test names are limited in length and provide little information.
5. (Optional) Click the Test Comment to add comments for the test.
6. Select Test, Save or click the Save button in the toolbar.
7. Enter a name for the new test.
8. Click OK.

Naming Tests

These rules apply to test names:

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

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

Copying a Test

To copy a test:

1. Highlight a test in the PTF Explorer.
2. Select Edit, Copy or press CTRL-C to copy the test.
3. Highlight the folder where the copy of the test is to be located.

4. Select Edit, Paste or press CTRL-V to paste the test.
5. Enter a name for the new test.

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

Recording Tests

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

PTF Recorder populates these fields for each step:

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

This is an example of test steps:

Seq		ID	Comment	Active	Scroll ID	Type	Action	Recognition	Value
1		1		☑		Browser	Start_Login		
2		2		☐		Link	Click	innerText=Sign in to PeopleSoft	
3		3		☐		Text	Set_Value	Name=userid	QEDMO
4		4		☐		Pwd	Set_Value	Name=pwd	1ENC41B9FB38D060CB90E0108BF840409B50B0309B...
5		5		☐		Button	Click	Name=Submit	
6		6		☐		Link	Click	id=fldra_PT_PEOPLETOOLS	
7		7		☐		Link	Click	id=fldra_PT_SECURITY	
8		8		☑		Browser	FrameSet	TargetContent	
9		9		☐		Link	Click	Name=defaultlinnerText=Copy User Profiles	
10		10		☑		Page	Prompt	MAINTAIN_SECURITY.USER_SAVEAS.GBL	update
11		11		☑		Text	Set_Value	Name=PSOPRDEFN_SRCH_OPRID	QEDMO
12		12		☑		Page	PromptOk		
13		13		☑		Text	Set_Value	Name=DERIVED_CLONE_OPRID	JGDMO
14		14		☑		Text	Set_Value	Name=DERIVED_CLONE_OPRDEFNDESC	Jim's ID
15		15		☑		Pwd	Set_Value	Name=DERIVED_CLONE_OPERPSWD	1ENCF39A6DEC0870C5BB7257ADF0801CC5579ACB0...
16		16		☑		Pwd	Set_Value	Name=DERIVED_CLONE_OPERPSWDCONF	1ENCF39A6DEC0870C5BB7257ADF0801CC5579ACB0...
17		17		☑		Page	Save		
*				☑					

Example of test steps

Seq and ID are system-generated fields.

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

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

See Chapter 3, "Using PeopleSoft Test Framework," Test Step Fields, page 41.

Test Action Tools

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

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

See Also

Chapter 3, "Using PeopleSoft Test Framework," Using the PTF Test Recorder, page 42

Chapter 9, "Using the PTF Test Language," PTF Test Language, page 118

Recording a Test

To record a test:

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

   See Chapter 3, "Using PeopleSoft Test Framework," Using the PTF Test Recorder, page 42.
5. *Hook* a browser, that is, associate a PeopleSoft application browser with the test.

   To hook a browser, you can either:

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

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

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

See Also

Chapter 3, "Using PeopleSoft Test Framework," Using the PTF Test Recorder, page 42

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

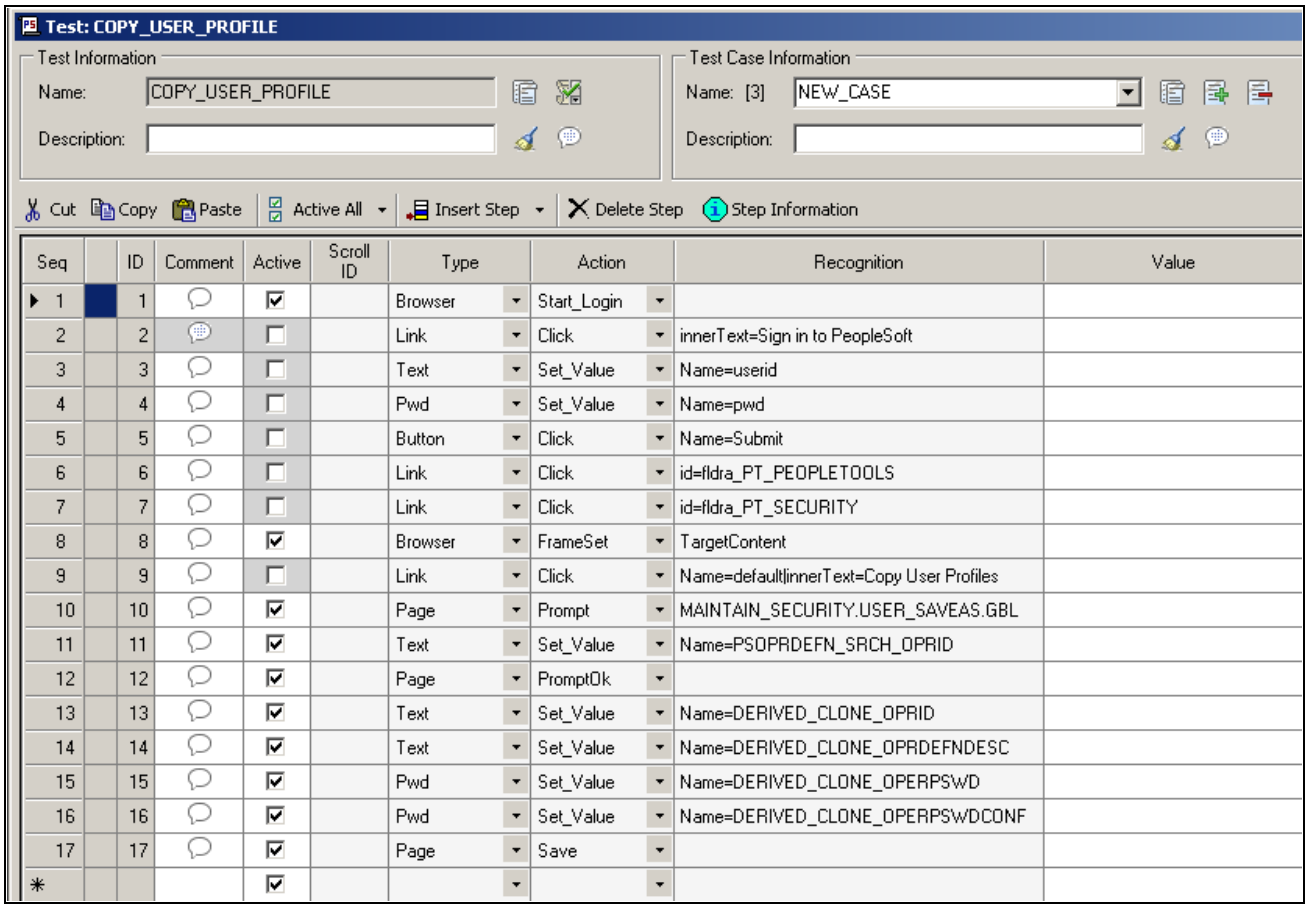
To create a new test case:

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 Test Case Propertiesicon to enter a long description of the test case.

   It is a good practice to provide a good description for a test case. Test names are limited in length and provide the user with little information.
5. (Optional) Click the Test Case Comment icon to add a comment for the test case.
6. The new test case provides a set of new, empty Value fields for all the steps of the test.

   Enter a new value for each step that requires a value.
7. Save the test case.

This example shows a new test case with blank values:



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

Exporting and Importing Test Cases

You can export and import test cases using character-delimited text files to quickly create new test cases.

To export and import test cases:

1. With a test open, select Test, Export.
2. Specify a location for the file and select a separator character, such as comma.
3. Specify whether to export one or all test cases.
4. Click Start.
5. Modify the character-delimited file to modify or add test cases.
6. In PTF, with a test open, select Test, Import.
7. Enter the file path or browse to the file.
8. Specify the separator character.
9. Specify whether to ignore or replace existing cases.
10. Click Start.

Executing Tests

This section describes how to execute tests and test cases.

Executing a Test

To execute a test:

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

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

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

   Select an execution option from the list and click Accept.

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

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

Note. When PTF executes a test it disables Num Lock and Caps Lock on your keyboard and restores them when the test completes. If the execution terminates abnormally, the test does not complete, or hangs up, and PTF does not restore Num Lock and Caps Lock. Select Test, End or click the End icon in the toolbar to end the test and restore Num Lock and Caps Lock.

See Also

Chapter 4, "Creating Tests and Test Cases," Reviewing Test Logs, page 60

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 60

Executing a Test from the Command Line

You can also execute a test from the command line.

Syntax

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

```
PsTestFw -CD=ConnectionName -CP=ConnectionPassword -TST=TestName
-TC=TestCaseName [-PFX=Prefix] -EXO=ExecutionOption [-LOG=LogFileName]
```

Use this syntax to explicitly specify connection parameters:

```
PsTestFw  -CS=Server -CNO=NodeName -PS=ProxyServer -PU=ProxyUser -PP=ProxyPassword⇒
 -CO=UserName  -CP=ConnectionPassword -TST=TestName
-TC=TestCaseName [-PFX=Prefix] -EXO=ExecutionOption [-LOG=LogFileName]
```

Parameters

Parameter	Description
–CD=	Specify the name of the environment signon to use for connection. This is the Database Name you would select in the PeopleSoft Test Framework Signon dialog box when signing on to PTF. The environment signon settings are stored in the environments.xml file in the PTF data directory (`C:\Documents and Settings\<User>\Application Data\PeopleSoft\PeopleSoft Test Framework` by default). If the environment connection data is not set in the environments.xml file, then you can explicitly specify the connection parameters. See the following table for a description of connection parameters. See Chapter 2, "Installing and Configuring PTF," Creating a Connection to a PTF Environment, page 16.
–CP=	Specify the user password.
–TST=	Specify the test name.
–TC=	Specify the test case name.
–PFX=	(Optional) Specify the prefix. See Chapter 10, "Test Language Reference," #PREFIX#, page 179.
–EXO=	Specify the execution option to be used in the execution. **Note.** The name of the execution option must not contain the following characters: space & ? / \ * < > ' "
–LOG=	(Optional) Specify the name for the log. The default is unattended.log.

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

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

Example

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

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

The following example explicitly sets connection parameters:

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

Log File

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

If the log file exists it will be overwritten.

This is an example of a log file for a passed execution:

```
<execution>
  <Started>2010-07-27 16:49:47</Started>
  <Param>
    <Database>QE851</Database>
    <TestName>TEST_CMD_LINE</TestName>
    <TestCase>DEFAULT</TestCase>
    <Prefix>01</Prefix>
    <ExecOpt>QE851-Folder</ExecOpt>
  </Param>
  <Status>Passed</Status>
  <Test>
    <LogFolder>Folder</LogFolder>
    <LogName>48. VP1 2010-07-27 16:49</LogName>
  </Test>
</execution>
```

This is an example of a log file for a failed execution:

```
<execution>
  <Started>2010-07-27 16:47:07</Started>
  <Param>
    <Database>QE851</Database>
    <TestName>TEST_CMD_LINE</TestName>
    <TestCase>DEFAULT</TestCase>
    <Prefix>01</Prefix>
    <ExecOpt>QE851-Folder</ExecOpt>
  </Param>
  <Status>Not Completed-FatalError</Status>
  <Test>
    <LogFolder>Folder</LogFolder>
    <LogName>46. VP1 2010-07-27 16:47</LogName>
    <Message>Browser not initialized.</Message>
  </Test>
</execution>
```

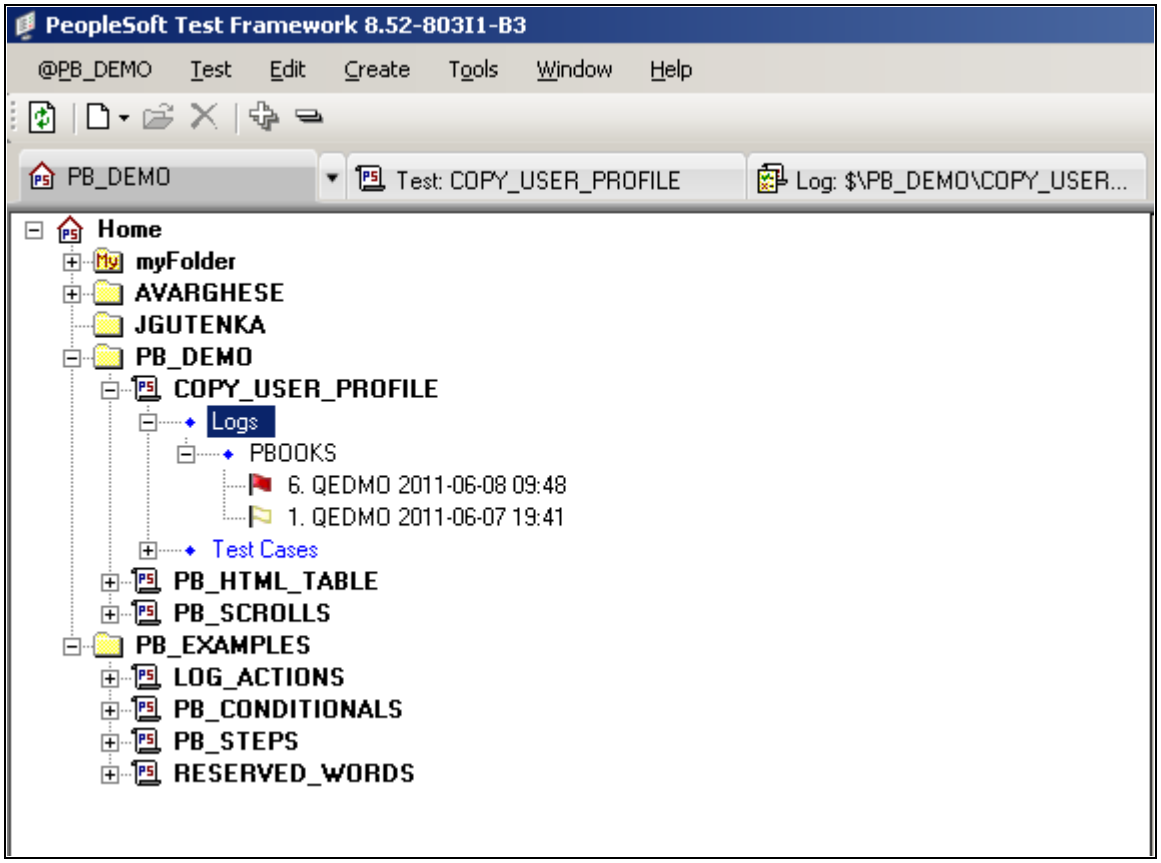
Reviewing Test Logs

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:

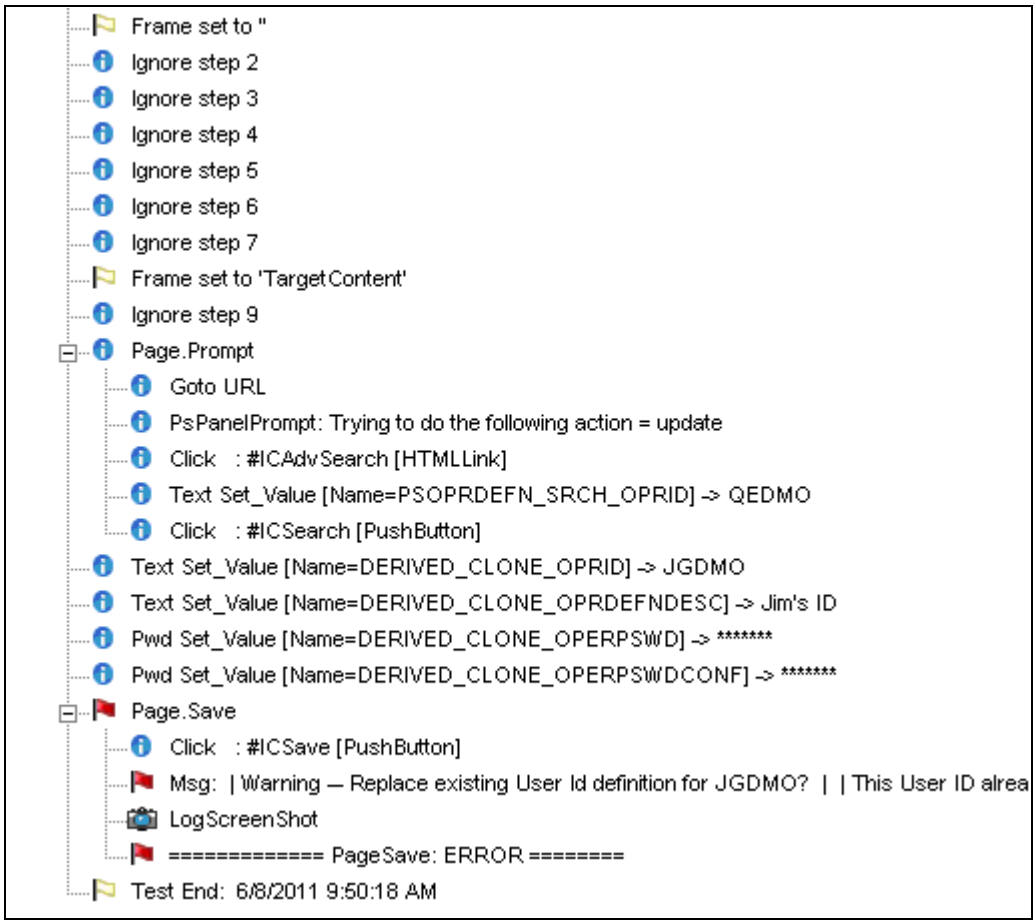


Example of PTF Explorer showing log entries in the Logs folder

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

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

This example shows log entries in a test log:



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:

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, or the test encountered an error while PTF was configured to stop on all errors.

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

See Also

Chapter 5, "Developing and Debugging Tests," Interpreting Logs, page 70

Chapter 3, "Using PeopleSoft Test Framework," Using the Log Viewer, page 46

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 might 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:

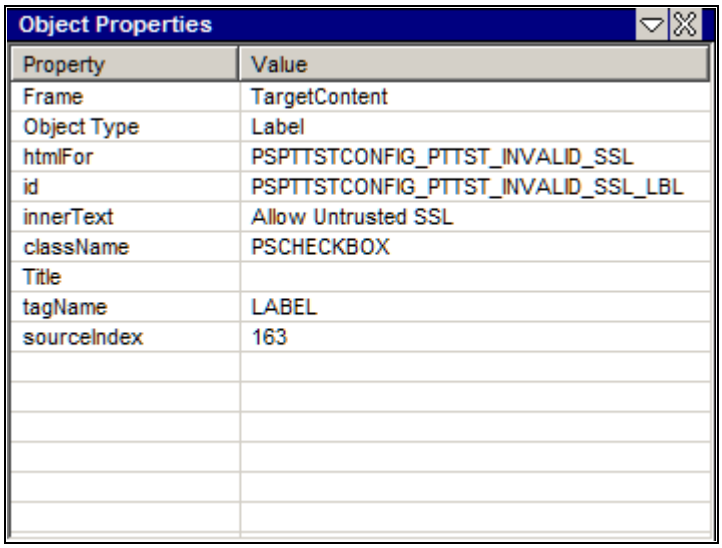
Message tool ID values

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:



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:

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 in PTF 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:

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:

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:

Example of a step that verifies a value

See Also

Chapter 4, "Creating Tests and Test Cases," Creating Test Cases, page 54

Chapter 10, "Test Language Reference," Reserved Words, page 173

Using Variables

This section discusses how to use variables.

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

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

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

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

- Use the variable in the Recognition field of a Conditional.If_Then step or 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 119

Chapter 10, "Test Language Reference," Variable, page 164

Chapter 10, "Test Language Reference," Conditional, page 129

Using Conditional Logic

This section discusses how to use conditional logic.

