PeopleTools 8.53: PeopleSoft Test Framework
Trademark Notice

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

License Restrictions Warranty/Consequential Damages Disclaimer

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

Warranty Disclaimer

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

Restricted Rights Notice

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

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

Hazardous Applications Notice

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

Third Party Content, Products, and Services Disclaimer

This software or hardware and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.
Contents

Preface .......................................................................................................................................................... xi

Understanding the PeopleSoft Online Help and PeopleBooks .............................................................. xi

PeopleSoft Hosted Documentation .......................................................................................................... xi

Locally Installed Help ................................................................................................................................ xi

Downloadable PeopleBook PDF Files ....................................................................................................... xi

Common Help Documentation .................................................................................................................. xi

Typographical Conventions .................................................................................................................... xii

ISO Country and Currency Codes .............................................................................................................. xiii

Region and Industry Identifiers ................................................................................................................ xiii

Access to Oracle Support .......................................................................................................................... xiv

Documentation Accessibility ..................................................................................................................... xiv

Using and Managing the PeopleSoft Online Help .................................................................................... xiv

PeopleTools Related Links ....................................................................................................................... xv

Contact Us ................................................................................................................................................ xv

Follow Us ................................................................................................................................................ xiv

Chapter 1: Understanding PeopleSoft Test Framework ............................................................................. 17

Understanding PeopleSoft Test Framework ............................................................................................ 17

Understanding PeopleSoft Test Framework ............................................................................................ 17

Terminology ............................................................................................................................................. 18

Chapter 2: Installing and Configuring PTF ............................................................................................... 21

Installing and Configuring PTF ................................................................................................................ 21

Understanding the PTF Development Environment .............................................................................. 21

Configuring an Environment for PTF ...................................................................................................... 22

Verifying Integration Broker Setup ....................................................................................................... 22

Setting Up Security .................................................................................................................................. 24

Configuring the Web Profile ..................................................................................................................... 26

Defining PTF Configuration Options ...................................................................................................... 27

Defining Configuration Options .............................................................................................................. 27

Installing a PTF Client ............................................................................................................................... 29

Verifying Requirements ........................................................................................................................... 30

Configuring the Browser Security Settings ............................................................................................ 31

Configuring Browser Settings with Windows 7 ..................................................................................... 32

Installing PTF Client Software ................................................................................................................ 33

Creating a Connection to a PTF Environment .......................................................................................... 34

Selecting a PTF Environment .................................................................................................................... 38

Configuring Local Options ....................................................................................................................... 39

Configuring Execution Options .............................................................................................................. 41

Configuring Execution Options in PTF Client .......................................................................................... 41

Configuring Execution Options in PeopleSoft Internet Architecture .................................................. 50

Configuring Execution Options ................................................................................................................ 50

Configuring Advanced Options ................................................................................................................ 52

Configuring PeopleTools Options ........................................................................................................... 53

Configuring Export Log Options ............................................................................................................. 55

Chapter 3: Using PeopleSoft Test Framework ........................................................................................ 57

Using PeopleSoft Test Framework .......................................................................................................... 57

Using PTF Explorer .................................................................................................................................. 57

Using the Test Editor ................................................................................................................................. 60
## Contents

<table>
<thead>
<tr>
<th>Chapter and Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Editor Menus</td>
<td>61</td>
</tr>
<tr>
<td>Test Editor Toolbar</td>
<td>64</td>
</tr>
<tr>
<td>Test Window</td>
<td>65</td>
</tr>
<tr>
<td>Test Window Fields</td>
<td>65</td>
</tr>
<tr>
<td>Test Window Toolbar</td>
<td>67</td>
</tr>
<tr>
<td>Test Step Fields</td>
<td>67</td>
</tr>
<tr>
<td>Using the PTF Test Recorder</td>
<td>68</td>
</tr>
<tr>
<td>Test Recorder Toolbar</td>
<td>69</td>
</tr>
<tr>
<td>Recording Action Tools</td>
<td>69</td>
</tr>
<tr>
<td>Step Modification Tools</td>
<td>71</td>
</tr>
<tr>
<td>Recorder Utility Tools</td>
<td>72</td>
</tr>
<tr>
<td>Using the Log Viewer</td>
<td>73</td>
</tr>
<tr>
<td>Chapter 4: Creating Tests and Test Cases</td>
<td>77</td>
</tr>
<tr>
<td>Creating Tests and Test Cases</td>
<td>77</td>
</tr>
<tr>
<td>Creating Tests</td>
<td>77</td>
</tr>
<tr>
<td>Creating a New Folder</td>
<td>77</td>
</tr>
<tr>
<td>Creating a New Test</td>
<td>78</td>
</tr>
<tr>
<td>Naming Tests</td>
<td>78</td>
</tr>
<tr>
<td>Copying a Test</td>
<td>78</td>
</tr>
<tr>
<td>Renaming a Test</td>
<td>79</td>
</tr>
<tr>
<td>Recording Tests</td>
<td>79</td>
</tr>
<tr>
<td>Test Action Tools</td>
<td>80</td>
</tr>
<tr>
<td>Recording a Test</td>
<td>81</td>
</tr>
<tr>
<td>Creating Test Cases</td>
<td>82</td>
</tr>
<tr>
<td>Creating a New Test Case</td>
<td>82</td>
</tr>
<tr>
<td>Creating a Test Case With Values</td>
<td>83</td>
</tr>
<tr>
<td>Exporting and Importing Test Cases</td>
<td>84</td>
</tr>
<tr>
<td>Exporting Test Cases</td>
<td>84</td>
</tr>
<tr>
<td>Importing Test Cases</td>
<td>85</td>
</tr>
<tr>
<td>Executing Tests</td>
<td>86</td>
</tr>
<tr>
<td>Executing a Test</td>
<td>86</td>
</tr>
<tr>
<td>Executing a Test Case</td>
<td>87</td>
</tr>
<tr>
<td>Executing a Test from the Command Line</td>
<td>87</td>
</tr>
<tr>
<td>Reviewing Test Logs</td>
<td>89</td>
</tr>
<tr>
<td>Exporting Test Logs to XML</td>
<td>92</td>
</tr>
<tr>
<td>Organizing Tests In PTF Explorer</td>
<td>94</td>
</tr>
<tr>
<td>Cutting and Pasting Multiple Tests or Folders</td>
<td>94</td>
</tr>
<tr>
<td>Deleting Multiple Tests or Folders</td>
<td>94</td>
</tr>
<tr>
<td>Expanding or Collapsing Tests and Folders</td>
<td>94</td>
</tr>
<tr>
<td>Chapter 5: Developing and Debugging Tests</td>
<td>97</td>
</tr>
<tr>
<td>Developing and Debugging Tests</td>
<td>97</td>
</tr>
<tr>
<td>Using the Message Tool</td>
<td>97</td>
</tr>
<tr>
<td>Using Reserved Words</td>
<td>99</td>
</tr>
<tr>
<td>Using Variables</td>
<td>101</td>
</tr>
<tr>
<td>Using Text Strings as Parameters in Functions</td>
<td>103</td>
</tr>
<tr>
<td>Using Persistent Variables</td>
<td>104</td>
</tr>
<tr>
<td>Setting Variable Option in the Test</td>
<td>104</td>
</tr>
<tr>
<td>Setting Persistent Variable Options</td>
<td>105</td>
</tr>
<tr>
<td>Managing Persistent Variables</td>
<td>105</td>
</tr>
<tr>
<td>Example of a Test that uses Persistent Variables</td>
<td>106</td>
</tr>
<tr>
<td>Using Conditional Logic</td>
<td>106</td>
</tr>
</tbody>
</table>
Handling Application Messages................................................................. 107
Interpreting Logs....................................................................................... 110
Incorporating Scroll Handling................................................................. 111
Calling Tests............................................................................................... 116
   Understanding Calling Tests................................................................. 116
   Using Library Tests............................................................................... 116
   Using Parameters with Library Tests.................................................. 117
   Using Shell Tests.................................................................................. 119
Sharing Test Assets.................................................................................. 119

Chapter 6: Administering PTF................................................................... 121
Administering PTF.................................................................................... 121
Managing PTF Logs.................................................................................. 121
   Understanding Log Manager............................................................... 121
   Using Log Manager Toolbar............................................................... 122
   Using Log Manager Fields................................................................. 123
   Using the Selection Pane................................................................. 123
   Using the Trace Pane......................................................................... 124
Migrating PTF Tests.................................................................................. 124