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

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

```
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 129

Handling Application Messages

This section discusses how to handle application messages.

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

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

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

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

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

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

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

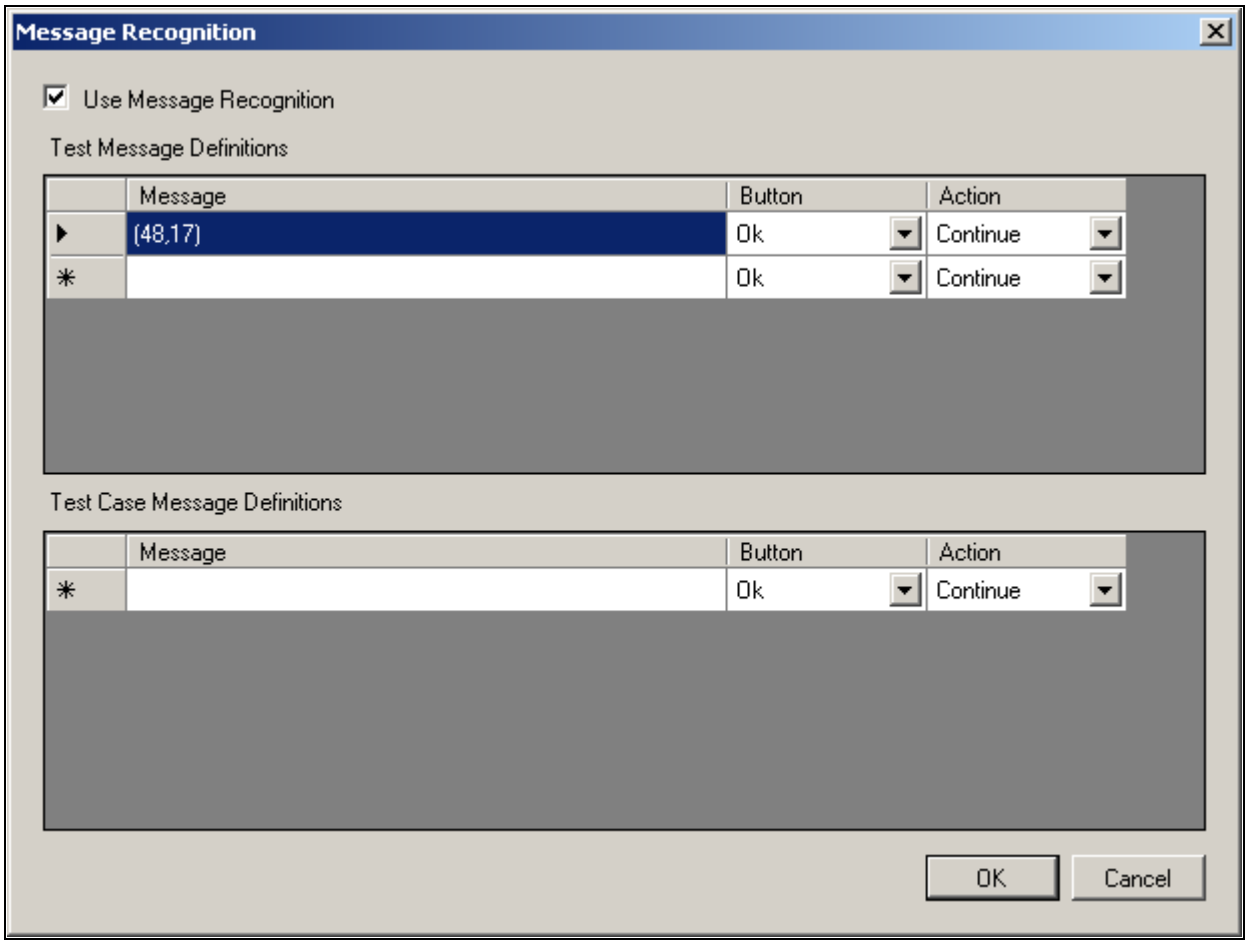
To create a message definition, access the Message Recognition dialog box.

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

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

   You can define message definitions at either the test or test case level. Message definitions defined at the test case level take precedence over message definitions defined at the test level.
3. Select which button the step should click.
4. Specify what action PTF should take after the button is clicked.
5. To delete a message, select the message line and press the Delete key.

This example shows the Message Recognition dialog box:



Message Recognition dialog box

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

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

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

Valid values are:

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

Action Select the action that PTF will take.

Valid actions are:

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

Interpreting Logs

This section discusses how to interpret logs.

This table lists and describes common log messages:

Status	***Message***	***Description***
Fail	Access is denied. Please check browser settings.	PTF is not able to read information from your browser. Check browser settings to make sure your browser is configured properly. See Chapter 2, "Installing and Configuring PTF," Configuring the Browser Security Settings, page 14.
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 14. 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 Recognition field of the step. See Chapter 5, "Developing and Debugging Tests," Using the Message Tool, page 63. • 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 83.

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.

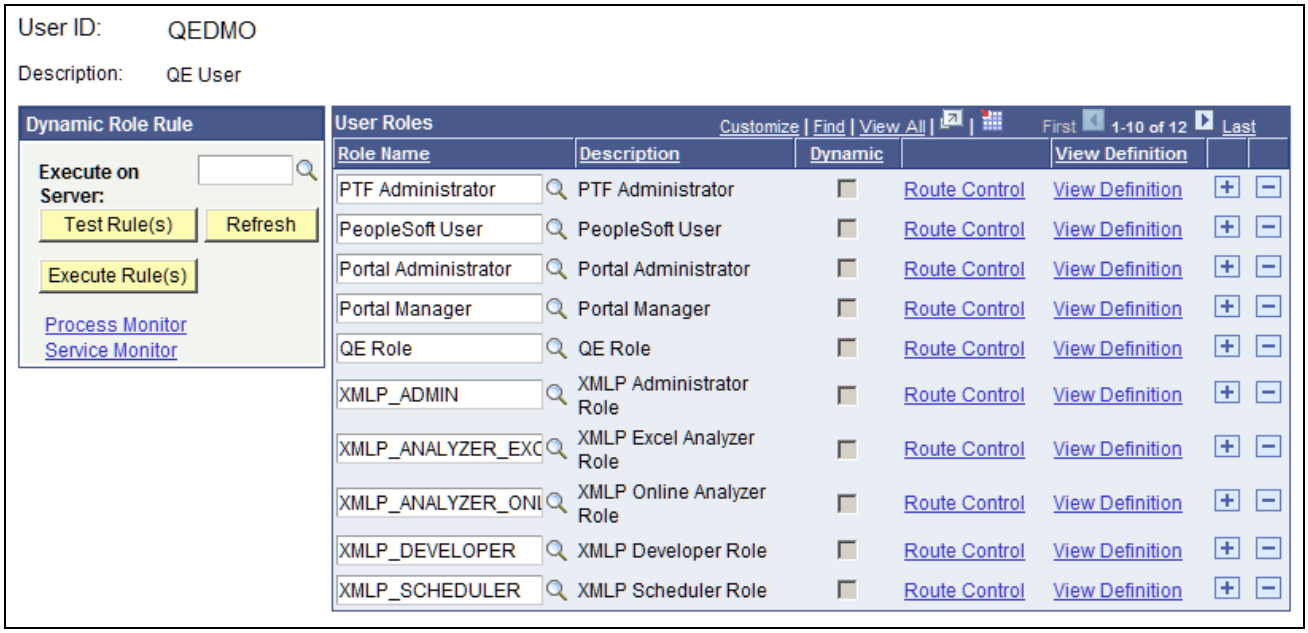
A scroll can be implemented as a scroll area or a grid. In scroll areas, the fields appear on the page in a freeform manner. In grids, fields appear as columns similar to those on a spreadsheet. Individual rows of data within a scroll or grid are uniquely identified by a set of one or more fields, or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

Scroll ID	Type	Action	Recognition	Value
	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 found the key value or searched all rows on all pages of the scroll.

Follow these steps to use a dynamic position index variable:

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

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.

Note. You can create Key_Set steps and Action steps during recording.

See Chapter 3, "Using PeopleSoft Test Framework," Using the PTF Test Recorder, page 42.

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

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

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

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 row 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 155

Calling Tests

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

- Use library tests.
- Use shell tests.
- Use parameters with library tests.
- Share test assets.

Understanding Calling Tests

If a test uses a sequence of steps repeatedly, you may want to isolate the repetitive sequence of steps and move them to another, smaller test. Doing so enables you to call the steps repeatedly and also make them available to other tests that use the same sequence of steps.

Moving shared test steps to a distinct test (a called, or child, test) provides these benefits:

- Reduces the amount of recording or development you need to do.
- Reduces the amount of debugging you need to do.
- Reduces the effects of development changes that require manual updates to existing tests.

You can use the Test.Exec action to call any other test, but you may be able to manage and identify the relationships between calling and called tests more easily on the PTF Explorer tree if you use library tests and shell tests.

Using Library Tests

A library test cannot be executed by itself. It must be called by another test.

To make a test a library test, select the Library Test check box in the Test Editor.

Complete these steps to create a library test from an existing test:

1. Open an existing test.
2. Identify repetitive steps within the test, copy them from the existing test, and paste them into a new test.
3. Select the Library Test check box on the new test.
4. Save the library test.

5. Remove the repetitive steps from the original test and replace them with a step that uses a Test.Exec action to call the new test.
6. Save the original test.

Using Parameters with Library Tests

Parameters enable you to pass dynamic values from a calling test to a library test.

To use parameters:

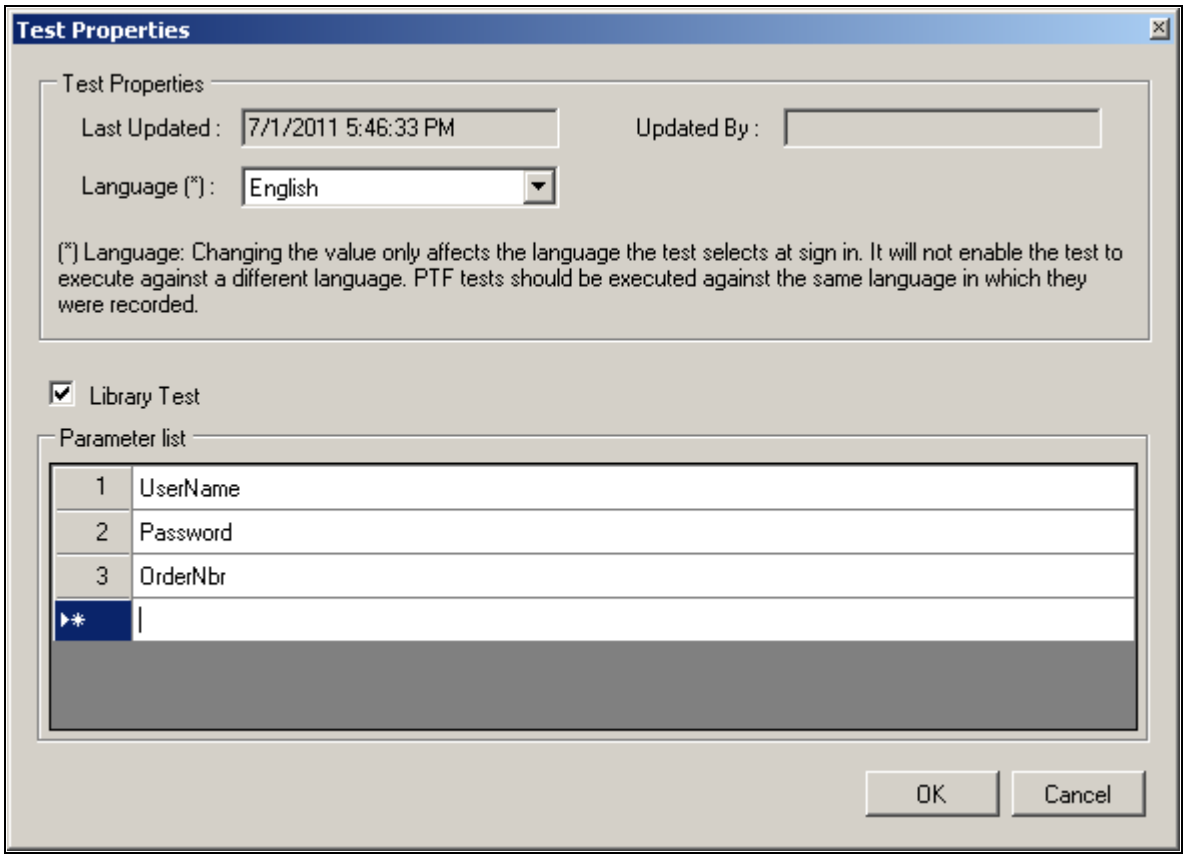
1. Define the parameters in the library test.
2. Assign values to the parameters in the calling test.
3. Use the parameters as you would text strings in the library test.

To define parameters in a library test:

1. Select the Library Test check box in the Test Properties dialog box.
2. Enter the names of parameters in the Parameter List.

   Parameter names can be up to 254 characters, and can contain only letters, numbers, and underscores.
3. Click OK to dismiss the dialog box.

The following example shows a parameter list:



Test Properties dialog box with a parameter list

In the calling test, place parameter assignments in Test.Exec steps in the Recognition field following the test name, separated by semicolons, with no spaces. You can place multiple parameter assignments, separated by semicolons. The last semicolon is optional.

Note. The semicolons must not be followed by spaces.

The following example shows a Test.Exec step that calls a library test and passes three parameters.

Test	Exec	**PB_PARMS_LIB;UserName=QEDMO;Password=QEDMO;OrderNbr=1001**	DEFAULT

Test.Exec step with parameters

In the library test, place a %param.parameter_name% construct wherever you want to use a parameter. The parameter is replaced at runtime by the text assigned in the calling test.

Text	Set_Value	Name=userid	%param.UserName%
Pwd	Set_Value	Name=pwd	%param.Password%
Variable	Set_Value	&OrderNbr=%param.OrderNbr%	

Library test steps with parameters

Using Shell Tests

To create a shell test, while in PTF Explorer select Create, Shell Test.

A shell test is a type of test that is meant to be used primarily to call other tests. For this reason, a shell test only supports these actions:

- Execution actions - modify the behavior of tests during execution. Execution actions include Skip_PageSave, Skip_RunRequest, Skip_Login, and StopOnError.
- Test.Exec - calling other tests. Test.Exec enables you to call multiple test cases with a test and, using the #Ignore reserved word, skip the test call for certain test cases.
- DataMover.Exec - calling data mover scripts
- Query.Exec - running queries
- Log.Message and Log.Screenshot.
- Variable.Set_Value - manipulating variables

Organizing component tests into shells enables you to identify large business-process oriented type tests (that is, the type that cross multiple components and online activities). The steps available in shell tests are intentionally limited in order to represent high-level business process flows through called test routines.

PTF variables are global, so you can set a variable in the shell test and use it in the called tests, or you can set a variable in a test and use it in the shell test or other called tests.

Sharing Test Assets

Often, test developers, application developers, testers, and others collaborate to develop tests. PTF enables you to send links to tests, test cases, and logs to other users, saving them from having to navigate through PTF Explorer to locate them.

To share the location of a test asset:

1. Copy the link to the clipboard.
   - In PTF Explorer, highlight the name of a test asset and select Edit, Copy Link to Clipboard.
   - In the Log Viewer, with a log open, select Log, Copy Link to Clipboard.
2. Paste the link text into a message or document to send to another user.
3. The recipient copies the text and selects Window, Quick Open.

   The Quick Open feature is available in PTF Explorer, Test Editor, and Log Viewer.

   The system automatically copies the link for the asset to the Quick Open dialog box.
4. The recipient clicks OK to open the asset.

You can also use Copy Link to Clipboard with the Quick Open feature to locate a folder in PTF Explorer.

Chapter 6

Administering PTF

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

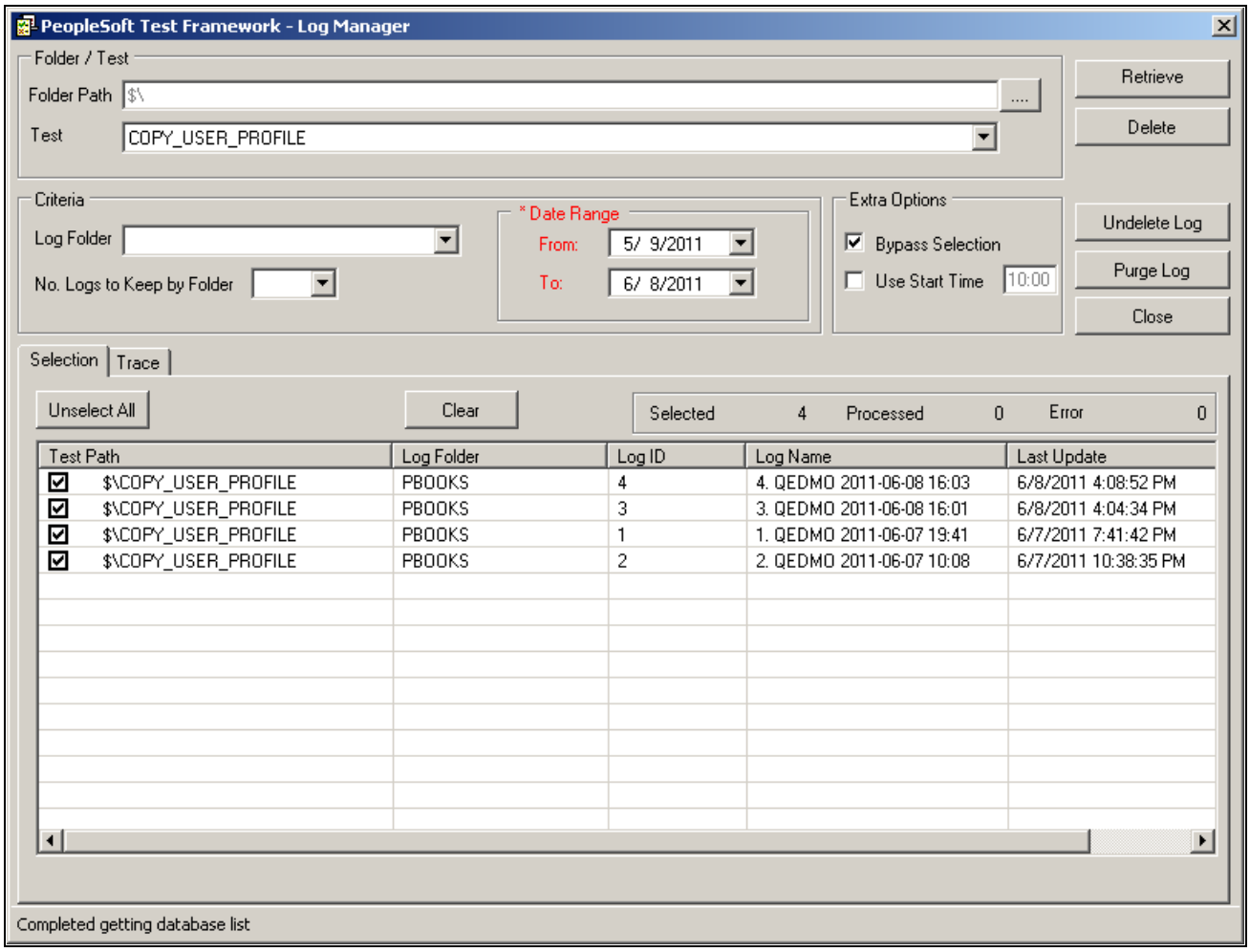
Over time, as you run tests, you will create a number of test logs. Because they reside in your application database, test logs, especially those containing many screen shots, can affect the storage demands on your database. PTF Log Manager enables you to minimize this demand and remove clutter from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

To help you decide which logs to remove from your database, Log Manager lists log entries from the test environment based on the criteria you specify. If all the fields are empty, then Log Manger lists all the logs in an environment that were created within the specified date range. The default date range is the current date.

To access the Log Manager, select Select 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 logs associated with the selected test. The drop-down list is restricted to the tests in the folder specified in Folder Path.

Criteria

Log Folder	Select a log folder. Log folders associated with the selected test are available in the list.
No. Logs to Keep by Folder (number of logs to keep by folder)	Select the number of logs to remain in the folder. For instance, if the folder contains six logs and you specify three, then the three newest logs will be retained (that is, they will not be included in the list as candidates for deletion) and the three oldest logs will be retrieved.
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 tests and test cases are PeopleTools managed objects, they can be copied from one database to another in the same way as other PeopleTools objects, such as record definitions, SQL definitions, and PeopleCode programs.

You can create a project in 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 8.52: 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 to another page, or renamed, any step that references that field will fail. Test developers must identify and update every step in each test that is affected by the change.

One way to identify the effects on tests is to run each test against the new application and note where the test fails. This manual process is time-consuming, expensive, and prone to errors. It also fails to identify those areas in the new application that are not covered by existing tests.

Because PeopleSoft Test Framework (PTF) test assets are PeopleTools metadata and because PTF tests incorporate references to PeopleTools metadata—that is, menus, components, pages, records, and fields—PTF is able to automate the process of correlating metadata changes with existing tests.

PTF delivers two tools that help test developers to determine the effect of changes:

- Test maintenance reports

  A test maintenance report correlates PeopleTools compare report data with PTF test metadata to identify certain changes to menus, components, pages, records, and fields that may impact the PTF tests.

- Test coverage reports

  A test coverage report correlates PeopleTools project data with PTF test metadata to identify menus, components, pages, records, and fields that are referenced in PTF tests.

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

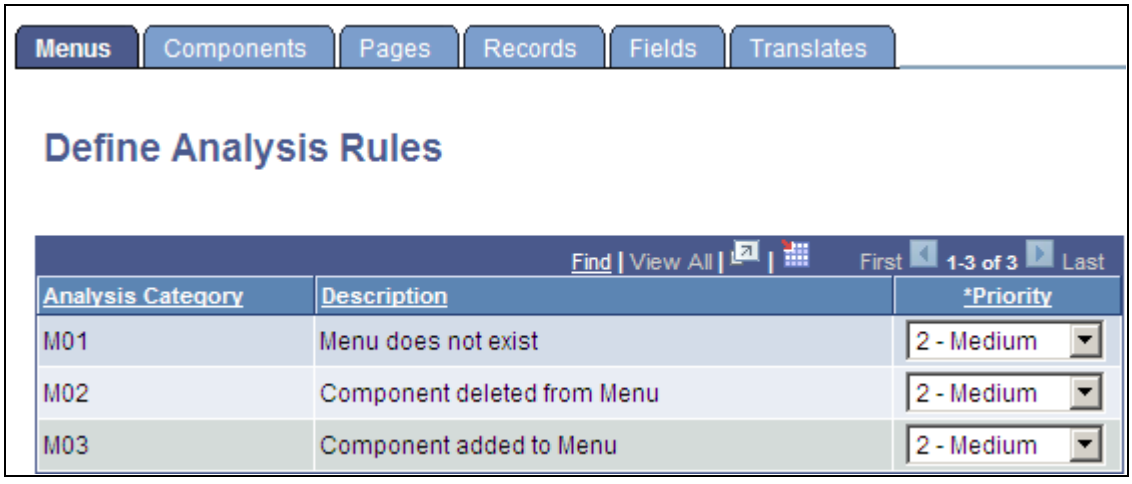
This section discusses the page used to define analysis rules.

Page Used to Define Analysis Rules

Page Name	*Definition Name*	*Navigation*	*Usage*
Analysis Rules	PSPTTSTANLMENU	PeopleTools, Lifecycle Tools, Test Framework, Define Analysis Rules	Define the priority for each of the attribute checks in a test maintenance report.

Defining Analysis Rules

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