Chapter 7: Identifying Change Impacts....................................................... 127
Identifying Change Impacts....................................................................... 127
Defining Analysis Rules............................................................................ 128
Creating Test Maintenance Reports......................................................... 130
   Step 1 of 4: Manual Tasks.................................................................. 130
   Step 2 of 4: Analyze Compare Data.................................................. 133
   Step 3 of 4: Select an Analyzed Project.............................................. 134
   Step 4 of 4: Generate Report............................................................. 134
Interpreting Test Maintenance Reports.................................................... 137
Understanding Test Coverage Reports..................................................... 139
Creating Test Coverage Reports............................................................... 140
   Step 1 of 2: Select a Project.............................................................. 140
   Step 2 of 2: Generate Report............................................................ 142
Using Usage Monitor Data with PTF......................................................... 143
Configuring Usage Monitor................................................................. 143
Generating Usage Monitor Data............................................................. 143
Administering Usage Monitor for PTF..................................................... 144
Interpreting Test Coverage Reports......................................................... 144
Running Test Details Reports................................................................. 145
Creating a Test Compare Report.............................................................. 147
Querying PTF Report Tables..................................................................... 150

Chapter 8: Incorporating Best Practices................................................... 153
Incorporating Best Practices..................................................................... 153
Incorporating PTF Best Practices............................................................. 153
   Keep your Desktop Simple.............................................................. 153
   Adopt Naming Conventions............................................................. 153
   Record First....................................................................................... 154
   Document Tests............................................................................... 155
   Clean Up Tests............................................................................... 155
   Use Execution Options..................................................................... 156
   Use Page Prompting....................................................................... 156
Contents

- Use the Process Step Type................................................................. 157
- Make Tests Dynamic........................................................................ 157
- Reduce Duplication........................................................................ 158

Chapter 9: Using the PTF Test Language........................................ 159

- Using the PTF Test Language......................................................... 159
- Understanding the PTF Test Structure.......................................... 159
- PTF Test Language........................................................................ 160
  - Validation.................................................................................... 161
  - Parameters................................................................................ 161
  - Variables.................................................................................. 162
  - Reserved Words....................................................................... 162
  - System Variables..................................................................... 163
  - Check Syntax........................................................................... 163
  - Context Sensitive Help within Grid for Function Parameter Details. 166

Chapter 10: Test Language Reference............................................. 167

- Test Language Reference............................................................. 167
- Step Types................................................................................... 167
- Browser........................................................................................ 167
  - Close....................................................................................... 167
  - FrameSet............................................................................... 167
  - Set_URL............................................................................... 168
  - Start....................................................................................... 168
  - Start_Login........................................................................... 168
  - WaitForNew......................................................................... 168
- Button.......................................................................................... 168
  - Click...................................................................................... 169
  - Exists..................................................................................... 169
  - Get_Property......................................................................... 169
- Chart............................................................................................. 169
  - ChartClick............................................................................ 169
  - GetChartType....................................................................... 169
  - GetText.................................................................................. 170
  - SectionCount.......................................................................... 170
- CheckBox...................................................................................... 171
  - Exists..................................................................................... 171
  - Get_Label.............................................................................. 171
  - Get_Property......................................................................... 171
  - Set_Value............................................................................... 172
  - Verify...................................................................................... 172
- ComboBox.................................................................................... 172
  - Exists..................................................................................... 172
  - Get_Property......................................................................... 172
  - GetLabel................................................................................ 172
  - Set_Value............................................................................... 172
  - Verify...................................................................................... 172
- Command...................................................................................... 173
  - Exec....................................................................................... 174
- Conditional................................................................................... 175
  - If_Then.................................................................................... 175
  - Else......................................................................................... 175
  - End_If..................................................................................... 176
Contents

DataMover................................................................................................................................................176
Exec......................................................................................................................................................176
Div.......................................................................................................................................................177
Click....................................................................................................................................................177
Execution................................................................................................................................................177
Set_Options...........................................................................................................................................177
Skip_PageSave....................................................................................................................................177
Skip_RunRequest.................................................................................................................................178
Skip_Login..........................................................................................................................................178
StopOnError.......................................................................................................................................178
File....................................................................................................................................................178
Download...........................................................................................................................................178
Upload...............................................................................................................................................179
HTMLTable.........................................................................................................................................180
CellClick...........................................................................................................................................180
CellClickOnChkB................................................................................................................................180
CellClickOnImage................................................................................................................................181
CellClickOnLink.................................................................................................................................181
CellExists..........................................................................................................................................181
CellGetIndex.....................................................................................................................................182
CellGetValue.....................................................................................................................................183
ColCount...........................................................................................................................................183
RowCount............................................................................................................................................184
Image................................................................................................................................................184
Click..................................................................................................................................................184
Exists..............................................................................................................................................185
Get_Property......................................................................................................................................185
RightClick.........................................................................................................................................185
Link..................................................................................................................................................185
Click..................................................................................................................................................185
Exists..............................................................................................................................................185
Get_Property......................................................................................................................................186
SaveTargetAs.....................................................................................................................................186
Verify................................................................................................................................................186
Log....................................................................................................................................................186
Fail....................................................................................................................................................187
Message...........................................................................................................................................187
Pass...................................................................................................................................................187
SnapShot..........................................................................................................................................187
Warning.............................................................................................................................................187