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 analyzed.

The following table describes the analysis categories:

Analysis Category	***Description***
M01	Menu does not exist
M02	Component deleted from Menu
M03	Component added to Menu
C01	Page deleted from Component
C02	Page added to Component
C03	Search Record changed on Component
P01	Field deleted on Page
P02	Required Field added to Page
P03	Fieldname changed on Page
P04	Field Type changed on Page
P05	Field Label changed on Page
P06	Non-Required Field added to Page
P07	Recname changed on Page
P08	Recname & Fieldname changed on Page
R01	RecordField now required on Page
R02	RecordField no longer required on Page
R03	RecordField is now a Search Key
R04	RecordField no longer a Search Key
R05	RecordField is now a List Box Item

Analysis Category	*Description*
R06	RecordField no longer a List Box Item
F01	Field Type changed
F02	Field Length changed
F03	Field Format changed
F04	Field Decimal Positions changed
X01	Translate Value does not exist
X02	Translate Value added

Creating Test Maintenance Reports

This section discusses how to create test maintenance reports using the Create Test Maintenance Report wizard.

Access the Create Test Maintenance Report wizard (PeopleTools, Lifecycle Tools, Test Framework, Create Test Maintenance Report).

The wizard consists of three steps:

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

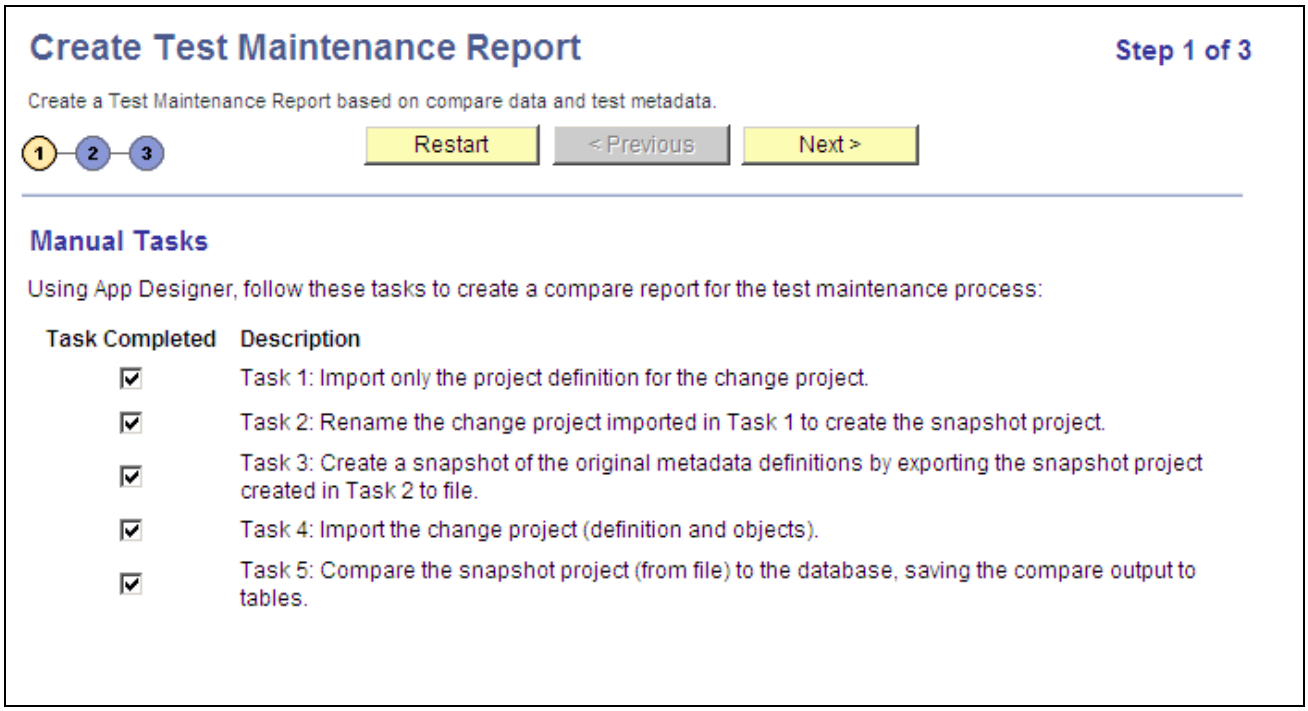
Step 1 of 3: Manual Tasks

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 you exit the wizard and then return to the wizard again, the wizard sends you to the last visited page.

This tracking is done according to user ID. This means that if two users share the same user ID, the second user might not enter the first page when accessing the wizard. The wizard will take the second user to where the previous user left the wizard.

If two users with different user IDs work on the same project, the wizard does not track the state for the second user. For instance, if the first user completed the tasks for Step 1, the second user needs to check the five tasks on Step 1 to bypass that step.

Task 1: Import only the project definition for the change project.

Suppose you are about to apply a change project named SA_PROJ_ UPGD. Before the change project is applied, import the change project, SA_PROJ_ UPGD, as a project definition only.

1. In Application Designer, select Tools, Copy Project, From File.
2. In the selection box, highlight the change project, *SA_PROJ_UPGD*.
3. Click Select.

   The Copy From File dialog box appears.

4. Click the Deselect All button to deselect all of the definition types so that only the empty project structure is imported.

   At this point, you are importing only the names of the definitions into the project. None of the actual definitions in the upgrade project are copied.

   If a definition exists in the original database with the same name as a definition in the upgrade project, the upgrade project in the database will (correctly) contain the original, unmodified definition of the object.

5. Click Copy.

Task 2: Rename the change project imported in Task 1 to create the snapshot project.

Create a copy of the change project, SA_PROJ_UPGD.

Select File, Save Project As and name the new project SA_PROJ_SNAP.

Creating the Snapshot Project is required to avoid naming conflicts later in the process.

Task 3: Create a snapshot of the original metadata definitions by exporting the snapshot project created in Task 2 to file.

Export the snapshot project, SA_PROJ_SNAP, to file:

1. Select Tools, Copy Project to File.
2. If this project is large, as upgrades often are, you can save time during the compare process by deselecting all definitions in the Copy Options window, except for the five definitions that are referenced by PTF tests:

   - Menus
   - Components
   - Pages
   - Records
   - Fields

3. Accept the defaults and click OK.

Task 4: Import the change project (definition and objects).

Import the original change project, SA_PROJ_UPGD, from file:

1. Select Tools, Copy Project From File.
2. Include all the definition types.

At this point, the database contains the new metadata.

Task 5: Compare the snapshot project (from file) to the database, saving the compare output to tables.

Compare the current (target) database with the pre-change (source) project, SA_PROJ_SNAP:

1. Select Tools, Compare and Report, From File.
2. Highlight the snapshot project, SA_PROJ_SNAP, and click the Select button.

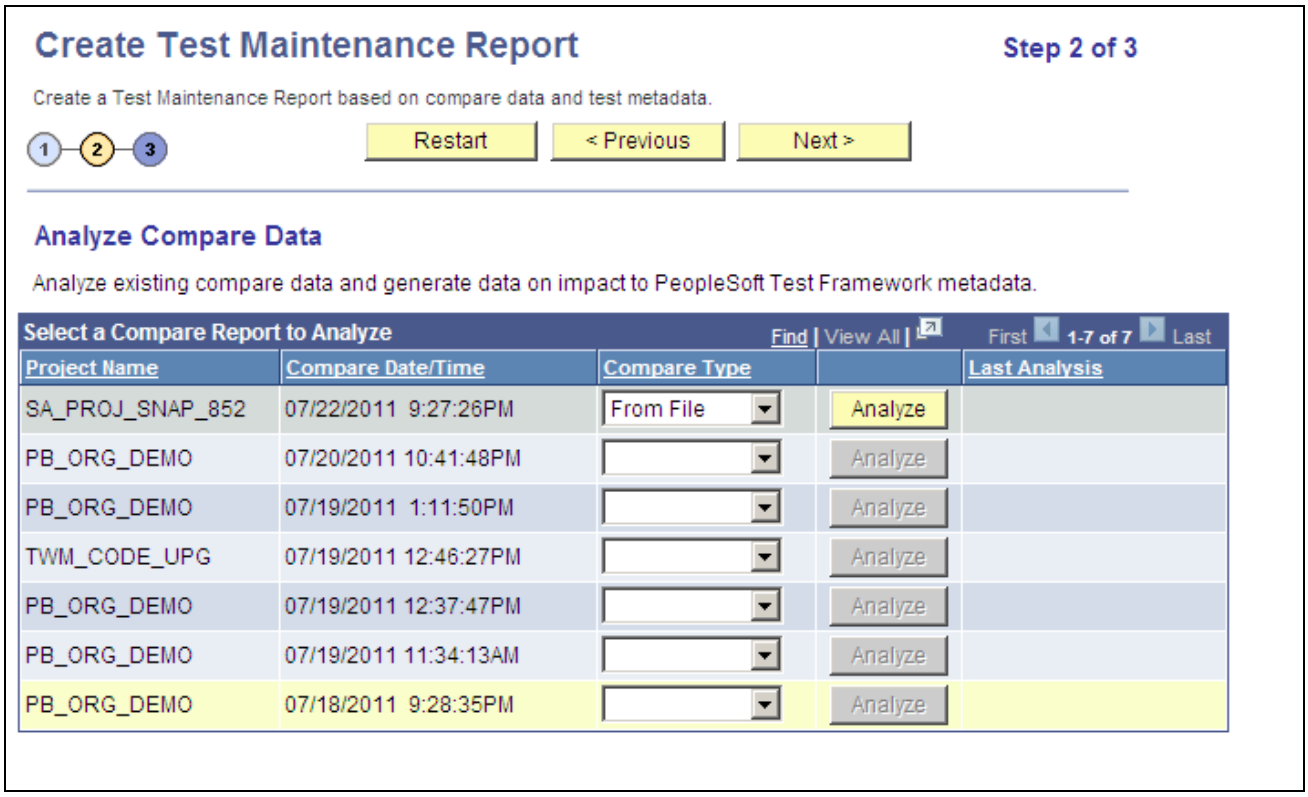
   The Replace existing SA_PROJ_SNAP? dialog box appears.
3. Click Yes.

   The Compare and Report From File dialog box appears.
4. Click Options to access the Upgrade Options dialog box.
5. Select the Report Options tab.
6. Select the Generate Output to Tables check box.
7. Select the Report Filter tab.
8. Click Select All.
9. Click OK.
10. Click Compare.

The wizard will use data from this compare report to create the test maintenance report.

Step 2 of 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.

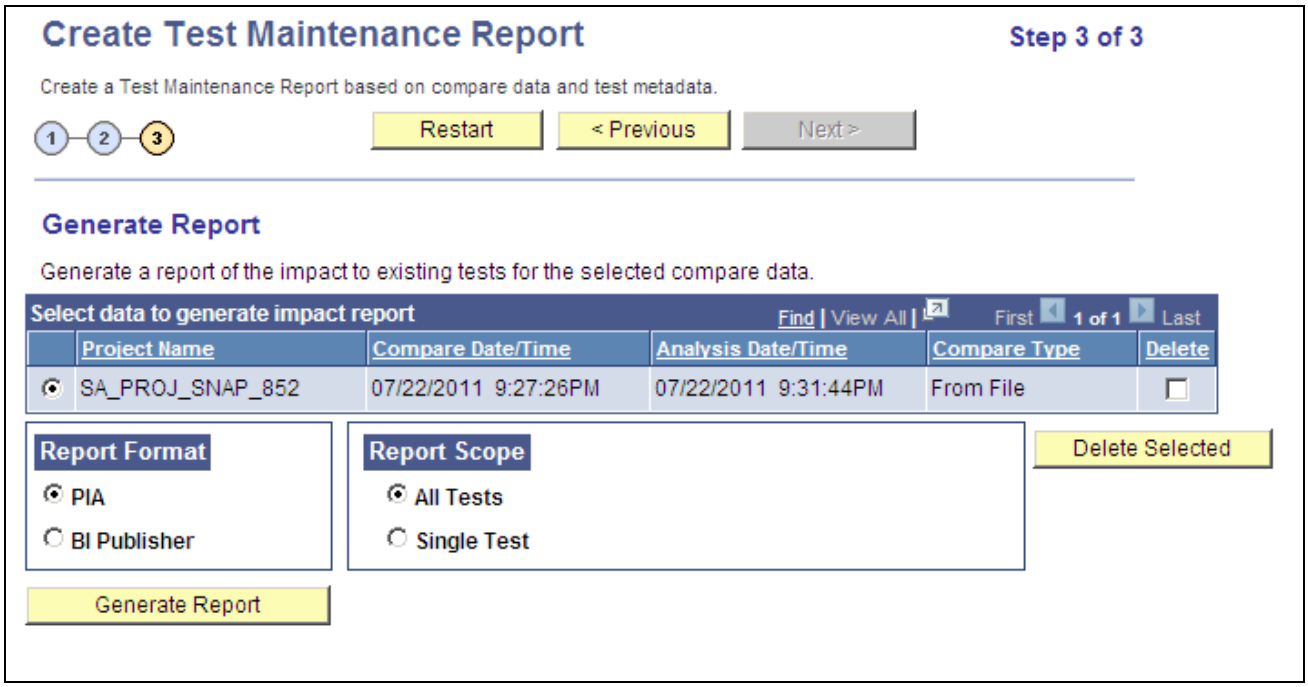
1. For the project compare report you created in Wizard Step 1, Task 5, specify whether the Compare Type is *From File* or *To Database.*
2. Click Analyze.

   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 1 and 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.

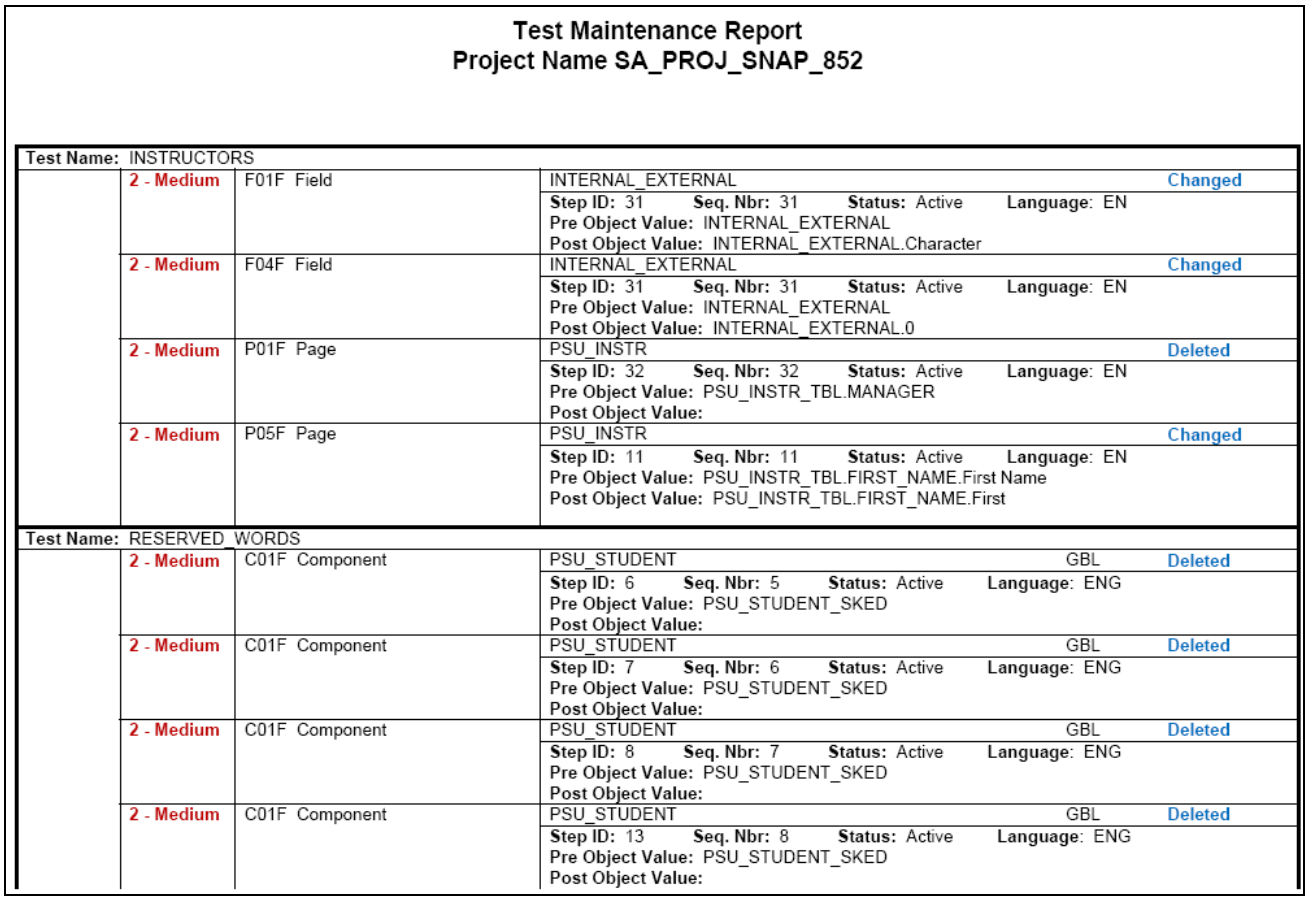
4. Select an analysis from the list and click Open.

   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 BI Publisher .

The following example shows a report in BI Publisher format:

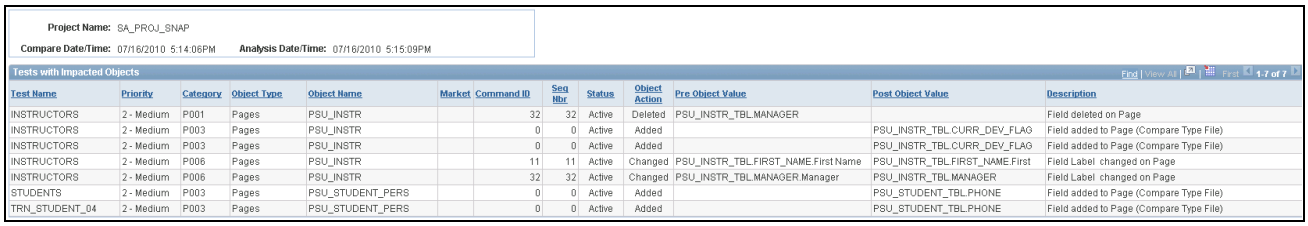


Test Maintenance Report
Project Name SA_PROJ_SNAP_852

Test Name: INSTRUCTORS				
	2 - Medium	F01F Field	INTERNAL_EXTERNAL	Changed
			Step ID: 31 **Seq. Nbr:** 31 **Status:** Active **Language:** EN **Pre Object Value:** INTERNAL_EXTERNAL **Post Object Value:** INTERNAL_EXTERNAL.Character	
	2 - Medium	F04F Field	INTERNAL_EXTERNAL	Changed
			Step ID: 31 **Seq. Nbr:** 31 **Status:** Active **Language:** EN **Pre Object Value:** INTERNAL_EXTERNAL **Post Object Value:** INTERNAL_EXTERNAL.0	
	2 - Medium	P01F Page	PSU_INSTR	Deleted
			Step ID: 32 **Seq. Nbr:** 32 **Status:** Active **Language:** EN **Pre Object Value:** PSU_INSTR_TBL.MANAGER **Post Object Value:**	
	2 - Medium	P05F Page	PSU_INSTR	Changed
			Step ID: 11 **Seq. Nbr:** 11 **Status:** Active **Language:** EN **Pre Object Value:** PSU_INSTR_TBL.FIRST_NAME.First Name **Post Object Value:** PSU_INSTR_TBL.FIRST_NAME.First	
Test Name: RESERVED_WORDS				
	2 - Medium	C01F Component	PSU_STUDENT GBL	Deleted
			Step ID: 6 **Seq. Nbr:** 5 **Status:** Active **Language:** ENG **Pre Object Value:** PSU_STUDENT_SKED **Post Object Value:**	
	2 - Medium	C01F Component	PSU_STUDENT GBL	Deleted
			Step ID: 7 **Seq. Nbr:** 6 **Status:** Active **Language:** ENG **Pre Object Value:** PSU_STUDENT_SKED **Post Object Value:**	
	2 - Medium	C01F Component	PSU_STUDENT GBL	Deleted
			Step ID: 8 **Seq. Nbr:** 7 **Status:** Active **Language:** ENG **Pre Object Value:** PSU_STUDENT_SKED **Post Object Value:**	
	2 - Medium	C01F Component	PSU_STUDENT GBL	Deleted
			Step ID: 13 **Seq. Nbr:** 8 **Status:** Active **Language:** ENG **Pre Object Value:** PSU_STUDENT_SKED **Post Object Value:**	

Example of a Test Maintenance report in BI Publisherformat

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

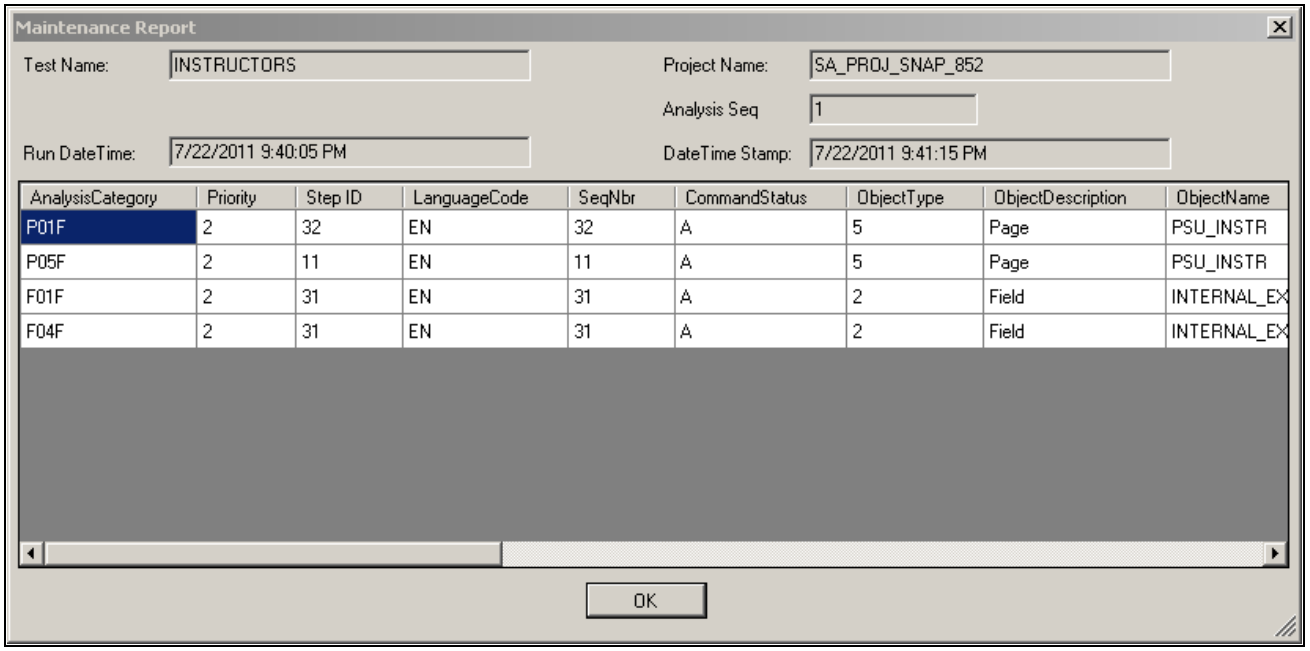
From a PIA report, you can click the Download icon to output the report to a spreadsheet. The following example shows a report in spreadsheet format:

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	Step ID	Seq Nbr	Status	Object Action	Pre Object Value	Post Object Value	Description
2	INSTRUCTORS	2 - Medium	F01F	Field	INTERNAL_EXTERNAL		31	31	Active	Changed	INTERNAL_EXTERNAL	haracter	Type File)
3	INSTRUCTORS	2 - Medium	F04F	Field	INTERNAL_EXTERNAL		31	31	Active	Changed	INTERNAL_EXTERNAL	INTERNAL_EXTERNAL.O	Field Decimal Positions changed (Compare Type File)
4	INSTRUCTORS	2 - Medium	P01F	Page	PSU_INSTR		32	32	Active	Deleted	PSU_INSTR_TBL.MANAGER		Type File)
5	INSTRUCTORS	2 - Medium	P05F	Page	PSU_INSTR		11	11	Active	Changed	PSU_INSTR_TBL.FIRST_NAME .First Name	PSU_INSTR_TBL.FIRST_ NAME.First	Field Label changed on Page (Compare Type File)
6	RESERVED_WORDS	2 - Medium	C01F	Component	PSU_STUDENT	Global	6	5	Active	Deleted	PSU_STUDENT_SKED		Page deleted from Component (Compare Type File)
7	RESERVED_WORDS	2 - Medium	C01F	Component	PSU_STUDENT	Global	7	6	Active	Deleted	PSU_STUDENT_SKED		Page deleted from Component (Compare Type File)
8	RESERVED_WORDS	2 - Medium	C01F	Component	PSU_STUDENT	Global	8	7	Active	Deleted	PSU_STUDENT_SKED		Page deleted from Component (Compare Type File)
9	RESERVED_WORDS	2 - Medium	C01F	Component	PSU_STUDENT	Global	13	8	Active	Deleted	PSU_STUDENT_SKED		Page deleted from Component (Compare Type File)
10	RESERVED_WORDS	2 - Medium	C01F	Component	PSU_STUDENT	Global	14	9	Active	Deleted	PSU_STUDENT_SKED		Page deleted from Component (Compare Type File)

Example of a Test Maintenance report in spreadsheet format

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



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 84.

Column Name	*Description*
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 84.
Step 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
Pre Object Value	The value before the object was changed.
Post Object Value	The value after the object was changed.

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 98.

Using Usage Monitor Data with PTF

PTF used by itself tracks the use of five step types – records, menus, fields, pages, and components. You can use PTF together with Usage Monitor to track 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 8.52: Performance Monitor, "Setting Up the Performance Monitor"

PeopleTools 8.52: Performance Monitor, "Working With Usage Monitor," Enabling Usage Monitor

Generating Usage Monitor Data

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

1. Create a new test in PTF.
2. Launch the Recorder.
3. Hook the browser.
4. Start recording.
5. In the PeopleSoft application browser, select 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.

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_SNAP_852

Impacted Objects and Objects with No Coverage — Find | View All | First 1-25 of 63 Last

Test Name	Object Type	Object Name	Object Detail	Test Metadata	Usage Monitor Data
** No Test Coverage **	Page	PSU_COURSE		No Data	No Data
** No Test Coverage **	Page	PSU_COURSE_MATL		No Data	No Data
** No Test Coverage **	Page	PSU_INSTR_PHOTO		No Data	No Data
** No Test Coverage **	Panel Group PeopleCode	PSU_CUST	GBL.SavePostChange	No Data	No Data
** No Test Coverage **	Panel Group Record Field PeopleCode	PSU_INSTR	GBL.DERIVED_ED_SVCS.DELETE_BTN FieldChange	No Data	No Data
** No Test Coverage **	Panel Group Record Field PeopleCode	PSU_INSTR	GBL.DERIVED_ED_SVCS.INSERT_BTN FieldChange	No Data	No Data
** No Test Coverage **	Translate	INTERNAL_EXTERNAL	0.2010-07-16	No Data	No Data
INSTRUCTORS	Component	PSU_INSTR		Active	No Data
INSTRUCTORS	Field	INTERNAL_EXTERNAL		Active	No Data
INSTRUCTORS	Page	PSU_INSTR		Active	No Data
INSTRUCTORS	Record	PSU_INSTR_TBL		Active	No Data
RESERVED_WORDS	Component	PSU_STUDENT	GBL	Active	No Data

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):

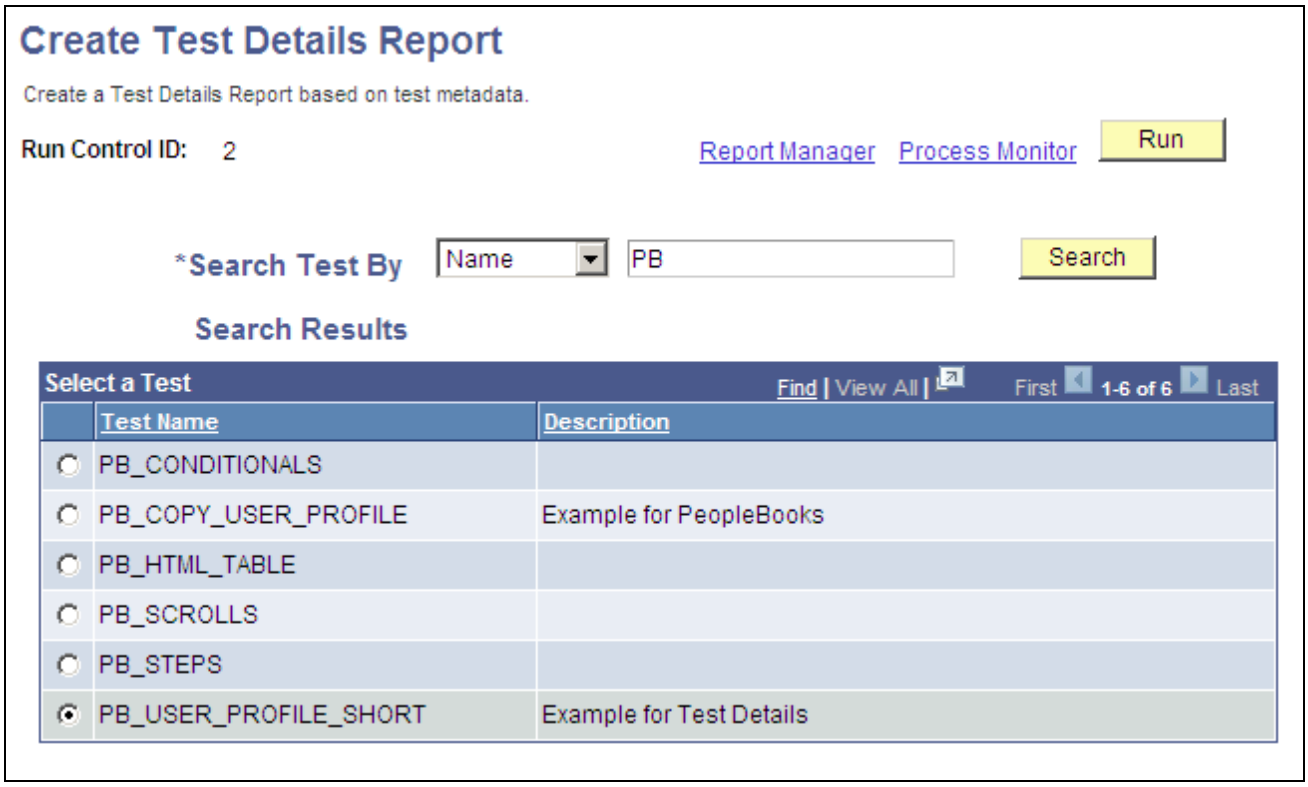
Column Name	***Description***
Test Name	The test that is impacted by a change.
Object	The type of definition that was in the change project.

Column Name	*Description*
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 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.
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.*

Running Test Details Reports

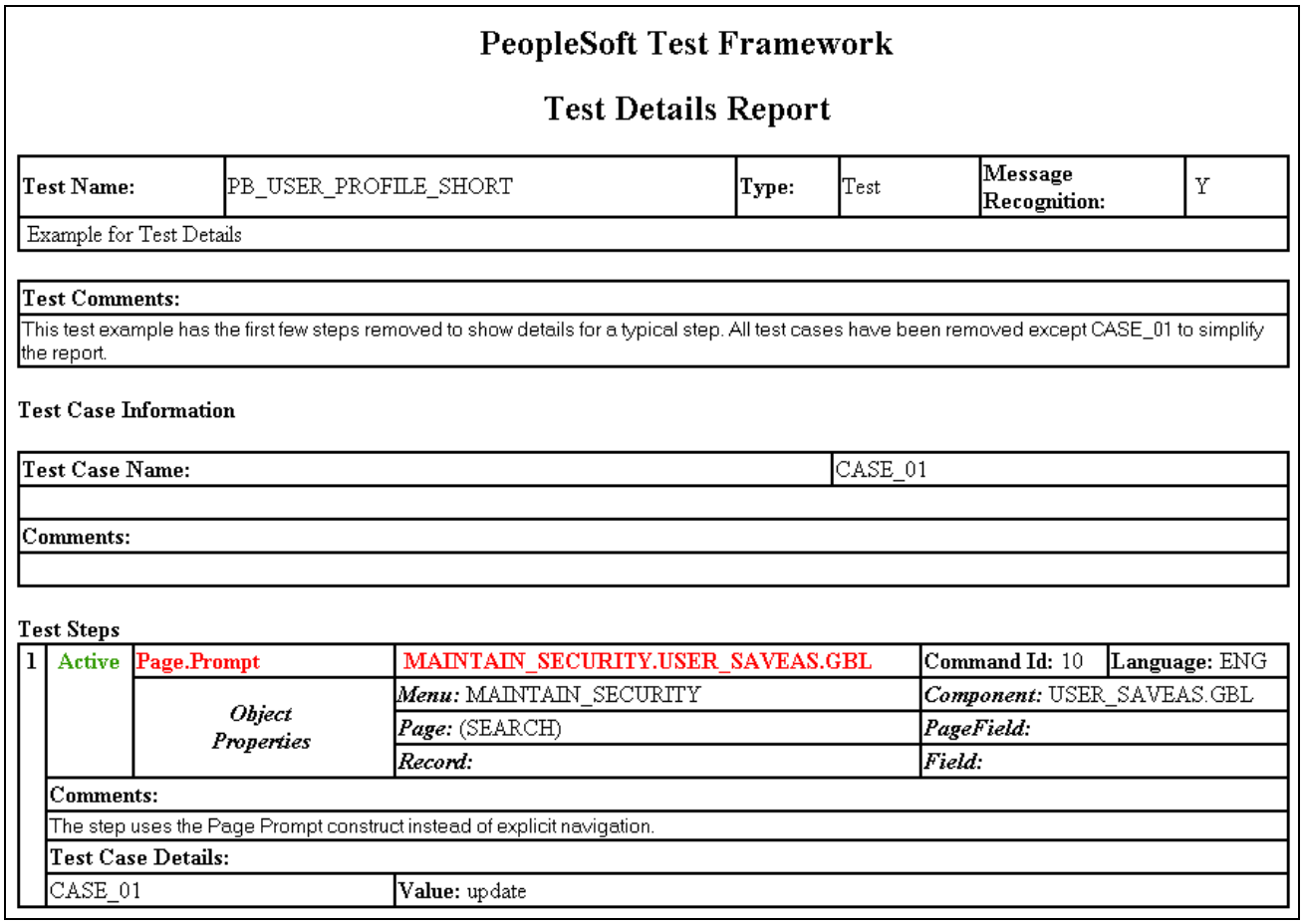
A test details report is an HTML file with details for a PTF test and its associated test cases, including comments in rich text format with images.

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



Create Test Details Report page

This example shows a test details report:



PeopleSoft Test Framework

Test Details Report

Test Name:	PB_USER_PROFILE_SHORT	**Type:**	Test	**Message Recognition:**	Y
Example for Test Details					

Test Comments:
This test example has the first few steps removed to show details for a typical step. All test cases have been removed except CASE_01 to simplify the report.

Test Case Information

Test Case Name:	CASE_01
Comments:	

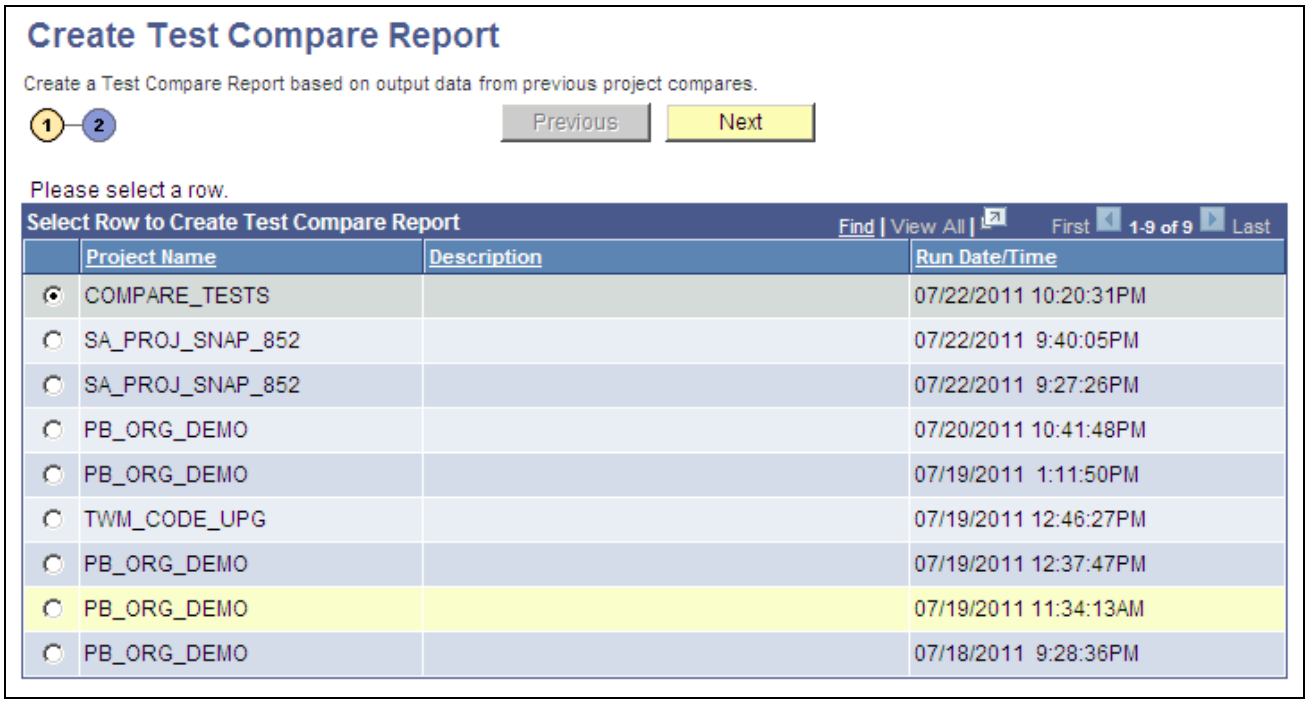
Test Steps

1	Active	Page.Prompt	MAINTAIN_SECURITY.USER_SAVEAS.GBL	**Command Id:** 10	**Language:** ENG
		Object Properties	***Menu:*** MAINTAIN_SECURITY	***Component:*** USER_SAVEAS.GBL	
			Page: (SEARCH)	***PageField:***	
			Record:	***Field:***	
	Comments:				
	The step uses the Page Prompt construct instead of explicit navigation.				