LongText..........................................................................................................................................188
Exists..............................................................................................................................................188
Get_Property......................................................................................................................................188
GetLabel............................................................................................................................................189
SetValue............................................................................................................................................189
SetValue_InModal..............................................................................................................................189
Verify................................................................................................................................................189
Loop..................................................................................................................................................189
Do......................................................................................................................................................189
End_Loop..........................................................................................................................................190
Exit....................................................................................................................................................190
<table>
<thead>
<tr>
<th>Action</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>While</td>
<td>190</td>
</tr>
<tr>
<td>For</td>
<td>191</td>
</tr>
<tr>
<td>Next</td>
<td>191</td>
</tr>
<tr>
<td>MsgBox</td>
<td>191</td>
</tr>
<tr>
<td>Actions for MsgBox Step Type</td>
<td>192</td>
</tr>
<tr>
<td>MultiSelect</td>
<td>193</td>
</tr>
<tr>
<td>Exists</td>
<td>193</td>
</tr>
<tr>
<td>Get_Property</td>
<td>193</td>
</tr>
<tr>
<td>GetLabel</td>
<td>194</td>
</tr>
<tr>
<td>Set_Value</td>
<td>194</td>
</tr>
<tr>
<td>Verify</td>
<td>194</td>
</tr>
<tr>
<td>Page</td>
<td>194</td>
</tr>
<tr>
<td>Expand</td>
<td>195</td>
</tr>
<tr>
<td>Go_To</td>
<td>195</td>
</tr>
<tr>
<td>Prompt</td>
<td>195</td>
</tr>
<tr>
<td>PromptOK</td>
<td>195</td>
</tr>
<tr>
<td>Save</td>
<td>196</td>
</tr>
<tr>
<td>SecPage_Close</td>
<td>196</td>
</tr>
<tr>
<td>SecPage_Open</td>
<td>196</td>
</tr>
<tr>
<td>Process</td>
<td>197</td>
</tr>
<tr>
<td>Run</td>
<td>197</td>
</tr>
<tr>
<td>Run_Def</td>
<td>198</td>
</tr>
<tr>
<td>Pwd</td>
<td>199</td>
</tr>
<tr>
<td>Exists</td>
<td>199</td>
</tr>
<tr>
<td>GetLabel</td>
<td>199</td>
</tr>
<tr>
<td>Set_Value</td>
<td>199</td>
</tr>
<tr>
<td>Query</td>
<td>199</td>
</tr>
<tr>
<td>Exec</td>
<td>199</td>
</tr>
<tr>
<td>Exec_Private</td>
<td>200</td>
</tr>
<tr>
<td>Radio</td>
<td>201</td>
</tr>
<tr>
<td>Exists</td>
<td>201</td>
</tr>
<tr>
<td>Get_Property</td>
<td>202</td>
</tr>
<tr>
<td>GetLabel</td>
<td>202</td>
</tr>
<tr>
<td>Set_Value</td>
<td>202</td>
</tr>
<tr>
<td>Verify</td>
<td>202</td>
</tr>
<tr>
<td>RichText</td>
<td>202</td>
</tr>
<tr>
<td>GetLabel</td>
<td>203</td>
</tr>
<tr>
<td>Set_Value</td>
<td>203</td>
</tr>
<tr>
<td>Scroll</td>
<td>203</td>
</tr>
<tr>
<td>Action</td>
<td>203</td>
</tr>
<tr>
<td>Definition</td>
<td>204</td>
</tr>
<tr>
<td>ModalGrid_Close</td>
<td>205</td>
</tr>
<tr>
<td>ModalGrid_Open</td>
<td>206</td>
</tr>
<tr>
<td>Key_Set</td>
<td>206</td>
</tr>
<tr>
<td>Reset</td>
<td>206</td>
</tr>
<tr>
<td>RowCount</td>
<td>206</td>
</tr>
<tr>
<td>Span</td>
<td>207</td>
</tr>
<tr>
<td>Click</td>
<td>207</td>
</tr>
<tr>
<td>Exists</td>
<td>207</td>
</tr>
<tr>
<td>Get_Property</td>
<td>207</td>
</tr>
<tr>
<td>GetLabel</td>
<td>208</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
</tr>
<tr>
<td>MouseOver</td>
<td>208</td>
</tr>
<tr>
<td>MouseOverClose</td>
<td>208</td>
</tr>
<tr>
<td>Verify</td>
<td>209</td>
</tr>
<tr>
<td>Test</td>
<td>209</td>
</tr>
<tr>
<td>Exec</td>
<td>209</td>
</tr>
<tr>
<td>Text</td>
<td>210</td>
</tr>
<tr>
<td>Exists</td>
<td>210</td>
</tr>
<tr>
<td>Get_Property</td>
<td>210</td>
</tr>
<tr>
<td>GetLabel</td>
<td>210</td>
</tr>
<tr>
<td>Set_Value</td>
<td>211</td>
</tr>
<tr>
<td>Verify</td>
<td>211</td>
</tr>
<tr>
<td>Variable</td>
<td>211</td>
</tr>
<tr>
<td>Set_Value</td>
<td>211</td>
</tr>
<tr>
<td>Wait</td>
<td>212</td>
</tr>
<tr>
<td>For_Seconds</td>
<td>212</td>
</tr>
<tr>
<td>For_Value</td>
<td>212</td>
</tr>
<tr>
<td>Common Actions</td>
<td>212</td>
</tr>
<tr>
<td>Click</td>
<td>213</td>
</tr>
<tr>
<td>Exists</td>
<td>213</td>
</tr>
<tr>
<td>Get_Property</td>
<td>215</td>
</tr>
<tr>
<td>GetLabel</td>
<td>216</td>
</tr>
<tr>
<td>Set_Value</td>
<td>217</td>
</tr>
<tr>
<td>Verify</td>
<td>218</td>
</tr>
<tr>
<td>Reserved Words</td>
<td>219</td>
</tr>
<tr>
<td>#CHECK#</td>
<td>219</td>
</tr>
<tr>
<td>#DIS#</td>
<td>220</td>
</tr>
<tr>
<td>#DTTM#</td>
<td>220</td>
</tr>
<tr>
<td>#EXIST#</td>
<td>220</td>
</tr>
<tr>
<td>#FAIL#</td>
<td>221</td>
</tr>
<tr>
<td>#IGNORE</td>
<td>221</td>
</tr>
<tr>
<td>#LIKEF#</td>
<td>222</td>
</tr>
<tr>
<td>#LIKEW#</td>
<td>222</td>
</tr>
<tr>
<td>#LIST#</td>
<td>223</td>
</tr>
<tr>
<td>#NOTEXIST#</td>
<td>224</td>
</tr>
<tr>
<td>#NOTHING</td>
<td>224</td>
</tr>
<tr>
<td>#PREFIX#</td>
<td>225</td>
</tr>
<tr>
<td>#TODAY</td>
<td>225</td>
</tr>
<tr>
<td>#WARN#</td>
<td>226</td>
</tr>
<tr>
<td>System Variables</td>
<td>226</td>
</tr>
<tr>
<td>Functions</td>
<td>227</td>
</tr>
<tr>
<td>Add</td>
<td>228</td>
</tr>
<tr>
<td>Concat</td>
<td>228</td>
</tr>
<tr>
<td>Date</td>
<td>229</td>
</tr>
<tr>
<td>Day</td>
<td>229</td>
</tr>
<tr>
<td>Divide</td>
<td>230</td>
</tr>
<tr>
<td>GetField</td>
<td>231</td>
</tr>
<tr>
<td>Hour</td>
<td>231</td>
</tr>
<tr>
<td>InStr</td>
<td>232</td>
</tr>
<tr>
<td>LCase</td>
<td>233</td>
</tr>
<tr>
<td>Left</td>
<td>233</td>
</tr>
<tr>
<td>Len</td>
<td>234</td>
</tr>
</tbody>
</table>
## Appendix A: Reserved Words Quick Reference