	Test Case Details:				
	CASE_01	**Value:** update			

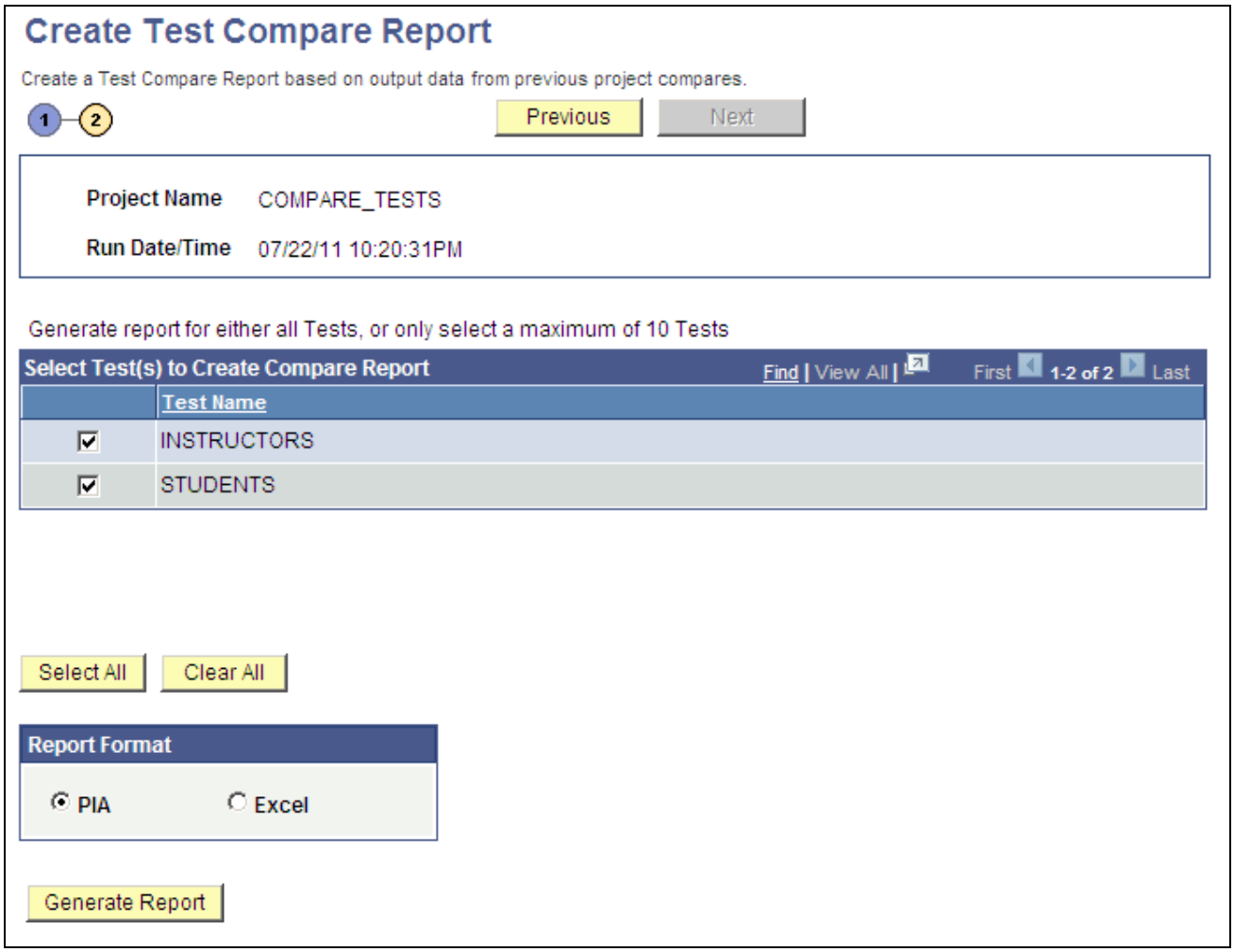
Example of a test details report

Creating a Test Compare Report

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

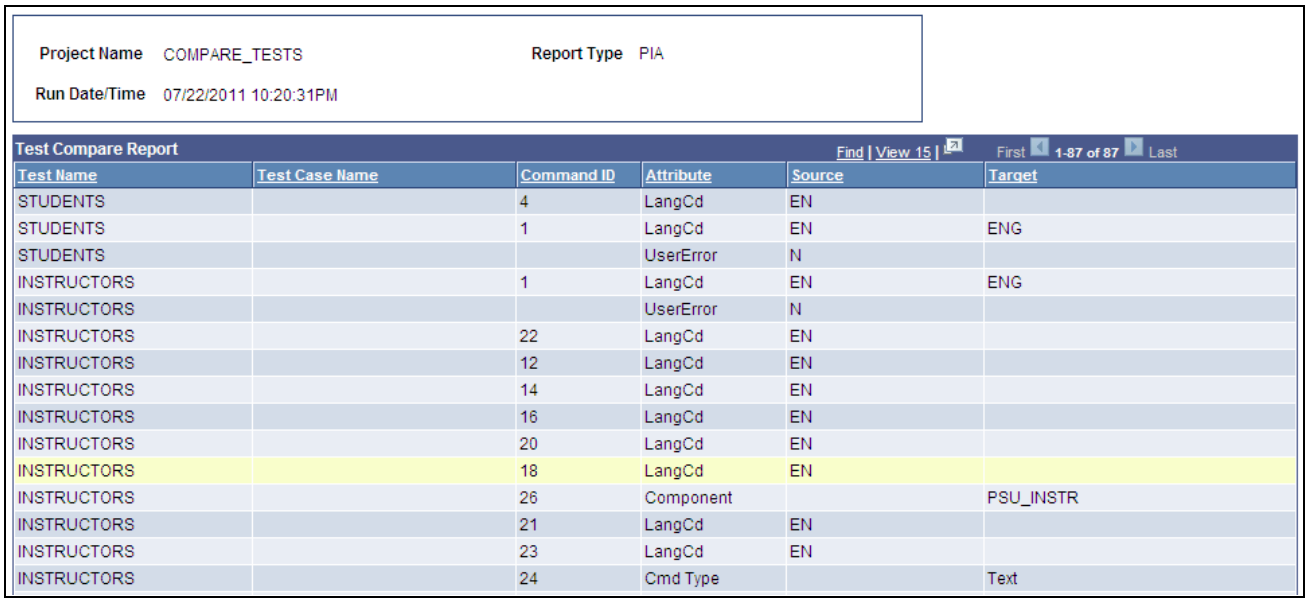


Create Test Compare Report Page 1 of 2



Create Test Compare Report Page 2 of 2

This is an example of a test compare report in PIA format:



Project Name COMPARE_TESTS Report Type PIA

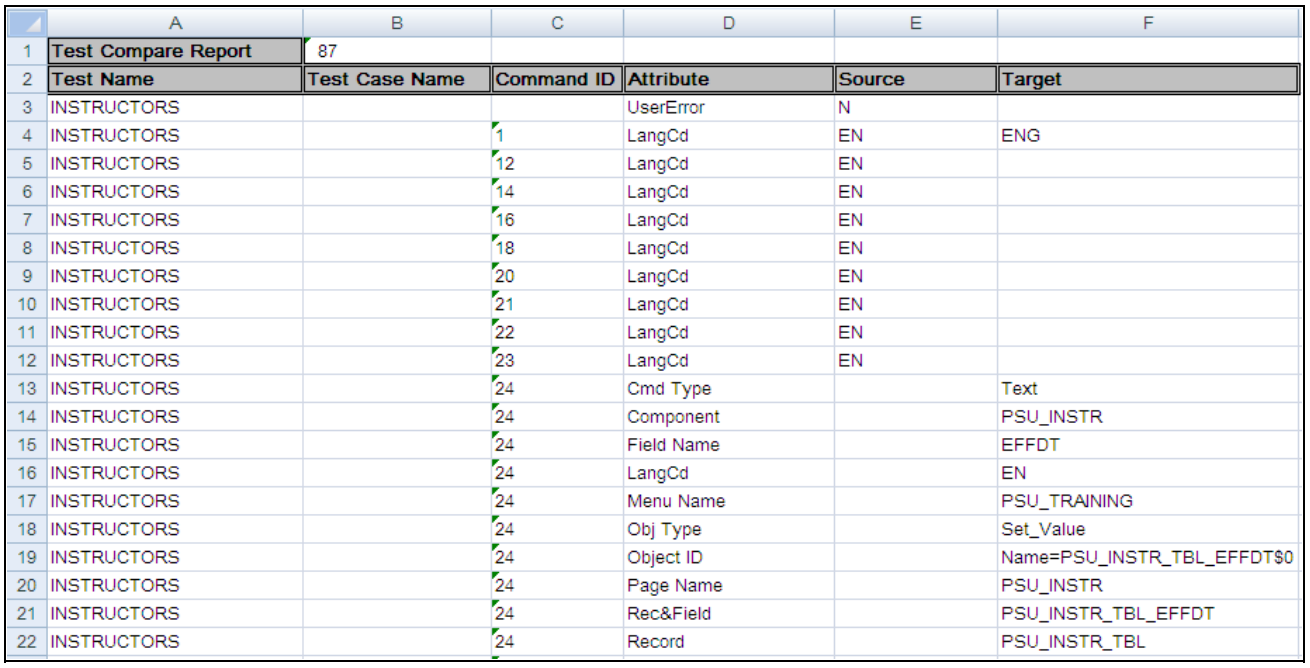
Run Date/Time 07/22/2011 10:20:31PM

Test Compare Report — Find | View 15 | First 1-87 of 87 Last

Test Name	Test Case Name	Command ID	Attribute	Source	Target
STUDENTS		4	LangCd	EN	
STUDENTS		1	LangCd	EN	ENG
STUDENTS			UserError	N	
INSTRUCTORS		1	LangCd	EN	ENG
INSTRUCTORS			UserError	N	
INSTRUCTORS		22	LangCd	EN	
INSTRUCTORS		12	LangCd	EN	
INSTRUCTORS		14	LangCd	EN	
INSTRUCTORS		16	LangCd	EN	
INSTRUCTORS		20	LangCd	EN	
INSTRUCTORS		18	LangCd	EN	
INSTRUCTORS		26	Component		PSU_INSTR
INSTRUCTORS		21	LangCd	EN	
INSTRUCTORS		23	LangCd	EN	
INSTRUCTORS		24	Cmd Type		Text

Example of a test compare report in PIA format

This is an example of a test compare report in Excel format:



	A	B	C	D	E	F
1	Test Compare Report	87				
2	Test Name	Test Case Name	Command ID	Attribute	Source	Target
3	INSTRUCTORS			UserError	N	
4	INSTRUCTORS		1	LangCd	EN	ENG
5	INSTRUCTORS		12	LangCd	EN	
6	INSTRUCTORS		14	LangCd	EN	
7	INSTRUCTORS		16	LangCd	EN	
8	INSTRUCTORS		18	LangCd	EN	
9	INSTRUCTORS		20	LangCd	EN	
10	INSTRUCTORS		21	LangCd	EN	
11	INSTRUCTORS		22	LangCd	EN	
12	INSTRUCTORS		23	LangCd	EN	
13	INSTRUCTORS		24	Cmd Type		Text
14	INSTRUCTORS		24	Component		PSU_INSTR
15	INSTRUCTORS		24	Field Name		EFFDT
16	INSTRUCTORS		24	LangCd		EN
17	INSTRUCTORS		24	Menu Name		PSU_TRAINING
18	INSTRUCTORS		24	Obj Type		Set_Value
19	INSTRUCTORS		24	Object ID		Name=PSU_INSTR_TBL_EFFDT$0
20	INSTRUCTORS		24	Page Name		PSU_INSTR
21	INSTRUCTORS		24	Rec&Field		PSU_INSTR_TBL_EFFDT
22	INSTRUCTORS		24	Record		PSU_INSTR_TBL

Example of a test compare report in Excel format

Querying PTF Report Tables

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

- PSPTTSTMAINTRPT

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

- PSPTTSTCOVRGRPT

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

Chapter 8

Incorporating Best Practices

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:

- Keep your desktop simple.
- Adopt naming conventions.
- Record first.
- Document tests.
- Clean up tests.
- Use execution options.
- Use page prompting.
- Use message recognition.
- Use the Process step type.
- Make tests dynamic.

Keep your Desktop Simple

Close all programs other than PTF and the PeopleSoft application browser during recording and execution of PTF tests. Other applications – especially those that alert the user with spontaneous notifications, such as email and instant messaging programs – can interfere with PTF and the PeopleSoft application browser and thus cause unexpected results during record and playback.

Adopt Naming Conventions

You should understand these PTF test asset naming limitations:

- Test names and test case names must be unique.

  Tests and test case are PeopleTools managed objects, a very important type of PeopleSoft metadata. As a result, test names must be unique to a database and test case names must be unique to a test.

- Test and test case names are limited to letters, numbers, and underscore characters.

  The first character of any test or test case name must be a letter.

- Test and test case names are limited to 30 characters.

You will find it easier to manage test assets if you adopt a systematic naming convention that reflects important characteristics of the tests and test cases you create. Here are a few suggested test characteristics that you can incorporate in your PTF test names:

- Functionality being tested.

  Include a brief indication of the functional area or business process validated by the test.

- Priority. A short code, such as "P1," to indicate priority can help testers more easily locate the tests that are critical or likely to be run most often.

- Execution order.

  PTF Explorer sorts tests alphabetically within each folder. It is sometimes useful to view tests in order of intended execution, especially if some tests depend on other tests having been run previously in the same database. Include a numeric component early in the name of the test to make alphabetic order equal to execution order.

It may seem simpler to use folders to categorize tests by the above characteristics. However, folder names are often not visible to the tester and do not appear on some PTF reports, such as the Test Maintenance Report. So, regardless of whether you categorize tests by folder, it is helpful to build test characteristics into the names of your tests.

The following three test names are examples of a naming convention that reflects functionality, priority, and intended execution order:

- WRKFRC_P1_01_HIRE_REQD_FLDS
- WRKFRC_P1_02_PERSON_CHNG 21
- WRKFRC_P1_03_JOB_DATA_UPDAT

Record First

A critical best practice when automating with PTF is to record first, and record whenever possible, to drive as much information, including PeopleTools metadata, into the recorded test as possible.

You must assertively record every test step from the test. That is, even if the input fields are already set to the expected value, you must explicitly set or verify that value when recording. The recorder only captures actions that you take against objects during record time. If you skip an object during recording because it is already set to the desired value, the test will ignore it and fail if the default value of the same object is different during execution.

When you record a date field that includes a calendar object that enables the user to select a date, it is best to explicitly enter the date value in the edit box.

You may need to perform two actions instead of one. For instance, if the test instructions say to select a check box but the check box is already selected, then you will need to pause the recording, deselect the checkbox, restart the recording, and then select the checkbox. Alternatively, you could click the checkbox twice and then delete the extra step from your test.

If you need to add a new step to a test, select the spot in the test where the new step will occur and record the step at the insertion point. Recording a step is more likely to drive accurate information into your test than trying to construct the entire step by editing the fields through the Test Editor.

See Also

Chapter 4, "Creating Tests and Test Cases," Recording Tests, page 51

Document Tests

Make it part of your automation routine to take advantage of the description fields in PTF tests and test cases as often as possible. PTF test and test case names tend to be short and abstract, so longer descriptions will help you and future testers understand the functional purpose of available PTF tests and test data.

You can find description fields in:

- Test description
- Test properties
- Test case description
- Test case properties
- Step information

Use the Log actions to make your tests self-documenting. For instance, you can add a Log.Message before a Test.Exec step to describe the test you are calling, and you can put a Log.Message in the called test to document what the test does.

You can add comments in RTF format to the test, test case, and steps to document your tests. Comments can include application snapshots.

Clean Up Tests

Immediately after recording a test, review the recorded test data in the Test Editor and correct actions taken inadvertently during recording, such as:

- Unnecessary or extra clicks on a clickable item, such as a check box.
- Clicks on the wrong objects, such as links and images.
- Incorrect text entered into text boxes.

Revise the recorded values in the Value field (for editable fields) and delete unneeded steps.

This might be a good time to evaluate whether you should incorporate reserved words and variables to replace static values that may be different when the test is executed.

See Also

Chapter 8, "Incorporating Best Practices," Make Tests Dynamic, page 113

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	Comment	Active	Scroll ID	Type	Action	Recognition	Value
1		1		☑		Browser	Start_Login		
2		2		☐		Link	Click	innerText=Sign in to PeopleSoft	
3		3		☐		Text	Set_Value	Name=userid	QEDMO
4		4		☐		Pwd	Set_Value	Name=pwd	1ENC41B9FB38D060CB90E0108BF840409B50B0309...
5		5		☐		Button	Click	Name=Submit	

Example of using Browser.Start_Login

See Also

Chapter 2, "Installing and Configuring PTF," Configuring Execution Options, page 23

Chapter 10, "Test Language Reference," Start_Login, page 124

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.

PTF can be configured to insert page prompt constructs automatically during recording. The default behavior is set by the PTF adminstrator on the Define Configuration Options page and can be overriden during recording using the recorder utility tools.

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

See Also

Chapter 10, "Test Language Reference," Prompt, page 148

Chapter 10, "Test Language Reference," PromptOK, page 148

Chapter 2, "Installing and Configuring PTF," Defining Configuration Options, page 11

Use the Process Step Type

The Process step 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 149

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 take advantage of 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 66

Chapter 5, "Developing and Debugging Tests," Using Conditional Logic, page 67

Chapter 5, "Developing and Debugging Tests," Incorporating Scroll Handling, page 71

Chapter 5, "Developing and Debugging Tests," Using Reserved Words, page 64

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 54

Chapter 5, "Developing and Debugging Tests," Calling Tests, page 75

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	Comment	Active	Scroll ID	Type	Action	Recognition	Field Label	Value
1		1		☑		Browser	Start_Login			
2		8		☑		Browser	FrameSet	TargetContent		
3		10		☑		Page	Prompt	MAINTAIN_SECURITY.USER_SAVEAS.GBL		update
4		11		☑		Text	Set_Value	Name=PSOPRDEFN_SRCH_OPRID		QEDMO
5		18		☐		Log	Message	This step will not execute		
6		12		☑		Page	PromptOk			
7		13		☑		Text	Set_Value	Name=DERIVED_CLONE_OPRID	New User ID	JGDMO
8		14		☑		Text	Set_Value	Name=DERIVED_CLONE_OPRDEFNDESC	Description	Jim's ID
9		15		☑		Pwd	Set_Value	Name=DERIVED_CLONE_OPERPSWD	New Password	1ENCF39A6DEC0870C5BB7257A...
10		16		☑		Pwd	Set_Value	Name=DERIVED_CLONE_OPERPSWDCONF	Confirm Password	1ENCF39A6DEC0870C5BB7257A...
11		17		☑		Page	Save			
▶*				☑						

Example of a PTF test

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

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

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

Test maintenance reports use the ID value.

Comment — Click in this field to add a comment for the step.

Active — Deselect this field to inactivate a step. PTF l skips 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 step types are Text, Checkbox, Browser, and so on.

Action The action the test is to take on the object. The two most common actions used on a Text object, for example, would be Set_Value and Verify.

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

Field Label Displays the label for the field referenced in the step.

The Field Label column appears in a test when Column is selected in the Show Field Label field in Local Options. If Tooltip is selected in the Show Field Label field, the field label appears in a tooltip.

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

In a step recorded in verify mode, this would be the value that was present in the object when it was verified.

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	Comment	Active	Scroll ID	Type	Action	Recognition	Value
1		1		☑		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 step types and actions.

PTF Test Language

This section discusses:

- Validation
- Parameters
- Variables
- Reserved words

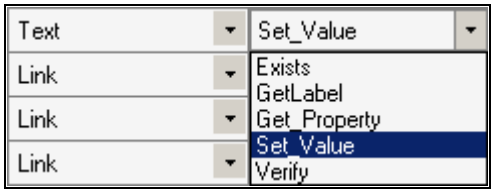
Validation

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

The PTF Test Editor automatically limits your choice of actions based on the step type selected.

For example, if a step has the Type field set to *Text* and the Action field set to *Set_Value*, you can change the Action field to *Verify* since it is included in the list of available values for a text object.

This example shows a drop-down list with *Verify* as one of the values for the Action field when the Type field is set to *Text*:



Action drop-down list for Text

Parameters

Typically, you place parameters in the Recognition 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 153.

Steps that return a value require the parameter `ret=&variable;`

For example:

```
ret=&chart_val;
```

The system ignores unneeded parameters. For example, Browser.Start and Browser.Start_Login do not take any parameters, so the system ignores any values in the Recognition field for Browser.Start or Browser.Start_Login.

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

System variables are predefined variables that PTF populates at runtime. For instance, %test% is populated with the test name at runtime.

See Chapter 9, "Using the PTF Test Language," System Variables, page 120.

See Also

Chapter 10, "Test Language Reference," Variable, page 164

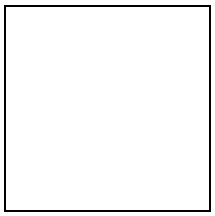
Chapter 5, "Developing and Debugging Tests," page 63

Chapter 9, "Using the PTF Test Language," System Variables, page 120

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.



Example of using the #TODAY reserved word

See Also

Chapter 5, "Developing and Debugging Tests," Using Reserved Words, page 64

Chapter 10, "Test Language Reference," Reserved Words, page 173

System Variables

System variables are predefined variables that PTF populates at runtime. System variables provide data about the current environment, execution options, test, and application.

Because system variables are replaced with a text string at runtime, you can place a system variable wherever you would place a text string. Commonly, test developers assign a system value to a user-defined variable.

The following example shows two methods of assigning system variables to user-defined variables:

Variable	Set_Value	&User	%eo.dbuser%
Variable	Set_Value	&Test=%test%	

Example of using system variables

See Also

Chapter 10, "Test Language Reference," System Variables, page 181

Chapter 10

Test Language Reference

This chapter discusses:

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

Step Types

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

Browser

These are the actions associated with the Browser step type.

Close

Description

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

FrameSet

Description

Sets the focus in a browser frame.

Embedded frames include a number, such as ptModFrame_1. You can substitue ## for the number, for example ptModFrame_## and PTF will search through all frames starting with ptModFrame_ and use the one with the highest number.

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 step type.

Click

Description

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

See Chapter 10, "Test Language Reference," Click, page 166.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

Chart

These are the actions associated with the Chart step 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

Returns the text value for the specified 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 stept 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 index 2 is &chart_val;	

Example of GetText action for a Chart step type

CheckBox

These are the actions associated with the CheckBox step type.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

Set_Value

Description

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

See Chapter 10, "Test Language Reference," Set_Value, page 171.

Verify

Description

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

See Chapter 10, "Test Language Reference," Verify, page 172.

ComboBox

These are the actions associated with the ComboBox step type.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

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.

When used with #LIST#, Set_Value returns an error if the expected value (the bracketed value) is not the same as the actual value.

Example

This example sets the field value to French and verifies that the values English, French, and Finnish exist in the list.

ComboBox	Set_Value	Name=PSOPRDEFN_LANGUAGE_CD	#LIST#English|French|Finnish

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

See Also

Chapter 10, "Test Language Reference," #LIST#, page 178

Verify

Description

Compares the value in the browser to the expected value and adds a Pass or Fail log entry for the validation. Use a vertical pipe (|) to separate values to be verified. Use square brackets ([]) to specify which value is expected to be selected.

Example

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

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

Example of Verify action

Conditional

You can control the flow of execution of tests using conditional constructs. A conditional construct begins with an If_Then action and ends with an End_If action. You can optionally include an Else action.

Conditional constructs can be nested.

These are the actions associated with the Conditional step type.

If_Then

Description

The first step in a conditional contstruct. The system evaluates the logical expression in the Recognition field of the If_Then step. If the expression evaluates to True, the system executes the steps between the If_Then step and the End_If step or the Else step, if it exists. If the expression evaluates to False, the system jumps to the Else step, if it exists, or to the End_If step if there is no Else, and continues execution.

If_Then supports these logical operators:

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

You can use the AND and OR logical operators to specify multiple conditions.

Else

Description

(Optional) If the logical expression evaluates to False, the system executes the steps between the Else step and the End_If step.

End_If

Description

The close statement of the If_Then construct.

Example

This example shows the use of multiple conditions and nested conditionals:

Variable	Set_Value	&Integer	3
Conditional	If_Then	&Integer>=1 AND &Integer<=10 OR &Integer=0	
Log	Message	Determine whether Integer is odd or even	
Log	Message	Enter first nested conditional	
Conditional	If_Then	&Integer = 0	
Log	Message	Do not divide by 0.	
Conditional	Else		
Variable	Set_Value	&Half=Divide(&Integer|2|dec=1)	
Log	Message	Enter second nested conditional	
Variable	Set_Value	&Odd=Substr(&Half|3)	
Conditional	If_Then	&Odd = 5	
Log	Message	The number is odd	
Conditional	Else		
Log	Message	The number is even	
Conditional	End_If		
Conditional	End_If		
Conditional	End_If		

Example of the If_Then conditional construct

DataMover

This is the action associated with the DataMover step 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 path is not set in the execution options, it uses the system temp folder.

Example

This example shows the use of the Exec action:

Example of the DataMover Exec action

Div

This is the action associated with the Div step type.

Click

Description

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

See Chapter 10, "Test Language Reference," Click, page 166.

Execution

Use the Execution actions in shell tests to modify the behavior of tests during execution. You typically set these options as you are developing tests to facilitate the development process.

The Execution step type is available only in shell tests.

Set_Options

Description

Override the default execution option. Specify the name of a valid execution option in the Recognition column.

Skip_PageSave

Description

Specify *True* or *False* in the Recognition column. Specify True to skip Page.Save steps. You would, for instance, select this option to avoid creating duplicate values if you plan to run a test repeatedly.

This action overrides the Skip PageSave setting in Execution Options.

Skip_RunRequest

Description

Specify *True* or *False* in the Recognition column. Specify True to skip Process.Run steps.

This action overrides the Skip RunRequest setting in Execution Options.

Parameter	***Description***
True	Skip Process.Run steps.
False	Execute Process.Run steps.

Skip_Login

Description

Specify*True* or *False* in the Recognition column. Specify True to skip Browser.Start_Login steps.

Parameter	***Description***
True	Skip Browser.Start_Login steps.
False	Execute Browser.Start_Login steps.

StopOnError

Description

Specify *True* or *False* in the Recognition column. Specify True to stop execution when a Fail is logged.

This action overrides the Stop on Error setting in Test Editor.

Parameter	***Description***
True	Stop execution when a test encounters a Fail.
False	Continue execution when a test encounters a Fail.

See Also

Chapter 2, "Installing and Configuring PTF," Configuring Execution Options, page 23

File

This is the action associated with the File step type. The File step type corresponds to the object in PIA that allows users to upload files to the PeopleSoft application.

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:

Example of the File Upload action

HTMLTable

These are the actions associated with the HTMLTable step type.

CellClick

Description

Clicks on a specific HTMLTable cell based on the index parameter.

Parameter	*Description*

index=*I*/*R*/*C*:	The table, row, column index string.
	For example:
	`index=&CellIndex`
	`index=1/2/3`
	`;`
	In the second example, CellClick clicks on the third column of the second row of the first table.

CellClickOnChkB

Description

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

Parameter	*Description*

index=*I*/*R*/*C*:	The table, row, column index string.
	For example:
	`index=&CellIndex`
	index=1/2/3;
	In the second example, this function clicks on a checkbox within the third column of the second row of the first table.

chkidx=*value*;	The CheckBox object index inside the cell.
check=*value*;	`check=Y` – Select the checkbox. `check=N` – Clear the checkbox. This parameter is optional. The default value is Y.

CellClickOnImage

Description

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

Parameters

Parameter	*Description*

index=*I*/*R*/*C*:	The table, row, column index string.
	For example:
	`index=&CellIndex`
	`index=1/2/3;`
	In the second example, this function clicks on an image within 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=*I*/*R*/*C*:	The table, row, column index string.
	For example:
	`index=&CellIndex`
	`index=1/2/3;`
	In the second example, this function clicks on a link within 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=*I*/*R*/*C*:	The table, row, column index string.
	For example:
	`index=&CellIndex`
	`index=1/2/3;`
	In the second example, this function verifies whether a cell exists within the third column of the second row of the first table.

ret=*&variable*	The return value. True – the cell exists. False – the cell does not exist.

CellGetIndex

Description

Searches the page for the text value specified in the text parameter and returns the index string for the first cell that contains the text. The index string is in the form of *I/R/C*, where *I* is the table index, *R* is the row number, and *C* is the column number. Other actions, such as CellClick, CellGetValue, and so on, use an index string to reference a specific cell.

Parameters

Parameter	*Description*
text=*value*;	The text to look for on the page.
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.

Parameter	Description
ret=*&variable*	The return value is an index string in the form of I/R/C, where I is the table index, R is the row number, and C is the column number. For example: `ret=&CellIndex`

CellGetValue

Description

Returns the contents of an HTMLTableCell.

Parameters

Parameter	Description

index=*I*/*R*/*C*:	The table, row, column index string.
	For example:
	`index=&CellIndex`
	`index=1/2/3;`
	In the second example, this function returns the contents of the third column of the second row of the first table.

ret=*&variable*	The return value.

ColCount

Description

Returns the number of columns for the HTMLTable row.

Parameters

Parameter	Description
table=*value*;	The table index.
row=*value*;	The row index.

Parameter	*Description*
index=*I*/*R*/*C*:	An index string in the form of I/R/C, where I is the table index, R is the row number, and C is the column number. As an alternative to specifying the table and row parameters, you can specify an index string, such as the return value from a CellGetIndex action. For example: `index=&CellIndex;`
ret=*&variable*	The return value.

RowCount

Description

Returns the number of rows for the HTMLTable.

Parameters

Parameter	*Description*
table=*value*;	The table index.
index=*I*/*R*/*C*:	An index string in the form of I/R/C, where I is the table index, R is the row number, and C is the column number. As an alternative to specifying the table parameter, you can specify an index string, such as the return value from a CellGetIndex action. For example: `index=&CellIndex;`
ret=*&variable*	The return value.

Example

This is an example of several of the HTMLTable actions:

Browser	Start_Login		
HTMLTable	CellGetIndex	text=My Reports; ret=&HTMLIndex; equal=true	
Conditional	If_Then	&HTMLIndex=0/0/0	
Link	Click	id=pttabpercontent	
HTMLTable	CellGetIndex	text=My Reports; ret=&HTMLIndex; equal=true	
Variable	Set_Value	&CheckIndex	sum(&HTMLIndex, -1, 3, /)
HTMLTable	CellClick	index=&CheckIndex	
Page	Save		
Conditional	End_If		

Example of actions for the HTMLTable step type

Image

These are the actions associated with the Image step type.

Click

Description

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

See Chapter 10, "Test Language Reference," Click, page 166.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

RightClick

Description

Performs a right-click on the image. This action is supported only for a related-content image glyph.

Link

These are the actions associated with the Link step type.

Click

Description

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

See Chapter 10, "Test Language Reference," Click, page 166.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

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.

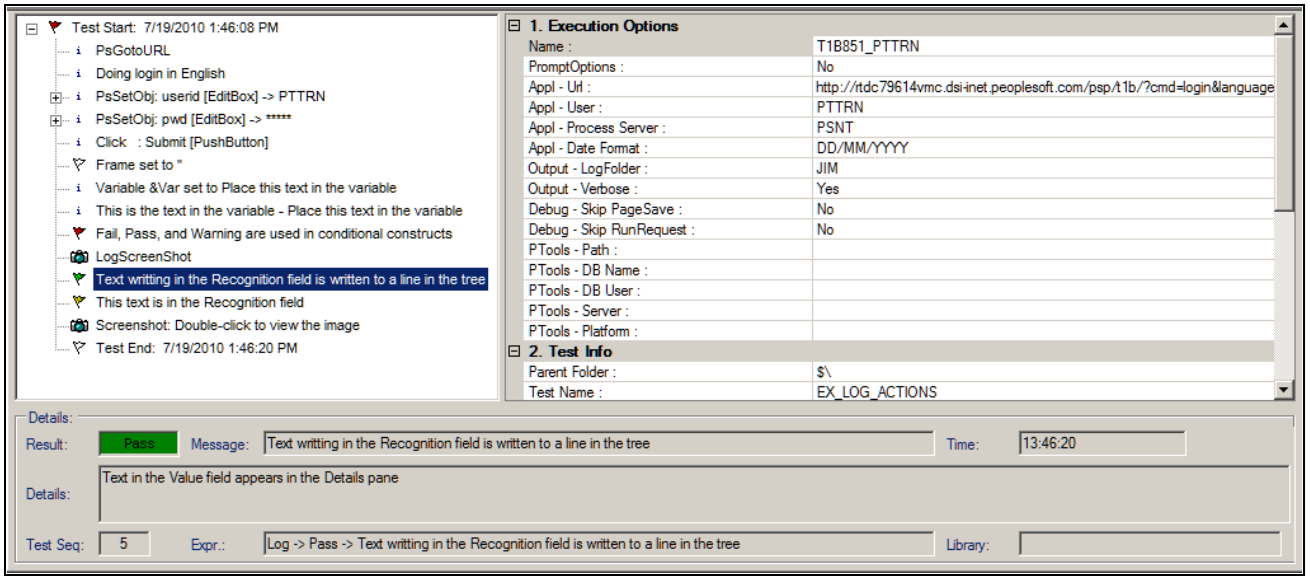
Example

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 step type.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

Set_Value

Description

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

See Chapter 10, "Test Language Reference," Set_Value, page 171.

SetValue_InModal

Description

Use the SetValue_InModal action to set the value of a long text field on a modal page.

Verify

Description

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

See Chapter 10, "Test Language Reference," Verify, page 172.

Loop

These are the actions associated with the Loop step type.

Do

Description

Executes the steps between the Loop.Do step and the Loop.End_Loop until a Loop.Exit step is encountered.

Example

This example shows the Loop.Do construct:

Variable	Set_Value	&Var = 0	
Loop	Do		
Log	Message	The variable is &Var	
Conditional	If_Then	&Var > 3	
Loop	Exit_Loop		
Conditional	End_If		
Loop	End_Loop		

Example of a Loop.Do construct

End_Loop

Description

Terminates a Loop.Do or Loop.While construct.

Exit

Description

Exits a Loop.For or Loop.While construct. Execution continues with the step following the End_Loop step. Typically, a Loop.Exit step is placed within a conditional construct.

While

Description

Executes the steps between the Loop.While step and the Loop.End_Loop while the expression in the Recognition field evaluates to True.

When the expression evaluates to false, flow skips to the step following the End_Loop step.

Example

The following example shows the Loop.While construct:

Variable	Set_Value	&Var = 0	
Loop	While	&Var <=3	
Log	Message	The variable is &Var	
Variable	Set_Value	&Var = add(&Var;|1)	
Loop	End_Loop		

Example of Loop.While

For

Syntax

&variable=*begin_value* to *end_value*;

Description

Executes the steps between the Loop.For step and the Loop.Next step until the expression in the Recognition field evaluates to False, then flow skips to the step following the Loop.Next step.

Parameters

Parameter	***Description***
&variable	The variable to be used in the comparison. This variable is incremented in the Loop.Next step.
begin_value	The starting value.
end_value	The ending value.

Next

Description

Terminates a Loop.For construct. Loop.Next increments the variable in the Loop.For step.

See Chapter 10, "Test Language Reference," Add, page 182.

Example

The following example shows the Loop.For construct:

Variable	Set_Value	&Var = 0	
Loop	For	&Var; 1 to 5	
Log	Message	The variable is &Var	
Loop	Next		

Example of a Loop.For construct

MultiSelect

These are the actions associated with the MultiSelect step type.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

Set_Value

Description

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

See Chapter 10, "Test Language Reference," Set_Value, page 171.

Verify

Description

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

See Chapter 10, "Test Language Reference," Verify, page 172.

Page

These are the actions associated with the Page step type.

Expand

Description

Attempts to expand all the collapsed sections on the current page.

Go_To

Description

Accesses a page by selecting a page tab. Enter the 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:

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.

SecPage_Close

Description

Closes the secondary page.

SecPage_Open

Description

Opens a secondary page.

Parameters

Parameter	***Description***
page= *value*	Specify the name of the secondary page.

Example

The following example shows the SecPage_Open action:

Example SecPage_Open

Process

The Process actions run processes through Process Scheduler.

Run

Description

Runs a Process Scheduler process.

Parameters

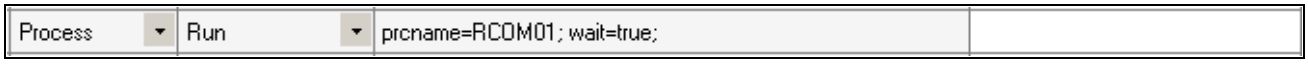
Parameter	*Description*
prcname=*value*;	The process name.
prctype=*value*:	The process type.
wait=*value*;	True - the test waits for the process to finish. False - the test does not wait for the process to finish. The default is False.
outtype=*value*;	The process output type.
outformat=*value*;	The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

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:



Example of the Run action

Run_Def

Description

Changes the default behavior of the run process. Insert these steps prior to the Process Run step that they modify. 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.
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 22

Pwd

These are the actions associated with the Pwd step type.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

Set_Value

Description

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

See Chapter 10, "Test Language Reference," Set_Value, page 171.

Query

This is the action associated with the Query step type.

Exec

Description

Runs a public query in PeopleSoft Query and downloads the results. To run a private query, use the Exec_Private action.

Exec_Private

Description

Runs a private query in PeopleSoft Query and downloads the results.

Parameter	*Description*
outFile=*value*;	The query output file.
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.
0rows=*value*;	If the query returns zero rows, add a log entry of type value. Valid values are Pass, Fail, or Warning.
Nrows=*value*;	If the query returns one or more rows, add a log entry of type value. Valid values are Pass, Fail, or Warning.

Use the following formats to specify query parameters:

Page Control	*Format*
Text	param=*value*
Radio button	param={RB}*value*
Combo box	param={CB}*value*
Check box	param={CH*}value*
Text box	param={EB}*value*

Radio

These are the actions associated with the Radiostep type.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

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 Recognition field using the `Value=` parameter.

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 step type

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

Set_Value

Description

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

See Chapter 10, "Test Language Reference," Set_Value, page 171.

Verify

Description

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

See Chapter 10, "Test Language Reference," Verify, page 172.

RichText

These are the actions associated with the RichText step type.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

Set_Value

Description

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

See Chapter 10, "Test Language Reference," Set_Value, page 171.

Scroll

These are the actions associated with the Scroll step type.

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

See Also

Chapter 5, "Developing and Debugging Tests," Incorporating Scroll Handling, page 71

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.
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 override the default name object of scroll buttons and the scroll parent. This is necessary, for example, when a page uses custom objects (such as links or pushbuttons) to handle conventional scroll actions such as adding or deleting rows from the scroll.

Parameters

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

type=*value*;	Thestep 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	Override the Add new row button.
del	Override the Delete row button.
next	Override the Next button.
prev	Override the Previous button.
first	Override the First button.
last	Override the Last button.
parent	Reassign the parent for a specific scroll area.

Value	*Description*
parent	Reassign the parent for a specific scroll area.

ModalGrid_Close

Description

Closes a modal grid.

ModalGrid_Open

Description

Opens a modal grid.

Key_Set

Description

Defines each key field of a scroll. Use multiple Key_Set steps for scrolls with multiple key fields.

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

Parameters

Parameter	*Description*
type=value;	The step 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 example uses the Definition action with the `def=Add` parameter to assign the Add action to the image $ICField3$newm$0$$mg$0:

1	Scroll	Definition	def=Add; Type=Image; Value=Name=$ICField3$newm$0$$mg$0	

Example of a Definition action with def=Add parameter

In this example the Definition action with a `def=parent` parameter reassigns the parent:

1	Scroll	Key_Set	Type=Text; Name=PSU_EMP_REVIEW_REVIEW_YEAR	2009
1	Scroll	Action	Ret=&Scr1	upd
	Text	Set_Value	Name=PSU_EMP_REVIEW_REVIEW_DAYS&Scr1	365
2	Scroll	Definition	def=parent; value=&Scr1	
2	Scroll	Key_Set	Type=ComboBox; Name=PSU_EMP_RVW_RVR_REVIEWER_ID	00013
2	Scroll	Action	ret=&Scr2	upd
	ComboBox	Set_Value	Name=PSU_EMP_RVW_RVR_REVIEW_TYPE&Scr2	S

Example of the Definition action with the def=parent paramenter

Span

These are the actions associated with the Span step type.

Click

Description

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

See Chapter 10, "Test Language Reference," Click, page 166.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

MouseOver

Description

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

See Also

PeopleTools 8.52: 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 8.52: 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 record a verification of a field value in a mouseover popup window:

1. From the recorder tool bar, click and drag the Verify icon 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 Verify icon 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 172.

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 step type.

Exec

Description

Calls another test or library test.

Specify the child test name in the Recognition field and one or more test case names in the Value field, separated by commas.

You can also click in the Recognition field then click the elipsis icon and select a test case to automically populate the Recognition and Value fields with the test name and the test case name.

Test.Exec supports the use of parameters that enable you to pass dynamic values from a calling test to a library test.

Right-click the test name in the Recognition field and select Open Test to open the child test.

Use the #IGNORE reserved word in the Value field to skip the call to the child test for a given test case.

See Chapter 5, "Developing and Debugging Tests," Using Parameters with Library Tests, page 76.

Example

This test calls the test COPY_USER_PROFILE with the DEFAULT, CASE_01, and CASE_02 test caseS, then skips the call to PROCESS:

Test	Exec	**COPY_USER_PROFILE**	DEFAULT, CASE_01, CASE_02
Test	Exec	**PROCESS**	#IGNORE

Example of the Test Exec action

Text

These are the actions associated with the Text step type.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

Set_Value

Description

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

See Chapter 10, "Test Language Reference," Set_Value, page 171.

Verify

Description

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

See Chapter 10, "Test Language Reference," Verify, page 172.

Variable

This is the action associated with the Variable step 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 119

Wait

These are the actions associated with the Wait step type.

For_Seconds

Description

Specify in the Recognition field the number of seconds PTF will wait before proceeding to the next step.

For_Value

Description

Wait until the field contains the value specified in the Value field.

Parameters

Parameter	*Description*
type= *value*	Specify the step type, such as Text, Combobox, Link, and so on.
timeout=*value*	[Optional] Specify the time out value, in seconds. The default is 300 seconds.
refresh=*value*	[Optional] Specify the refresh button name. If the refresh= parameter is specified, PTF waits five seconds between each validation process and the refresh click.