Reserved Words Quick Reference.. 247
Reserved Words................................................. 247
Preface

Understanding the PeopleSoft Online Help and PeopleBooks

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

PeopleSoft Hosted Documentation

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

Locally Installed Help

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

In addition, if you locally install the PeopleSoft Online Help, you can use any search engine for full-text searching. Your installation documentation includes instructions about how to set up Oracle Secure Enterprise Search for full-text searching.

See PeopleTools 8.53 Installation for your database platform, "Installing PeopleSoft Online Help.” If you do not use Secure Enterprise Search, see the documentation for your chosen search engine.

Note: Before users can access the search engine on a locally installed help website, you must enable the Search portlet and link. Click the Help link on any page in the PeopleSoft Online Help for instructions.

Downloadable PeopleBook PDF Files

You can access downloadable PDF versions of the help content in the traditional PeopleBook format. The content in the PeopleBook PDFs is the same as the content in the PeopleSoft Online Help, but it has a different structure and it does not include the interactive navigation features that are available in the online help.

Common Help Documentation

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

• Application Fundamentals
• Using PeopleSoft Applications

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

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

**Typographical Conventions**

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

<table>
<thead>
<tr>
<th><strong>Typographical Convention</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bold</td>
<td>Highlights PeopleCode function names, business function names, event names, system function names, method names, language constructs, and PeopleCode reserved words that must be included literally in the function call.</td>
</tr>
<tr>
<td>Italic</td>
<td>Highlights field values, emphasis, and PeopleSoft or other book-length publication titles. In PeopleCode syntax, italic items are placeholders for arguments that your program must supply. Italics also highlight references to words or letters, as in the following example: Enter the letter <strong>O</strong>.</td>
</tr>
<tr>
<td>Key+Key</td>
<td>Indicates a key combination action. For example, a plus sign (+) between keys means that you must hold down the first key while you press the second key. For Alt+W, hold down the Alt key while you press the W key.</td>
</tr>
<tr>
<td>Monospace font</td>
<td>Highlights a PeopleCode program or other code example.</td>
</tr>
<tr>
<td>. . . (ellipses)</td>
<td>Indicate that the preceding item or series can be repeated any number of times in PeopleCode syntax.</td>
</tr>
<tr>
<td>{ } (curly braces)</td>
<td>Indicate a choice between two options in PeopleCode syntax. Options are separated by a pipe (</td>
</tr>
<tr>
<td>[ ] (square brackets)</td>
<td>Indicate optional items in PeopleCode syntax.</td>
</tr>
</tbody>
</table>
Typographical Convention | Description
--- | ---
& (ampersand) | When placed before a parameter in PeopleCode syntax, an ampersand indicates that the parameter is an already instantiated object. Ampersands also precede all PeopleCode variables.
⇒ | This continuation character has been inserted at the end of a line of code that has been wrapped at the page margin. The code should be viewed or entered as a single, continuous line of code without the continuation character.

ISO Country and Currency Codes

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

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

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

Region and Industry Identifiers

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

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

Region Identifiers

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

- Asia Pacific
- Europe
- Latin America
- North America
Industry Identifiers

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

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

Access to Oracle Support

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

Documentation Accessibility

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

Using and Managing the PeopleSoft Online Help

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

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

PeopleTools Related Links

Oracle's PeopleSoft PeopleTools 8.53 Documentation Home Page [ID 1494462.1]

PeopleSoft Information Portal on Oracle.com

My Oracle Support

PeopleSoft Training from Oracle University

PeopleSoft Video Feature Overviews on YouTube
Contact Us

Send us your suggestions Please include release numbers for the PeopleTools and applications that you are using.

Follow Us

Get the latest PeopleSoft updates on Facebook.

Follow PeopleSoft on Twitter@PeopleSoft_Info.
Chapter 1

Understanding PeopleSoft Test Framework

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

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 Understanding Change Impacts.

- Features that help users manipulate data within the PeopleSoft rowset-oriented data structure.
  
  See Incorporating Scroll Handling.

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

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

You should be aware that PTF is not designed to:

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

### Terminology

This table defines some PTF terms:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset</td>
<td>See Test Asset.</td>
</tr>
<tr>
<td>Execution Options</td>
<td>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.</td>
</tr>
<tr>
<td>Explorer</td>
<td>See PTF Explorer.</td>
</tr>
<tr>
<td>Grid</td>
<td>See Scroll</td>
</tr>
<tr>
<td>Hook</td>
<td>Establish a connection between a PTF test and a PeopleSoft application browser.</td>
</tr>
<tr>
<td>Library</td>
<td>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.</td>
</tr>
<tr>
<td>Log</td>
<td>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.</td>
</tr>
<tr>
<td>Log Manager</td>
<td>A tool that enables PTF administrators to purge unneeded logs</td>
</tr>
<tr>
<td>Maintenance</td>
<td>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</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PTF Client</strong></td>
<td>An instance of the PTF executable program installed on an individual user’s machine.</td>
</tr>
<tr>
<td><strong>PTF Environment</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>PTF Explorer</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Recorder</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Screen Shot</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Scroll</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Scroll Area</strong></td>
<td>See Scroll.</td>
</tr>
<tr>
<td><strong>Step</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Test Asset</strong></td>
<td>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:</td>
</tr>
</tbody>
</table>

- 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

Installing and Configuring PTF

This topic 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:

Image: Diagram of the PTF development environment

This diagram illustrates the PTF development environment.