Example

This example shows the use of Wait:

Wait	For_Seconds	60	
Wait	For_Value	Type=Link;Name=STATUS;	
Wait	For_Value	Type=Link;Name=STATUS; Refresh=Name=RA_REFRESH_BTN; timeout=90;	

Example of the Wait actions

Common Actions

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

Click

Description

Performs a mouse click on the specified object.

This is the list of objects that support this action:

- Button

  See Chapter 10, "Test Language Reference," Button, page 124.

- Image

  See Chapter 10, "Test Language Reference," Image, page 139.

- Link

  See Chapter 10, "Test Language Reference," Link, page 140.

- Span

  See Chapter 10, "Test Language Reference," Span, page 160.

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 124.

- CheckBox

  See Chapter 10, "Test Language Reference," CheckBox, page 126.

- ComboBox

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

- Image

  See Chapter 10, "Test Language Reference," Image, page 139.

- Link

  See Chapter 10, "Test Language Reference," Link, page 140.

- LongText

  See Chapter 10, "Test Language Reference," LongText, page 142.

- MultiSelect

  See Chapter 10, "Test Language Reference," MultiSelect, page 146.

- Pwd

  See Chapter 10, "Test Language Reference," Pwd, page 151.

- Radio

  Refer to the Radio step type entry for details.

  See Chapter 10, "Test Language Reference," Radio, page 153.

- Span

  See Chapter 10, "Test Language Reference," Span, page 160.

- Text

  See Chapter 10, "Test Language Reference," Text, page 163.

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:

Example of the Exists action

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

Use the HTML Browser feature of the Message tool to identify the properties and values of an object.

See Chapter 5, "Developing and Debugging Tests," Using the Message Tool, page 63.

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.

This is the list of objects that support this action:

- Button

  See Chapter 10, "Test Language Reference," Button, page 124.
- CheckBox

  See Chapter 10, "Test Language Reference," CheckBox, page 126.
- ComboBox

  See Chapter 10, "Test Language Reference," ComboBox, page 127.
- Image

  See Chapter 10, "Test Language Reference," Image, page 139.
- Link

  See Chapter 10, "Test Language Reference," Link, page 140.
- LongText

  See Chapter 10, "Test Language Reference," LongText, page 142.
- MultiSelect

  See Chapter 10, "Test Language Reference," MultiSelect, page 146.
- Radio

  Refer to the Radio step type entry for details.

  See Chapter 10, "Test Language Reference," Radio, page 153.
- Span

  See Chapter 10, "Test Language Reference," Span, page 160.
- Text

  See Chapter 10, "Test Language Reference," Text, page 163.

Parameters

Parameter	***Description***
prop=*value*;	The property name.

Parameter	*Description*
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

GetLabel

Description

Gets the label for the specified HTML object.

This is the list of objects that support this action:

- CheckBox

  See Chapter 10, "Test Language Reference," CheckBox, page 126.
- ComboBox

  See Chapter 10, "Test Language Reference," ComboBox, page 127.
- LongText

  See Chapter 10, "Test Language Reference," LongText, page 142.
- MultiSelect

  See Chapter 10, "Test Language Reference," MultiSelect, page 146.
- Pwd

  See Chapter 10, "Test Language Reference," Pwd, page 151.
- Radio

  Refer to the Radio step type entry for details.

  See Chapter 10, "Test Language Reference," Radio, page 153.

- RichText

  Refer to the RichText step type entry for details.

  See Chapter 10, "Test Language Reference," RichText, page 155.

- Span

  See Chapter 10, "Test Language Reference," Span, page 160.

- Text

  See Chapter 10, "Test Language Reference," Text, page 163.

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 126.

- ComboBox

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

- LongText

  Use the SetValue_InModal action to set the value of a long text field on a modal page.

  See Chapter 10, "Test Language Reference," SetValue_InModal, page 143.

  See Chapter 10, "Test Language Reference," LongText, page 142.

- MultiSelect

  See Chapter 10, "Test Language Reference," MultiSelect, page 146.

- Pwd

  See Chapter 10, "Test Language Reference," Pwd, page 151.

- Radio

  Refer to the Radio step type entry for details.

  See Chapter 10, "Test Language Reference," Radio, page 153.

- Text

  See Chapter 10, "Test Language Reference," Text, page 163.

Example

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

LongText	Set_Value	name=SQA_SIMPLEREC_SQA_DESCRIPTION	long
Text	Set_Value	name=SQE_SIMPLEREC_PSE_DESCR	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 126.
- ComboBox

  See Chapter 10, "Test Language Reference," ComboBox, page 127.
- LongText

  See Chapter 10, "Test Language Reference," LongText, page 142.
- MultiSelect

  See Chapter 10, "Test Language Reference," MultiSelect, page 146.
- Radio

  Refer to the Radio step type entry for details.

  See Chapter 10, "Test Language Reference," Radio, page 153.
- Span

  See Chapter 10, "Test Language Reference," Span, page 160.
- Text

  See Chapter 10, "Test Language Reference," Text, page 163.

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 in the first step of the previous example. The test case that verifies the value as KUSPTEST would be constructed as shown in the following example:

Text	Set_Value	ID=PA_CLC_SUMMARY_CALC_NAME	#CHECK#KUSPTEST

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

#DIS#

Description

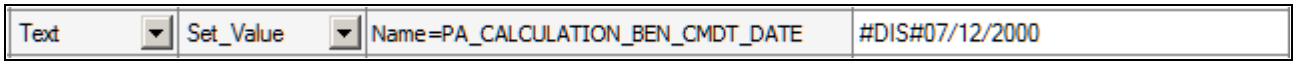
This reserved word verifies a value and also checks whether the object is display-only. It logs a Fail if the object is not display-only or if the expected value does not match the application value.

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:



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 180

#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 178

#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 181

#IGNORE

Description

Place the #IGNORE reserved word in the Value field of a Test.Exec step to skip the call to the child test. Suppose you have a parent test with two test cases. In the first test case, the parent test calls a child test. In the second test case, the parent does not call the child. Use the #IGNORE reserved word in the Value field with the second test case to skip calling the child for that test case.

In this example, the first step for the test case calls the test CHILD_ONE with test case CASE_01. The second step skips the call to CHILD_TWO for this test case.

Test	Exec	COPY_USER_PROFILE	DEFAULT, CASE_01, CASE_02
Test	Exec	PROCESS	#IGNORE

Example of the #IGNORE reserved word

#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, &

Type of Match	*Pattern*	*Match (Log a Pass)*	*No Match (Log a Fail or Warn)*
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 176

#LIST#

Description

This reserved word checks the values of a ComboBox. It works with either the full text entries in the combo box or the metadata translation (XLAT) values of the entries.

Use #LIST# with a Set_Value action to check one or more values and then set an item in a drop-down list box, list the items separated by a vertical pipe (|) and place brackets ([]) around the item that you want to select. If the value in the field is not the same as the value in brackets, PTF logs an error and sets the value in the field to the value in brackets.

Example

This example shows the use of the #LIST# reserved word:

In this example, the first step verifies the existence of items in the list. The second step verifies that the items exist and verifies that Individual is selected.

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 174

#NOTHING

Description

PTF ignores any step with a Set_Value or Verify action where the Value field is empty, or blank. The #NOTHING reserved word enables you to use SET_VALUE to set a field to blank or select a blank value from a drop-down list box. You can use #NOTHING with a Verify action to verify that a field is blank.

The #NOTHING reserved word does not have a closing pound sign (#). It cannot be used in combination with other reserved words.

Note. Leaving the Value field of a test step blank does not have the same effect as using the #NOTHING reserved word. PTF ignores any Set_Value or Verify action where the Value field is blank.

Example

In the following example, in the first step #NOTHING selects a blank value in the Calculation Reason field and then, in the next step, it verifies that the field is blank:

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 175

System Variables

System variables are populated by PTF at runtime. The following table lists PTF system variables.

Variable	*Description*
%case%	Current test case name.
%component%	Current component name.
%env.appserver%	Application server name.
%env.database%	Database name and database type. For example: T852U11 /DB2
%eo.dbname%	Execution option database name.
%eo.dbserver%	Execution option database server name.
%eo.dbuser%	Execution option database user name.
%eo.name%	Current execution option name.
%eo.platform%	Execution option platform.
%eo.pserver%	Execution option process server.
%eo.pshome%	Execution option PS_HOME path.
%env.toolsrel%	The PeopleTools version.
%eo.verbose%	Execution option verbose flag.

Variable	*Description*
%frame%	The current frame name.
%log.id%	Current log number.
%page%	Current page name.
%pid%	The last process instance number.
%test%	Current test name.

Functions

This section lists the PTF functions.

Add

Syntax

Add(*number1*|*number2*[|*number3*]...)

Description

Use the Add function to add a series of numbers. Decimals and negative numbers are supported.

Parameters

Parameter	*Description*
number1	Number to be added.
number2	Number to be added.
number3…	(Optional) A series of additional numbers to be added.

Returns

Sum of the numbers in the parameters.

Example

The following table presents examples of using Add.

Expression	*Result*
Add(10| -12| -3| 4| 6)	5
Add(10.43| 10.55| -6.789| -178)	-163.809

Concat

Syntax

Concat(*string1*|*string2*[|*string3*…])

Description

Concatenates the strings in the parameters.

Parameters

Parameter	*Description*
string1	Beginning string for concatenation
string2	A string to be concatenated to string1.
string3…	Additional strings to be concatenated.

Returns

Returns a string resulting from concatenating the strings in the parameters.

Example

The following table presents examples of using the concat function:

Expression	*Result*
concat(hello |and |welcome |to |PTF)	hello and welcome to PTF

Date

Syntax

Date()

Description

Returns the current date, using the date format specified in the current execution option.

Returns

The current date.

Example

The following table presents example of using the Date function:

Expression	***Result***
Date()	07/04/2011

Day

Syntax

Day*(date_value)*

Description

Returns the day portion of the date value provided as a parameter.

Parameters

Parameter	***Description***
date_value	A date value, such as the value returned by the Date() function.

Returns

Returns the day portion of the date value provided as a parameter.

Example

The following table presents examples of using the Day function:

Expression	*Result*
Day(February 13, 2012)	13
Day(Date())	13

Divide

Syntax

Divide(*number1*|*number2*[|*dec=dec_places*])

Description

Use the Divide function to perform division. Decimals and negative numbers are supported. Optionally specify the number of decimal places.

Parameters

Parameter	*Description*
number1	The dividend.
number2	The quotient.
dec=dec_places	(Optional) The number of decimal places. The maximum is 10. The default is zero. Note that "dec=" must be included in the parameter.

Returns

The result of dividing number1 by number2 rounded to dec_places decimals.

Example

The following table presents examples of using the Divide function:

Expression	*Result*
Divide(75| 13.5)	6

Expression	Result
Divide(-75| 13.5| dec=5)	-5.55556

GetField

Syntax

GetField*(string,segment,delimiter)*

Description

GetField returns the substring from a specified segment of a character-delimited text string.

Parameters

Parameter	Description
string	A character-delimited text string.
segment	An integer specifying which segment of the string will be returned, counting left to right. Specify a negative integer to count right to left.
delimiter	The character that delimits each segment in the string.

Returns

Returns the substring between the delimiters in the specified segment of the string.

Example

The following table presents examples of using GetField.

Expression	Result
GetField(a/b/c| 1| /)	a
GetField(a/b/c| 2| /)	b
GetField(a/b/c| 5| /)	blank
GetField(a/b/c| -1|/)	c

Hour

Syntax

Hour*(time_value)*

Description

Returns the hour portion of the time value provided as a parameter.

Parameters

Parameter	*Description*
time_value	A time value, such as the value returned by the Time() function.

Returns

Returns the hour portion of the time value provided as a parameter.

Example

The following table presents examples of using the Hour function:

Expression	*Result*
Hour(13:07:25)	13
Hour(Time())	13

InStr

Syntax

InStr*(within_string|string)*

Description

Locates a substring within a string of text and returns the starting position of the substring as an integer..

Parameters

Parameter	*Description*
substring	The text you are searching for. The string parameter is not case sensitive.
within_string	The text string you are searching within.

Returns

Returns an integer indicating the starting position of substring in within_string.

InStr returns 0 if substring does not appear in within_string. It returns 1 if substring is empty.

Example

The following table presents examples of using the InStr function.

Expression	*Result*
instr(ABCDEFG|c)	3
instr(ABCDEFG|C)	3
instr(ABCDEFG|CDE)	3
instr(ABCDEFG|zz)	0
instr(ABCDEFG|)	1
instr(ABCDEFG|A)	1

LCase

Syntax

LCase*(string)*

Description

Converts a string to lowercase.

Parameters

Parameter	*Description*
string	The string to be converted.

Returns

Returns a string resulting from converting string to lowercase.

Example

The following table presents example of using the LCase function.

Expression	*Result*
lcase(Hello World 1234)	hello world 1234

Left

Syntax

Left*(string|length)*

Description

Extracts a substring of a specified number of characters from the left side of a string.

Parameters

Parameter	*Description*
string	A string from which to extract a substring.
length	A number specifying the number of characters in the substring.

Returns

Returns a substring length characters long from the left side of a string.

Example

The following table presents example of using Left function.

Expression	*Result*
left(Hello World|5)	Hello

Len

Syntax

Len*(string)*

Description

Returns the length of string as an integer.

Parameters

Parameter	*Description*
string	A text string.

Returns

Returns an integer indicating the length of string.

Example

The following table presents example of using Len function.

Expression	*Result*
len(Hello World)	12

MakeDate

Syntax

MakeDate*(year_value | month_value | day_value)*

Description

This function returns a date value based on the year, month, and day values passed to the function as parameters.

Parameters

Parameter	*Description*
year_value	A number representing the year, such as the value returned by the Year() function.
month_value	A number representing the month, such as the value returned by the Month() function.
day_value	A number representing the day, such as the value returned by the Day() function.

Returns

Returns a date value.

Example

The following table presents examples of using the Day function:

Expression	*Result*
MakeDate(Add(Year(Date())|1)|Add(Month(Date())|11)|Add(Day(Date())|1)	January 14, 2014
MakeDate(Add(Year(Date())|1)|Add(Month(Date())|-1)|Add(Day(Date())|-1)	January 12, 2013

MakeTime

Syntax

MakeTime*(hour_value | minute_value | second_value | rollover_boolean)*

Description

This function returns a time value based on the hour, minute, and second values passed to the function as parameters.

Parameters

Parameter	*Description*
hour_value	A number representing the hour, such as the value returned by the Hour() function.
minute_value	A number representing the minute, such as the value returned by the Minute() function.
second_value	A number representing the second, such as the value returned by the Second() function.
rollover_boolean	(Optional) Rolls over the hour to 0 when hour_value reaches 24. True - Returns (hour_value – 24) when hour_value is greater than 24. False - Returns hour_value even when hour_value is greater than 24. The default value is True.

Returns

Returns a time value.

Example

The following table presents examples of using the Day function:

Expression	*Result*
MakeTime(Add(Hour(Time())|12)|Add(Minute(Time())|-30)|Second(Time()))	7:00:00 AM
MakeTime(23|Add(60|30)|0|False)	24:30:00
MakeTime(23|Add(60|30)|0|True)	00:30:00
MakeTime(23|Add(60|-30)|0)	23:30:00

Minute

Syntax

Minute*(time_value)*

Description

Returns the minute portion of the time value provided as a parameter.

Parameters

Parameter	*Description*
time_value	A time value, such as the value returned by the Time() function.

Returns

Returns the minute portion of the time value provided as a parameter.

Example

The following table presents examples of using the Minute function:

Expression	*Result*
Minute(13:07:25)	7
Minute(Time())	7

Month

Syntax

Month*(date_value)*

Description

Returns the month portion of the date value provided as a parameter.

Parameters

Parameter	*Description*
date_value	A date value, such as the value returned by the Date() function.

Returns

Returns the month portion of the date value provided as a parameter.

Example

The following table presents examples of using the Month function:

Expression	*Result*
Month(February 13, 2012)	2
Month(Date())	2

Multiply

Syntax

Multiply(*number1*|*number2*[|*dec=dec_places*])

Description

Use the Multiply function to perform multiplication. Decimals and negative numbers are supported. Optionally specify the number of decimal places.

Parameters

Parameter	*Description*
number1	First factor.
number2	Second factor.
dec_places	(Optional) The number of decimal places. The maximum is 10. The default is zero. Note that "dec=" must be included in the parameter.

Returns

The result of multiplying number1 by number2 rounded to dec_places decimals.

Example

The following table presents examples of using Multiply function.

Expression	*Result*
Multiply(10.3| 13.45)	139
Multiply(10.3| -13.45|dec=3)	-138.535

Now

Syntax

Now()

Description

Returns the current datetime, using the date format specified in the current execution option.

Returns

The current datetime.

Example

The following table presents example of using Now function.

Expression	*Result*
Now()	07/04/2011 12:20 PM

Replace

Syntax

Replace*(source\|find\|replace)*

Description

Use the Replace function to replace every occurrence of a substring found in a string with a new substring.

Parameters

Parameter	*Description*
source	A string in which you want to replace substrings.
find	A string equal to the substring of source you want to replace.
replace	A string with which to replace occurrences of find in source.

Returns

Returns a string resulting from replacing every occurrence of find found in source with replace.

Example

The following table presents example of using Replace function.

Expression	*Result*
replace(original text|i|77)	Or77g77nal text

Right

Syntax

Right*(string|length)*

Description

Use the Right function to extract a substring of a specified number of characters from the right side of a string.

Parameters

Parameter	*Description*
string	A string from which to extract a substring.
length	A number specifying the number of characters in the substring.

Returns

Returns a substring length characters long from the right side of a string.

Example

The following table presents example of using Replace function.

Expression	*Result*
right(Hello World|5)	World

Round

Syntax

Round(*number*|[|*dec=dec_places*])

Description

Use the Round function to round a number. Decimals and negative numbers are supported. Optionally specify the number of decimal places.

Parameters

Parameter	*Description*
number1	First factor.
number2	Second factor.
dec_places	(Optional) The number of decimal places. The maximum is 10. The default is zero. Note that "dec=" must be included in the parameter.

Returns

The result of rounding number1 to dec_places decimal places.

Example

The following table presents example of using Round function.

Expression	*Result*
Round(-130.456)	-130
Round(-130.456|dec=2)	-130.46
Round(-130.455|dec=2)	-130.45

Second

Syntax

Second*(time_value)*

Description

Returns the second portion of the time value provided as a parameter.

Parameters

Parameter	*Description*
time_value	A time value, such as the value returned by the Time() function.

Returns

Returns the second portion of the time value provided as a parameter.

Example

The following table presents examples of using the Second function:

Expression	*Result*
Second(13:07:25)	25
Second(Time())	25

SubStr

Syntax

SubStr(*source_str*|*start_pos*[|*length*])

Description

Extracts a substring of a specified number of characters beginning at a specified location in a source string.

If length is not specified, SubStr returns the substring starting at the position specified in start_pos and continuing to the end of the string.

Parameters

Parameter	***Description***
source_str	A string from which to extract a substring.
start_pos	A number representing the character position in source_str where the substring starts, starting at 1.
length	A number specifying the number of characters in the substring.

Returns

Returns a string equal to a substring length characters long beginning at character start_pos of source_str.

Example

The following table presents examples of using SubStr function.

Expression	***Result***
substr(12345678|2|3)	234
substr(12345678|2)	2345678

Subtract

Syntax

Subtract(*number1*|*number2*[|*number3*]...])

Description

Use the Subtract function to subtract a series of numbers. Numbers can be decimal and negative.

Parameters

Parameter	*Description*
number1	Initial number.
number2	Number to be subracted.
number3…	(Optional) A series of additional numbers to be subtracted.

Returns

The result of subtracting number2, number3, etc., from number1.

Example

The following table presents examples of using Subtract function.

Expression	*Result*
Subtract(10|2|3)	5
Subtract(10|-2|-3)	15

Sum

Syntax

Sum,Index, *value,delimiter*)

Description

Sum works with the HTMLTable indexes.

Parameters

Parameter	*Description*
Index	The HTMLTable index string, such as 2/5/4. An index string is the return value of CellGetIndex. See Chapter 10, "Test Language Reference," CellGetIndex, page 136.
Value	The value that you want to add or subtract. The default action is addition.
Section	The section of the index that will be modified.
Delimiter	The character that delimits each section in the text value.

Example

The following table presents examples of using Sum.

Expression	*Result*
Sum(2/5/4, 2, 1, /)	4/5/4 2 is added to the first section of the string.
Sum(2/5/4, -1, 3, /)	2/5/3
Sum(&index, -4, 3, /)	4 is subtracted from the third section of the string in the variable &index.

Time

Syntax

Time()

Description

Returns the current time, using the date format specified in the current execution option.

Parameters

None

Returns

Returns a string with the current time.

Example

The following table presents example of using the Time function with the Regional Options set to 24 hour format and 12 hour format, respectively:

Expression	*Result*
Time()	13:07:25
Time()	1:07:25 PM

Trim

Syntax

Trim*(string)*

Description

Returns a string with all leading and trailing spaces removed.

Parameters

Parameter	*Description*
string	A text string.

Returns

Returns a string with all leading and trailing spaces removed.

Example

The following table presents example of using trim function.

Expression	*Result*
trim(Hello World)	Hello World

UCase

Syntax

UCase*(string)*

Description

Converts a string to uppercase.

Parameters

Parameter	*Description*
string	The string to be converted.

Returns

Returns a string resulting from converting string to uppercase.

Example

The following table presents example of using UCase function.

Expression	*Result*
ucase(Hello World 1234)	HELLO WORLD 1234

Year

Syntax

Year*(date_value)*

Description

Returns the year portion of the date value provided as a parameter.

Parameters

Parameter	*Description*
date_value	A date value, such as the value returned by the Date() function.

Returns

Returns the year portion of the date value provided as a parameter.

Example

The following table presents examples of using the Year function:

Expression	*Result*
Year(February 13, 2012)	2012
Year(Date())	2012

Chapter 11

Test Language Reference

This chapter discusses:

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

Step Types

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

Browser

These are the actions associated with the Browser step type.

Close

Description

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

FrameSet

Description

Sets the focus in a browser frame.

Embedded frames include a number, such as ptModFrame_1. You can substitue ## for the number, for example ptModFrame_## and PTF will search through all frames starting with ptModFrame_ and use the one with the highest number.

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 step type.

Click

Description

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

See Chapter 10, "Test Language Reference," Click, page 166.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

Chart

These are the actions associated with the Chart step 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

Returns the text value for the specified 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 stept 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 index 2 is &chart_val;	

Example of GetText action for a Chart step type

CheckBox

These are the actions associated with the CheckBox step type.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

Set_Value

Description

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

See Chapter 10, "Test Language Reference," Set_Value, page 171.

Verify

Description

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

See Chapter 10, "Test Language Reference," Verify, page 172.

ComboBox

These are the actions associated with the ComboBox step type.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

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.

When used with #LIST#, Set_Value returns an error if the expected value (the bracketed value) is not the same as the actual value.

Example

This example sets the field value to French and verifies that the values English, French, and Finnish exist in the list.

ComboBox	Set_Value	Name=PSOPRDEFN_LANGUAGE_CD	#LIST#English|French|Finnish

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

See Also

Chapter 10, "Test Language Reference," #LIST#, page 178

Verify

Description

Compares the value in the browser to the expected value and adds a Pass or Fail log entry for the validation. Use a vertical pipe (|) to separate values to be verified. Use square brackets ([]) to specify which value is expected to be selected.

Example

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

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

Example of Verify action

Conditional

You can control the flow of execution of tests using conditional constructs. A conditional construct begins with an If_Then action and ends with an End_If action. You can optionally include an Else action.

Conditional constructs can be nested.

These are the actions associated with the Conditional step type.

If_Then

Description

The first step in a conditional contstruct. The system evaluates the logical expression in the Recognition field of the If_Then step. If the expression evaluates to True, the system executes the steps between the If_Then step and the End_If step or the Else step, if it exists. If the expression evaluates to False, the system jumps to the Else step, if it exists, or to the End_If step if there is no Else, and continues execution.

If_Then supports these logical operators:

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

You can use the AND and OR logical operators to specify multiple conditions.

Else

Description

(Optional) If the logical expression evaluates to False, the system executes the steps between the Else step and the End_If step.

End_If

Description

The close statement of the If_Then construct.

Example

This example shows the use of multiple conditions and nested conditionals:

Variable	Set_Value	&Integer	3
Conditional	If_Then	&Integer>=1 AND &Integer<=10 OR &Integer=0	
Log	Message	Determine whether Integer is odd or even	
Log	Message	Enter first nested conditional	
Conditional	If_Then	&Integer = 0	
Log	Message	Do not divide by 0.	
Conditional	Else		
Variable	Set_Value	&Half=Divide(&Integer|2|dec=1)	
Log	Message	Enter second nested conditional	
Variable	Set_Value	&Odd=Substr(&Half|3)	
Conditional	If_Then	&Odd = 5	
Log	Message	The number is odd	
Conditional	Else		
Log	Message	The number is even	
Conditional	End_If		
Conditional	End_If		
Conditional	End_If		

Example of the If_Then conditional construct

DataMover

This is the action associated with the DataMover step 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 path is not set in the execution options, it uses the system temp folder.

Example

This example shows the use of the Exec action:

Example of the DataMover Exec action

Div

This is the action associated with the Div step type.

Click

Description

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

See Chapter 10, "Test Language Reference," Click, page 166.

Execution

Use the Execution actions in shell tests to modify the behavior of tests during execution. You typically set these options as you are developing tests to facilitate the development process.

The Execution step type is available only in shell tests.

Set_Options

Description

Override the default execution option. Specify the name of a valid execution option in the Recognition column.

Skip_PageSave

Description

Specify *True* or *False* in the Recognition column. Specify True to skip Page.Save steps. You would, for instance, select this option to avoid creating duplicate values if you plan to run a test repeatedly.

This action overrides the Skip PageSave setting in Execution Options.

Skip_RunRequest

Description

Specify *True* or *False* in the Recognition column. Specify True to skip Process.Run steps.

This action overrides the Skip RunRequest setting in Execution Options.

Parameter	***Description***
True	Skip Process.Run steps.
False	Execute Process.Run steps.

Skip_Login

Description

Specify*True* or *False* in the Recognition column. Specify True to skip Browser.Start_Login steps.

Parameter	***Description***
True	Skip Browser.Start_Login steps.
False	Execute Browser.Start_Login steps.

StopOnError

Description

Specify *True* or *False* in the Recognition column. Specify True to stop execution when a Fail is logged.

This action overrides the Stop on Error setting in Test Editor.

Parameter	*Description*
True	Stop execution when a test encounters a Fail.
False	Continue execution when a test encounters a Fail.

See Also

Chapter 2, "Installing and Configuring PTF," Configuring Execution Options, page 23

File

This is the action associated with the File step type. The File step type corresponds to the object in PIA that allows users to upload files to the PeopleSoft application.

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:

Example of the File Upload action

HTMLTable

These are the actions associated with the HTMLTable step type.

CellClick

Description

Clicks on a specific HTMLTable cell based on the index parameter.

Parameter	*Description*

index=*I*/*R*/*C*:	The table, row, column index string.
	For example:
	`index=&CellIndex`
	`index=1/2/3`
	`;`
	In the second example, CellClick clicks on the third column of the second row of the first table.

CellClickOnChkB

Description

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

Parameter	*Description*

index=*I*/*R*/*C*:	The table, row, column index string.
	For example:
	`index=&CellIndex`
	index=1/2/3;
	In the second example, this function clicks on a checkbox within the third column of the second row of the first table.

chkidx=*value*;	The CheckBox object index inside the cell.
check=*value*;	`check=Y` – Select the checkbox. `check=N` – Clear the checkbox. This parameter is optional. The default value is Y.

CellClickOnImage

Description

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

Parameters

Parameter	*Description*

index=*I*/*R*/*C*:	The table, row, column index string.
	For example:
	`index=&CellIndex`
	`index=1/2/3;`
	In the second example, this function clicks on an image within 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=*I*/*R*/*C*:	The table, row, column index string.
	For example:
	`index=&CellIndex`
	`index=1/2/3;`
	In the second example, this function clicks on a link within 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=*I*/*R*/*C*:	The table, row, column index string.
	For example:
	`index=&CellIndex`
	`index=1/2/3;`
	In the second example, this function verifies whether a cell exists within the third column of the second row of the first table.

ret=*&variable*	The return value. True – the cell exists. False – the cell does not exist.

CellGetIndex

Description

Searches the page for the text value specified in the text parameter and returns the index string for the first cell that contains the text. The index string is in the form of *I/R/C*, where *I* is the table index, *R* is the row number, and *C* is the column number. Other actions, such as CellClick, CellGetValue, and so on, use an index string to reference a specific cell.

Parameters

Parameter	*Description*
text=*value*;	The text to look for on the page.
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.

Parameter	Description
ret=*&variable*	The return value is an index string in the form of I/R/C, where I is the table index, R is the row number, and C is the column number. For example: `ret=&CellIndex`

CellGetValue

Description

Returns the contents of an HTMLTableCell.

Parameters

Parameter	Description

index=*I*/*R*/*C*:	The table, row, column index string.
	For example:
	`index=&CellIndex`
	`index=1/2/3;`
	In the second example, this function returns the contents of the third column of the second row of the first table.

ret=*&variable*	The return value.

ColCount

Description

Returns the number of columns for the HTMLTable row.

Parameters

Parameter	Description
table=*value*;	The table index.
row=*value*;	The row index.

Parameter	*Description*
index=*I*/*R*/*C*:	An index string in the form of I/R/C, where I is the table index, R is the row number, and C is the column number. As an alternative to specifying the table and row parameters, you can specify an index string, such as the return value from a CellGetIndex action. For example: `index=&CellIndex;`
ret=*&variable*	The return value.

RowCount

Description

Returns the number of rows for the HTMLTable.

Parameters

Parameter	*Description*
table=*value*;	The table index.
index=*I*/*R*/*C*:	An index string in the form of I/R/C, where I is the table index, R is the row number, and C is the column number. As an alternative to specifying the table parameter, you can specify an index string, such as the return value from a CellGetIndex action. For example: `index=&CellIndex;`
ret=*&variable*	The return value.

Example

This is an example of several of the HTMLTable actions:

Browser	Start_Login		
HTMLTable	CellGetIndex	text=My Reports; ret=&HTMLIndex; equal=true	
Conditional	If_Then	&HTMLIndex=0/0/0	
Link	Click	id=pttabpercontent	
HTMLTable	CellGetIndex	text=My Reports; ret=&HTMLIndex; equal=true	
Variable	Set_Value	&CheckIndex	sum(&HTMLIndex, -1, 3, /)
HTMLTable	CellClick	index=&CheckIndex	
Page	Save		
Conditional	End_If		

Example of actions for the HTMLTable step type

Image

These are the actions associated with the Image step type.

Click

Description

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

See Chapter 10, "Test Language Reference," Click, page 166.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

RightClick

Description

Performs a right-click on the image. This action is supported only for a related-content image glyph.

Link

These are the actions associated with the Link step type.

Click

Description

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

See Chapter 10, "Test Language Reference," Click, page 166.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

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.

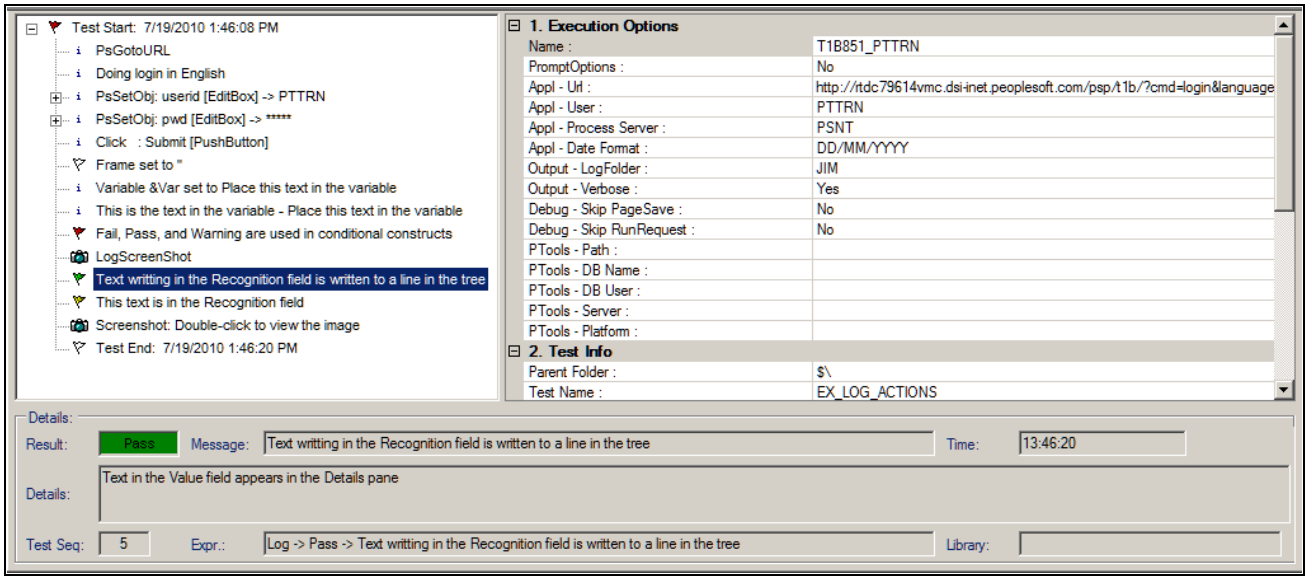
Example

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 step type.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

Set_Value

Description

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

See Chapter 10, "Test Language Reference," Set_Value, page 171.

SetValue_InModal

Description

Use the SetValue_InModal action to set the value of a long text field on a modal page.

Verify

Description

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

See Chapter 10, "Test Language Reference," Verify, page 172.

Loop

These are the actions associated with the Loop step type.

Do

Description

Executes the steps between the Loop.Do step and the Loop.End_Loop until a Loop.Exit step is encountered.

Example

This example shows the Loop.Do construct:

Variable	Set_Value	&Var = 0	
Loop	Do		
Log	Message	The variable is &Var	
Conditional	If_Then	&Var > 3	
Loop	Exit_Loop		
Conditional	End_If		
Loop	End_Loop		

Example of a Loop.Do construct

End_Loop

Description

Terminates a Loop.Do or Loop.While construct.

Exit

Description

Exits a Loop.For or Loop.While construct. Execution continues with the step following the End_Loop step. Typically, a Loop.Exit step is placed within a conditional construct.

While

Description

Executes the steps between the Loop.While step and the Loop.End_Loop while the expression in the Recognition field evaluates to True.

When the expression evaluates to false, flow skips to the step following the End_Loop step.

Example

The following example shows the Loop.While construct:

Variable	Set_Value	&Var = 0	
Loop	While	&Var <=3	
Log	Message	The variable is &Var	
Variable	Set_Value	&Var = add(&Var;|1)	
Loop	End_Loop		

Example of Loop.While

For

Syntax

&variable=*begin_value* to *end_value*;

Description

Executes the steps between the Loop.For step and the Loop.Next step until the expression in the Recognition field evaluates to False, then flow skips to the step following the Loop.Next step.

Parameters

Parameter	***Description***
&variable	The variable to be used in the comparison. This variable is incremented in the Loop.Next step.
begin_value	The starting value.
end_value	The ending value.

Next

Description

Terminates a Loop.For construct. Loop.Next increments the variable in the Loop.For step.

See Chapter 10, "Test Language Reference," Add, page 182.

Example

The following example shows the Loop.For construct:

Variable	Set_Value	&Var = 0	
Loop	For	&Var; 1 to 5	
Log	Message	The variable is &Var	
Loop	Next		

Example of a Loop.For construct

MultiSelect

These are the actions associated with the MultiSelect step type.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

Set_Value

Description

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

See Chapter 10, "Test Language Reference," Set_Value, page 171.

Verify

Description

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

See Chapter 10, "Test Language Reference," Verify, page 172.

Page

These are the actions associated with the Page step type.

Expand

Description

Attempts to expand all the collapsed sections on the current page.

Go_To

Description

Accesses a page by selecting a page tab. Enter the 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:

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.

SecPage_Close

Description

Closes the secondary page.

SecPage_Open

Description

Opens a secondary page.

Parameters

Parameter	***Description***
page= *value*	Specify the name of the secondary page.

Example

The following example shows the SecPage_Open action:

Example SecPage_Open

Process

The Process actions run processes through Process Scheduler.

Run

Description

Runs a Process Scheduler process.

Parameters

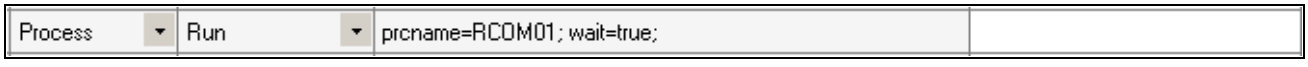
Parameter	*Description*
prcname=*value*;	The process name.
prctype=*value*:	The process type.
wait=*value*;	True - the test waits for the process to finish. False - the test does not wait for the process to finish. The default is False.
outtype=*value*;	The process output type.
outformat=*value*;	The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

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:



Example of the Run action

Run_Def

Description

Changes the default behavior of the run process. Insert these steps prior to the Process Run step that they modify. 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.
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 22

Pwd

These are the actions associated with the Pwd step type.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

Set_Value

Description

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

See Chapter 10, "Test Language Reference," Set_Value, page 171.

Query

This is the action associated with the Query step type.

Exec

Description

Runs a public query in PeopleSoft Query and downloads the results. To run a private query, use the Exec_Private action.

Exec_Private

Description

Runs a private query in PeopleSoft Query and downloads the results.

Parameter	*Description*
outFile=*value*;	The query output file.
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.
0rows=*value*;	If the query returns zero rows, add a log entry of type value. Valid values are Pass, Fail, or Warning.
Nrows=*value*;	If the query returns one or more rows, add a log entry of type value. Valid values are Pass, Fail, or Warning.

Use the following formats to specify query parameters:

Page Control	*Format*
Text	param=*value*
Radio button	param={RB}*value*
Combo box	param={CB}*value*
Check box	param={CH*}value*
Text box	param={EB}*value*

Radio

These are the actions associated with the Radiostep type.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

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 Recognition field using the `Value=` parameter.

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 step type

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

Set_Value

Description

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

See Chapter 10, "Test Language Reference," Set_Value, page 171.

Verify

Description

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

See Chapter 10, "Test Language Reference," Verify, page 172.

RichText

These are the actions associated with the RichText step type.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

Set_Value

Description

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

See Chapter 10, "Test Language Reference," Set_Value, page 171.

Scroll

These are the actions associated with the Scroll step type.

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

See Also

Chapter 5, "Developing and Debugging Tests," Incorporating Scroll Handling, page 71

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.
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 override the default name object of scroll buttons and the scroll parent. This is necessary, for example, when a page uses custom objects (such as links or pushbuttons) to handle conventional scroll actions such as adding or deleting rows from the scroll.

Parameters

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

type=*value*;	Thestep 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	Override the Add new row button.
del	Override the Delete row button.
next	Override the Next button.
prev	Override the Previous button.
first	Override the First button.
last	Override the Last button.
parent	Reassign the parent for a specific scroll area.

Value	*Description*
parent	Reassign the parent for a specific scroll area.

ModalGrid_Close

Description

Closes a modal grid.

ModalGrid_Open

Description

Opens a modal grid.

Key_Set

Description

Defines each key field of a scroll. Use multiple Key_Set steps for scrolls with multiple key fields.

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

Parameters

Parameter	*Description*
type=value;	The step 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 example uses the Definition action with the `def=Add` parameter to assign the Add action to the image $ICField3$newm$0$$mg$0:

1	Scroll	Definition	def=Add; Type=Image; Value=Name=$ICField3$newm$0$$mg$0	

Example of a Definition action with def=Add parameter

In this example the Definition action with a `def=parent` parameter reassigns the parent:

1	Scroll	Key_Set	Type=Text; Name=PSU_EMP_REVIEW_REVIEW_YEAR	2009
1	Scroll	Action	Ret=&Scr1	upd
	Text	Set_Value	Name=PSU_EMP_REVIEW_REVIEW_DAYS&Scr1	365
2	Scroll	Definition	def=parent; value=&Scr1	
2	Scroll	Key_Set	Type=ComboBox; Name=PSU_EMP_RVW_RVR_REVIEWER_ID	00013
2	Scroll	Action	ret=&Scr2	upd
	ComboBox	Set_Value	Name=PSU_EMP_RVW_RVR_REVIEW_TYPE&Scr2	S

Example of the Definition action with the def=parent paramenter

Span

These are the actions associated with the Span step type.

Click

Description

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

See Chapter 10, "Test Language Reference," Click, page 166.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

MouseOver

Description

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

See Also

PeopleTools 8.52: 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 8.52: 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 record a verification of a field value in a mouseover popup window:

1. From the recorder tool bar, click and drag the Verify icon 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 Verify icon 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 172.

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 step type.

Exec

Description

Calls another test or library test.

Specify the child test name in the Recognition field and one or more test case names in the Value field, separated by commas.

You can also click in the Recognition field then click the elipsis icon and select a test case to automically populate the Recognition and Value fields with the test name and the test case name.

Test.Exec supports the use of parameters that enable you to pass dynamic values from a calling test to a library test.

Right-click the test name in the Recognition field and select Open Test to open the child test.

Use the #IGNORE reserved word in the Value field to skip the call to the child test for a given test case.

See Chapter 5, "Developing and Debugging Tests," Using Parameters with Library Tests, page 76.

Example

This test calls the test COPY_USER_PROFILE with the DEFAULT, CASE_01, and CASE_02 test caseS, then skips the call to PROCESS:

Test	Exec	COPY_USER_PROFILE	DEFAULT, CASE_01, CASE_02
Test	Exec	PROCESS	#IGNORE

Example of the Test Exec action

Text

These are the actions associated with the Text step type.

Exists

Description

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

See Chapter 10, "Test Language Reference," Exists, page 167.

Get_Property

Description

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

See Chapter 10, "Test Language Reference," Get_Property, page 168.

GetLabel

Description

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

See Chapter 10, "Test Language Reference," GetLabel, page 170.

Set_Value

Description

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

See Chapter 10, "Test Language Reference," Set_Value, page 171.

Verify

Description

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

See Chapter 10, "Test Language Reference," Verify, page 172.

Variable

This is the action associated with the Variable step 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 119

Wait

These are the actions associated with the Wait step type.

For_Seconds

Description

Specify in the Recognition field the number of seconds PTF will wait before proceeding to the next step.

For_Value

Description

Wait until the field contains the value specified in the Value field.

Parameters

Parameter	*Description*
type= *value*	Specify the step type, such as Text, Combobox, Link, and so on.
timeout=*value*	[Optional] Specify the time out value, in seconds. The default is 300 seconds.
refresh=*value*	[Optional] Specify the refresh button name. If the refresh= parameter is specified, PTF waits five seconds between each validation process and the refresh click.

Example

This example shows the use of Wait:

Wait	For_Seconds	60	
Wait	For_Value	Type=Link;Name=STATUS;	
Wait	For_Value	Type=Link;Name=STATUS; Refresh=Name=RA_REFRESH_BTN; timeout=90;	

Example of the Wait actions

Common Actions

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

Click

Description

Performs a mouse click on the specified object.

This is the list of objects that support this action:

- Button

  See Chapter 10, "Test Language Reference," Button, page 124.

- Image

  See Chapter 10, "Test Language Reference," Image, page 139.

- Link

  See Chapter 10, "Test Language Reference," Link, page 140.

- Span

  See Chapter 10, "Test Language Reference," Span, page 160.

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 124.

- CheckBox

  See Chapter 10, "Test Language Reference," CheckBox, page 126.

- ComboBox

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

- Image

  See Chapter 10, "Test Language Reference," Image, page 139.

- Link

  See Chapter 10, "Test Language Reference," Link, page 140.

- LongText

  See Chapter 10, "Test Language Reference," LongText, page 142.

- MultiSelect

  See Chapter 10, "Test Language Reference," MultiSelect, page 146.

- Pwd

  See Chapter 10, "Test Language Reference," Pwd, page 151.

- Radio

  Refer to the Radio step type entry for details.

  See Chapter 10, "Test Language Reference," Radio, page 153.

- Span

  See Chapter 10, "Test Language Reference," Span, page 160.

- Text