A PTF development environment consists of the following elements:

• A PTF client instance.
• 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:

**Image: Integration Broker Gateways page**

This example illustrates the fields and controls on the Integration Broker Gateways page. You can find definitions for the fields and controls later on this page.

<table>
<thead>
<tr>
<th>Gateway ID</th>
<th>LOCAL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Local Gateway</strong></td>
<td>[ ]</td>
</tr>
<tr>
<td><strong>Load Balancer</strong></td>
<td>[ ]</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://myserver.us.oracle.com:8010/PSIGW/PeopleSoftListeningConnect">http://myserver.us.oracle.com:8010/PSIGW/PeopleSoftListeningConnect</a></td>
</tr>
</tbody>
</table>

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:

**Image: Integration Gateway Ping message**

This example illustrates a successful Integration Gateway Ping message.

---

**PeopleSoft Integration Gateway**

PeopleSoft Listening Connector
Tools Version : 8.53
Status: ACTIVE

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

**Image: PeopleSoft Node Configuration page**

This example illustrates the fields and controls on the PeopleSoft Node Configuration page. You can find definitions for the fields and controls later on this page.

<table>
<thead>
<tr>
<th>App Server URL</th>
<th>User ID</th>
<th>Password</th>
<th>Tools Release Domain</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>(JPServer 9040)</td>
<td>JPSERVER</td>
<td>JPSERVER</td>
<td>JPSERVER</td>
<td>JPSERVER</td>
</tr>
</tbody>
</table>

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 Creating a Connection to a PTF Environment.

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

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

1. Navigate to Integration Broker, Integration Setup, Nodes.
2. Select the ANONYMOUS node.
3. Note the Default User ID.
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.
   

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

See PeopleTools: PeopleSoft Integration Broker Administration

## 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:

<table>
<thead>
<tr>
<th>Privilege</th>
<th>PTF User</th>
<th>PTF Editor</th>
<th>PTF Administrator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run Tests</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Create, Modify, and Delete Tests</td>
<td>No*</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Create, Modify, and Delete Test Cases</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Create or Modify Execution Options</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Privilege</td>
<td>PTF User</td>
<td>PTF Editor</td>
<td>PTF Administrator</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------</td>
<td>------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Use Log Manager</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Define Configuration Options</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Create Test Maintenance Reports</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Create Test Coverage Reports</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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

Modify Test includes these actions:

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

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

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

**Image: Signon error message**

This example illustrates a 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.

For more information on entering roles for a user profile, see "Working With User Profiles (PeopleTools 8.53: Security Administration)"

**Related Links**
Defining PTF Configuration Options
Creating Test Maintenance Reports
Creating Test Coverage Reports

---

**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 check box.

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

5. Check the Show Connection & Sys Info check box.
If this option is not selected PTF will not record menu, component, and page metadata correctly.

**Image: Web Profile Configuration - Debugging page**

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

---

### Defining PTF Configuration Options

This section discusses the page used to configure PTF options.

#### Defining Configuration Options

Use the Define Configuration Options page (PSPTTSTCONFIG) to specify whether to allow untrusted SSL certificates, as well as record options and process server list.
Navigation

PeopleTools, Lifecycle Tools, Test Framework, Define Configuration Options

Image: Define Configuration Options Page

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

<table>
<thead>
<tr>
<th>Configuration Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow Untrusted SSL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Record Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Page Prompt</td>
</tr>
<tr>
<td>Use Message Recognition</td>
</tr>
<tr>
<td>Use Process Run</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Execution Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Server List</td>
</tr>
<tr>
<td>Session Name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>External Command Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command List</td>
</tr>
<tr>
<td>Command Path</td>
</tr>
<tr>
<td>Command File Name</td>
</tr>
<tr>
<td>Terminal (Seconds)</td>
</tr>
<tr>
<td>OPTIONS</td>
</tr>
<tr>
<td>OPTIONS</td>
</tr>
</tbody>
</table>

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 Page.