  See Chapter 10, "Test Language Reference," Text, page 163.

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:

Example of the Exists action

Get_Property

Description

Gets the property value of an HTML object based on the `prop=value` parameter and assigns it to the variable in `ret=&variable`.

Use the HTML Browser feature of the Message tool to identify the properties and values of an object.

See Chapter 5, "Developing and Debugging Tests," Using the Message Tool, page 63.

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.

This is the list of objects that support this action:

- Button

  See Chapter 10, "Test Language Reference," Button, page 124.
- CheckBox

  See Chapter 10, "Test Language Reference," CheckBox, page 126.
- ComboBox

  See Chapter 10, "Test Language Reference," ComboBox, page 127.
- Image

  See Chapter 10, "Test Language Reference," Image, page 139.
- Link

  See Chapter 10, "Test Language Reference," Link, page 140.
- LongText

  See Chapter 10, "Test Language Reference," LongText, page 142.
- MultiSelect

  See Chapter 10, "Test Language Reference," MultiSelect, page 146.
- Radio

  Refer to the Radio step type entry for details.

  See Chapter 10, "Test Language Reference," Radio, page 153.
- Span

  See Chapter 10, "Test Language Reference," Span, page 160.
- Text

  See Chapter 10, "Test Language Reference," Text, page 163.

Parameters

Parameter	***Description***
prop=*value*;	The property name.

Parameter	*Description*
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

GetLabel

Description

Gets the label for the specified HTML object.

This is the list of objects that support this action:

- CheckBox

  See Chapter 10, "Test Language Reference," CheckBox, page 126.
- ComboBox

  See Chapter 10, "Test Language Reference," ComboBox, page 127.
- LongText

  See Chapter 10, "Test Language Reference," LongText, page 142.
- MultiSelect

  See Chapter 10, "Test Language Reference," MultiSelect, page 146.
- Pwd

  See Chapter 10, "Test Language Reference," Pwd, page 151.
- Radio

  Refer to the Radio step type entry for details.

  See Chapter 10, "Test Language Reference," Radio, page 153.

- RichText

  Refer to the RichText step type entry for details.

  See Chapter 10, "Test Language Reference," RichText, page 155.

- Span

  See Chapter 10, "Test Language Reference," Span, page 160.

- Text

  See Chapter 10, "Test Language Reference," Text, page 163.

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 126.

- ComboBox

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

- LongText

  Use the SetValue_InModal action to set the value of a long text field on a modal page.

  See Chapter 10, "Test Language Reference," SetValue_InModal, page 143.

  See Chapter 10, "Test Language Reference," LongText, page 142.

- MultiSelect

  See Chapter 10, "Test Language Reference," MultiSelect, page 146.

- Pwd

  See Chapter 10, "Test Language Reference," Pwd, page 151.

- Radio

  Refer to the Radio step type entry for details.

  See Chapter 10, "Test Language Reference," Radio, page 153.

- Text

  See Chapter 10, "Test Language Reference," Text, page 163.

Example

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

LongText	Set_Value	name=SQA_SIMPLEREC_SQA_DESCRIPTION	long
Text	Set_Value	name=SQE_SIMPLEREC_PSE_DESCR	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 126.
- ComboBox

  See Chapter 10, "Test Language Reference," ComboBox, page 127.
- LongText

  See Chapter 10, "Test Language Reference," LongText, page 142.
- MultiSelect

  See Chapter 10, "Test Language Reference," MultiSelect, page 146.
- Radio

  Refer to the Radio step type entry for details.

  See Chapter 10, "Test Language Reference," Radio, page 153.
- Span

  See Chapter 10, "Test Language Reference," Span, page 160.
- Text

  See Chapter 10, "Test Language Reference," Text, page 163.

Example

This is an example of the Verify action:

Example of the Verify action

Reserved Words

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

#CHECK#

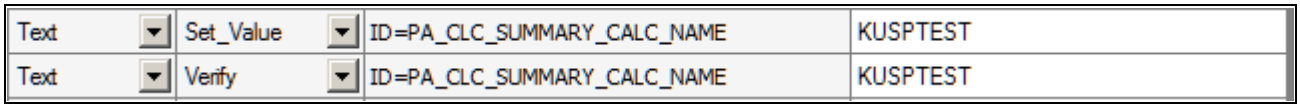
Description

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

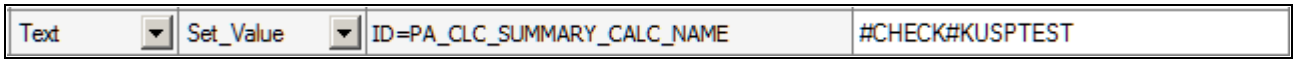
For example, a text box could be set and verified with the following two steps:



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 in the first step of the previous example. The test case that verifies the value as KUSPTEST would be constructed as shown in the following example:



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

#DIS#

Description

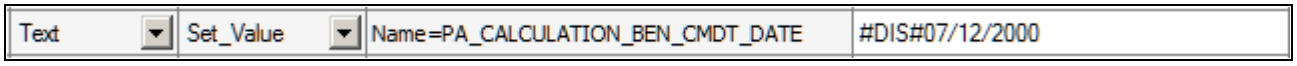
This reserved word verifies a value and also checks whether the object is display-only. It logs a Fail if the object is not display-only or if the expected value does not match the application value.

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:



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 180

#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 178

#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 181

#IGNORE

Description

Place the #IGNORE reserved word in the Value field of a Test.Exec step to skip the call to the child test. Suppose you have a parent test with two test cases. In the first test case, the parent test calls a child test. In the second test case, the parent does not call the child. Use the #IGNORE reserved word in the Value field with the second test case to skip calling the child for that test case.

In this example, the first step for the test case calls the test CHILD_ONE with test case CASE_01. The second step skips the call to CHILD_TWO for this test case.

Test	Exec	COPY_USER_PROFILE	DEFAULT, CASE_01, CASE_02
Test	Exec	PROCESS	#IGNORE

Example of the #IGNORE reserved word

#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, &

Type of Match	*Pattern*	*Match (Log a Pass)*	*No Match (Log a Fail or Warn)*
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 176

#LIST#

Description

This reserved word checks the values of a ComboBox. It works with either the full text entries in the combo box or the metadata translation (XLAT) values of the entries.

Use #LIST# with a Set_Value action to check one or more values and then set an item in a drop-down list box, list the items separated by a vertical pipe (|) and place brackets ([]) around the item that you want to select. If the value in the field is not the same as the value in brackets, PTF logs an error and sets the value in the field to the value in brackets.

Example

This example shows the use of the #LIST# reserved word:

In this example, the first step verifies the existence of items in the list. The second step verifies that the items exist and verifies that Individual is selected.

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 174

#NOTHING

Description

PTF ignores any step with a Set_Value or Verify action where the Value field is empty, or blank. The #NOTHING reserved word enables you to use SET_VALUE to set a field to blank or select a blank value from a drop-down list box. You can use #NOTHING with a Verify action to verify that a field is blank.

The #NOTHING reserved word does not have a closing pound sign (#). It cannot be used in combination with other reserved words.

Note. Leaving the Value field of a test step blank does not have the same effect as using the #NOTHING reserved word. PTF ignores any Set_Value or Verify action where the Value field is blank.

Example

In the following example, in the first step #NOTHING selects a blank value in the Calculation Reason field and then, in the next step, it verifies that the field is blank:

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 175

System Variables

System variables are populated by PTF at runtime. The following table lists PTF system variables.

Variable	*Description*
%case%	Current test case name.
%component%	Current component name.
%env.appserver%	Application server name.
%env.database%	Database name and database type. For example: T852U11 /DB2
%eo.dbname%	Execution option database name.
%eo.dbserver%	Execution option database server name.
%eo.dbuser%	Execution option database user name.
%eo.name%	Current execution option name.
%eo.platform%	Execution option platform.
%eo.pserver%	Execution option process server.
%eo.pshome%	Execution option PS_HOME path.
%env.toolsrel%	The PeopleTools version.
%eo.verbose%	Execution option verbose flag.

Variable	*Description*
%frame%	The current frame name.
%log.id%	Current log number.
%page%	Current page name.
%pid%	The last process instance number.
%test%	Current test name.

Functions

This section lists the PTF functions.

Add

Syntax

Add(*number1*|*number2*[|*number3*]...)

Description

Use the Add function to add a series of numbers. Decimals and negative numbers are supported.

Parameters

Parameter	*Description*
number1	Number to be added.
number2	Number to be added.
number3…	(Optional) A series of additional numbers to be added.

Returns

Sum of the numbers in the parameters.

Example

The following table presents examples of using Add.

Expression	*Result*
Add(10| -12| -3| 4| 6)	5
Add(10.43| 10.55| -6.789| -178)	-163.809

Concat

Syntax

Concat(*string1*|*string2*[|*string3*…])

Description

Concatenates the strings in the parameters.

Parameters

Parameter	*Description*
string1	Beginning string for concatenation
string2	A string to be concatenated to string1.
string3…	Additional strings to be concatenated.

Returns

Returns a string resulting from concatenating the strings in the parameters.

Example

The following table presents examples of using the concat function:

Expression	*Result*
concat(hello |and |welcome |to |PTF)	hello and welcome to PTF

Date

Syntax

Date()

Description

Returns the current date, using the date format specified in the current execution option.

Returns

The current date.

Example

The following table presents example of using the Date function:

Expression	*Result*
Date()	07/04/2011

Day

Syntax

Day*(date_value)*

Description

Returns the day portion of the date value provided as a parameter.

Parameters

Parameter	*Description*
date_value	A date value, such as the value returned by the Date() function.

Returns

Returns the day portion of the date value provided as a parameter.

Example

The following table presents examples of using the Day function:

Expression	*Result*
Day(February 13, 2012)	13
Day(Date())	13

Divide

Syntax

Divide(*number1*|*number2*[|*dec=dec_places*])

Description

Use the Divide function to perform division. Decimals and negative numbers are supported. Optionally specify the number of decimal places.

Parameters

Parameter	*Description*
number1	The dividend.
number2	The quotient.
dec=dec_places	(Optional) The number of decimal places. The maximum is 10. The default is zero. Note that "dec=" must be included in the parameter.

Returns

The result of dividing number1 by number2 rounded to dec_places decimals.

Example

The following table presents examples of using the Divide function:

Expression	*Result*
Divide(75| 13.5)	6

Expression	Result
Divide(-75| 13.5| dec=5)	-5.55556

GetField

Syntax

GetField*(string,segment,delimiter)*

Description

GetField returns the substring from a specified segment of a character-delimited text string.

Parameters

Parameter	Description
string	A character-delimited text string.
segment	An integer specifying which segment of the string will be returned, counting left to right. Specify a negative integer to count right to left.
delimiter	The character that delimits each segment in the string.

Returns

Returns the substring between the delimiters in the specified segment of the string.

Example

The following table presents examples of using GetField.

Expression	Result
GetField(a/b/c| 1| /)	a
GetField(a/b/c| 2| /)	b
GetField(a/b/c| 5| /)	blank
GetField(a/b/c| -1|/)	c

Hour

Syntax

Hour*(time_value)*

Description

Returns the hour portion of the time value provided as a parameter.

Parameters

Parameter	*Description*
time_value	A time value, such as the value returned by the Time() function.

Returns

Returns the hour portion of the time value provided as a parameter.

Example

The following table presents examples of using the Hour function:

Expression	*Result*
Hour(13:07:25)	13
Hour(Time())	13

InStr

Syntax

InStr*(within_string|string)*

Description

Locates a substring within a string of text and returns the starting position of the substring as an integer..

Parameters

Parameter	*Description*
substring	The text you are searching for. The string parameter is not case sensitive.
within_string	The text string you are searching within.

Returns

Returns an integer indicating the starting position of substring in within_string.

InStr returns 0 if substring does not appear in within_string. It returns 1 if substring is empty.

Example

The following table presents examples of using the InStr function.

Expression	*Result*
instr(ABCDEFG|c)	3
instr(ABCDEFG|C)	3
instr(ABCDEFG|CDE)	3
instr(ABCDEFG|zz)	0
instr(ABCDEFG|)	1
instr(ABCDEFG|A)	1

LCase

Syntax

LCase*(string)*

Description

Converts a string to lowercase.

Parameters

Parameter	*Description*
string	The string to be converted.

Returns

Returns a string resulting from converting string to lowercase.

Example

The following table presents example of using the LCase function.

Expression	*Result*
lcase(Hello World 1234)	hello world 1234

Left

Syntax

Left*(string|length)*

Description

Extracts a substring of a specified number of characters from the left side of a string.

Parameters

Parameter	*Description*
string	A string from which to extract a substring.
length	A number specifying the number of characters in the substring.

Returns

Returns a substring length characters long from the left side of a string.

Example

The following table presents example of using Left function.

Expression	*Result*
left(Hello World|5)	Hello

Len

Syntax

Len*(string)*

Description

Returns the length of string as an integer.

Parameters

Parameter	*Description*
string	A text string.

Returns

Returns an integer indicating the length of string.

Example

The following table presents example of using Len function.

Expression	*Result*
len(Hello World)	12

MakeDate

Syntax

MakeDate*(hour_value | minute_value | second_value | rollover_boolean)*

Description

This function returns a date value based on the year, month, and day values passed to the function as parameters.

Parameters

Parameter	*Description*
year_value	A number representing the year, such as the value returned by the Year() function.
month_value	A number representing the month, such as the value returned by the Month() function.
day_value	A number representing the day, such as the value returned by the Day() function.

Returns

Returns a date value.

Example

The following table presents examples of using the MakeDate function, assuming system date is February 13, 2012:

Expression	*Result*
MakeTime(Add(Hour(Time())|12)|Add(Minute(Time())|-30)|Second(Time()))	January 14, 2014
MakeDate(Add(Year(Date())|1)|Add(Month(Date())|-1)|Add(Day(Date())|-1)	January 12, 2013

MakeTime

Syntax

MakeTime*(hour_value | minute_value | second_value | rollover_boolean)*

Description

This function returns a time value based on the hour, minute, and second values passed to the function as parameters.

Parameters

Parameter	*Description*
hour_value	A number representing the hour, such as the value returned by the Hour() function.
minute_value	A number representing the minute, such as the value returned by the Minute() function.
second_value	A number representing the second, such as the value returned by the Second() function.
rollover_boolean	(Optional) Rolls over the hour to 0 when hour_value reaches 24. True - Returns (hour_value – 24) when hour_value is greater than 24. False - Returns hour_value even when hour_value is greater than 24. The default value is True.

Returns

Returns a time value.

Example

The following table presents examples of using the Day function:

Expression	*Result*
MakeTime(Add(Hour(Time())|12)|Add(Minute(Time())|-30)|Second(Time()))	7:00:00 AM
MakeTime(23|Add(60|30)|0|False)	24:30:00
MakeTime(23|Add(60|30)|0|True)	00:30:00
MakeTime(23|Add(60|-30)|0)	23:30:00

Minute

Syntax

Minute*(time_value)*

Description

Returns the minute portion of the time value provided as a parameter.

Parameters

Parameter	*Description*
time_value	A time value, such as the value returned by the Time() function.

Returns

Returns the minute portion of the time value provided as a parameter.

Example

The following table presents examples of using the Minute function:

Expression	*Result*
Minute(13:07:25)	7
Minute(Time())	7

Month

Syntax

Month*(date_value)*

Description

Returns the month portion of the date value provided as a parameter.

Parameters

Parameter	*Description*
date_value	A date value, such as the value returned by the Date() function.

Returns

Returns the month portion of the date value provided as a parameter.

Example

The following table presents examples of using the Month function:

Expression	*Result*
Month(February 13, 2012)	2
Month(Date())	2

Multiply

Syntax

Multiply(*number1*|*number2*[|*dec=dec_places*])

Description

Use the Multiply function to perform multiplication. Decimals and negative numbers are supported. Optionally specify the number of decimal places.

Parameters

Parameter	*Description*
number1	First factor.
number2	Second factor.
dec_places	(Optional) The number of decimal places. The maximum is 10. The default is zero. Note that "dec=" must be included in the parameter.

Returns

The result of multiplying number1 by number2 rounded to dec_places decimals.

Example

The following table presents examples of using Multiply function.

Expression	*Result*
Multiply(10.3| 13.45)	139
Multiply(10.3| -13.45|dec=3)	-138.535

Now

Syntax

Now()

Description

Returns the current datetime, using the date format specified in the current execution option.

Returns

The current datetime.

Example

The following table presents example of using Now function.

Expression	*Result*
Now()	07/04/2011 12:20 PM

Replace

Syntax

Replace*(source\|find\|replace)*

Description

Use the Replace function to replace every occurrence of a substring found in a string with a new substring.

Parameters

Parameter	*Description*
source	A string in which you want to replace substrings.
find	A string equal to the substring of source you want to replace.
replace	A string with which to replace occurrences of find in source.

Returns

Returns a string resulting from replacing every occurrence of find found in source with replace.

Example

The following table presents example of using Replace function.

Expression	*Result*
replace(original text|i|77)	Or77g77nal text

Right

Syntax

Right*(string|length)*

Description

Use the Right function to extract a substring of a specified number of characters from the right side of a string.

Parameters

Parameter	*Description*
string	A string from which to extract a substring.
length	A number specifying the number of characters in the substring.

Returns

Returns a substring length characters long from the right side of a string.

Example

The following table presents example of using Replace function.

Expression	*Result*
right(Hello World|5)	World

Round

Syntax

Round(*number*|[|*dec=dec_places*])

Description

Use the Round function to round a number. Decimals and negative numbers are supported. Optionally specify the number of decimal places.

Parameters

Parameter	*Description*
number1	First factor.
number2	Second factor.
dec_places	(Optional) The number of decimal places. The maximum is 10. The default is zero. Note that "dec=" must be included in the parameter.

Returns

The result of rounding number1 to dec_places decimal places.

Example

The following table presents example of using Round function.

Expression	*Result*
Round(-130.456)	-130
Round(-130.456|dec=2)	-130.46
Round(-130.455|dec=2)	-130.45

Second

Syntax

Second*(time_value)*

Description

Returns the second portion of the time value provided as a parameter.

Parameters

Parameter	*Description*
time_value	A time value, such as the value returned by the Time() function.

Returns

Returns the second portion of the time value provided as a parameter.

Example

The following table presents examples of using the Second function:

Expression	*Result*
Second(13:07:25)	25
Second(Time())	25

SubStr

Syntax

SubStr(*source_str*|*start_pos*[|*length*])

Description

Extracts a substring of a specified number of characters beginning at a specified location in a source string.

If length is not specified, SubStr returns the substring starting at the position specified in start_pos and continuing to the end of the string.

Parameters

Parameter	***Description***
source_str	A string from which to extract a substring.
start_pos	A number representing the character position in source_str where the substring starts, starting at 1.
length	A number specifying the number of characters in the substring.

Returns

Returns a string equal to a substring length characters long beginning at character start_pos of source_str.

Example

The following table presents examples of using SubStr function.

Expression	***Result***
substr(12345678|2|3)	234
substr(12345678|2)	2345678

Subtract

Syntax

Subtract(*number1*|*number2*[|*number3*]...])

Description

Use the Subtract function to subtract a series of numbers. Numbers can be decimal and negative.

Parameters

Parameter	*Description*
number1	Initial number.
number2	Number to be subracted.
number3…	(Optional) A series of additional numbers to be subtracted.

Returns

The result of subtracting number2, number3, etc., from number1.

Example

The following table presents examples of using Subtract function.

Expression	*Result*
Subtract(10|2|3)	5
Subtract(10|-2|-3)	15

Sum

Syntax

Sum,Index, *value,delimiter*)

Description

Sum works with the HTMLTable indexes.

Parameters

Parameter	*Description*
Index	The HTMLTable index string, such as 2/5/4. An index string is the return value of CellGetIndex. See Chapter 10, "Test Language Reference," CellGetIndex, page 136.
Value	The value that you want to add or subtract. The default action is addition.
Section	The section of the index that will be modified.
Delimiter	The character that delimits each section in the text value.

Example

The following table presents examples of using Sum.

Expression	*Result*
Sum(2/5/4, 2, 1, /)	4/5/4 2 is added to the first section of the string.
Sum(2/5/4, -1, 3, /)	2/5/3
Sum(&index, -4, 3, /)	4 is subtracted from the third section of the string in the variable &index.

Time

Syntax

Time()

Description

Returns the current time, using the date format specified in the current execution option.

Parameters

None

Returns

Returns a string with the current time.

Example

The following table presents example of using the Time function with the Regional Options set to 24 hour format and 12 hour format, respectively:

Expression	*Result*
Time()	13:07:25
Time()	1:07:25 PM

Trim

Syntax

Trim*(string)*

Description

Returns a string with all leading and trailing spaces removed.

Parameters

Parameter	*Description*
string	A text string.

Returns

Returns a string with all leading and trailing spaces removed.

Example

The following table presents example of using trim function.

Expression	*Result*
trim(Hello World)	Hello World

UCase

Syntax

UCase*(string)*

Description

Converts a string to uppercase.

Parameters

Parameter	*Description*
string	The string to be converted.

Returns

Returns a string resulting from converting string to uppercase.

Example

The following table presents example of using UCase function.

Expression	*Result*
ucase(Hello World 1234)	HELLO WORLD 1234

Year

Syntax

Year*(date_value)*

Description

Returns the year portion of the date value provided as a parameter.

Parameters

Parameter	*Description*
date_value	A date value, such as the value returned by the Date() function.

Returns

Returns the year portion of the date value provided as a parameter.

Example

The following table presents examples of using the Year function:

Expression	*Result*
Year(February 13, 2012)	2012
Year(Date())	2012

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 173.
#DIS#	Checks whether an object is display-only. See Chapter 10, "Test Language Reference," #DIS#, page 174.
#DTTM	Enters the current date and time into the application. See Chapter 10, "Test Language Reference," #DTTM, page 174.
#EXIST# and **#NOTEXIST#**	Check whether a field exists or does not exist on the page. #Exist can update the field if a value is passed following the closing # sign. See Chapter 10, "Test Language Reference," #EXIST#, page 174 and Chapter 10, "Test Language Reference," #NOTEXIST#, page 178.
#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 175 and Chapter 10, "Test Language Reference," #WARN#, page 181.
#IGNORE	Place the #IGNORE reserved word in the Value field of a Test.Exec step to skip the call to the child test. See Chapter 10, "Test Language Reference," #IGNORE, page 176.

#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 176 and Chapter 10, "Test Language Reference," #LIKEW#, page 177.
#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 178.
#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 179.
#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 179.
#TODAY	Enters the current date into the application. See Chapter 10, "Test Language Reference," #TODAY, page 180.

Index