Use Message Recognition

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

Use Process Run

Select to use Process Run steps during recording in place of menu navigation. The Use Process Run option is also available on the PTF Recorder toolbar. The option selected here is the default for all users in this environment. The option selected on the PTF Test Recorder toolbar overrides this selection for that recording session. When this option is selected, PTF will recognize when a user accesses a run page and will populate the Process.Run parameters with the process information. All
actions in the run page or process monitor page will be ignored, because those action will be handled by the Run.Process in execution.

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

See Configuring Execution Options.

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

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

This example illustrates the error message received when PTF is trying to do a connection with Untrusted SSL Certificate.

## External Command Processing Section

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

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

<table>
<thead>
<tr>
<th><strong>Command Name</strong></th>
<th>Enter the name to be used in PTF for the command line program. The command name will be all caps.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command Path</strong></td>
<td>Enter the path where the command line program is located.</td>
</tr>
<tr>
<td><strong>Command File Name</strong></td>
<td>Enter the name of the command line file.</td>
</tr>
<tr>
<td><strong>Timeout</strong></td>
<td>Enter the time in seconds before a command will time out.</td>
</tr>
</tbody>
</table>

See Command.

---

## 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.
If Microsoft .NET Framework v4.0 is not present, the PTF Installer returns the following error during installation:

**Image: Microsoft .NET Framework error message**

This example illustrates the message received when Microsoft .NET Framework Version 3.5 is not present in the environment.

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

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

**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://myserver.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:

**Image: Microsoft Internet Explorer Local intranet dialog box**

This example illustrates the Microsoft Internet Explorer Local intranet dialog box

---

**Configuring Browser Settings with Windows 7**

If you are using Windows 7, confirm the following settings:
• Enable Protected Mode should be set on.

In IE select Tools, Internet Options, Security, highlight Internet and select Enable protected Mode

**Image: Enable Protected Mode on**

This example illustrates setting Enable Protected Mode on.

• Browser zoom is set to 100%

Check the lower right-hand corner of the browser and make sure it displays 100% for the zoom level.

## 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.

6. Click the Install button.

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

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:

myserver.us.oracle.com:443

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

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

https://<machine_name>:<https_port>/PSIGW/HttpListeningConnector

For example:

https://myserver.us.oracle.com:443/PSIGW/HttpListeningConnector

**Use Proxy**

Select this field if using a proxy server.

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

**Proxy Information**

Click this link to enter details for the proxy server.

Enter the following information for the proxy server:

- **Server**: Enter the server name
- **Port**: Enter the server port.
- **User**: Enter the user ID for the proxy server.
  
  If you use network authentication, use the DOMAIN \USER format.
  
  - **Password**: Enter the password.

**Node ID**

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

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

See Verifying Integration Broker Setup.

**User**

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

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


**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.

**Image: Example of a Completed Environment Signon Dialog Box**

This example illustrates a completed environment signon dialog box.

![Image of Example of a Completed Environment Signon Dialog Box](image1.png)

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:

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

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.

![Image of Example of a Completed Environment Signon Dialog Box with Node ID Specified](image2.png)
**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:

**Image: Signon Error Message**

This example illustrates a Signon error message when PTF security has not been configured correctly.

Possible causes and solutions for this error are:

- The user ID and password you entered are not valid for the PeopleSoft application corresponding to the entry in the Server:Port field.

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

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

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

**Image: Wrong HTTPS Error Message**

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

The default port is 443. If a different port was specified during installation, you will need to contact your system administrator to determine the correct port number.
If you receive the following error message, select Allow Untrusted SSL on the Configuration Options page.

**Image: Untrusted SSL Certificate Error Message**

This example illustrates the error message received when Untrusted SSL Certificate has not been selected and no certificate has been installed.

See Defining PTF Configuration Options.

**Selecting a PTF Environment**

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

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

**Image: New PeopleSoft Test Framework - Signon Dialog Box**

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

If you have created other environment signons, click the Previous button to select another environment signon.
Click the Edit button to edit the currently selected environment signon.

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

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

![PeopleSoft Test Framework - Signon](image_url)

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.

**Image: Local Options Dialog Box**

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

![Local Options Dialog Box](image)

**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.

**Auto-Check Syntax**
Select Yes to be prompted to check syntax every time you save a test.

---

**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.

**Image: Execution Options dialog – Options tab**

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

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](Image)
  - Click to add a new execution option.

- ![Delete](Image)
  - Click to remove an execution option from the list.

- ![Accept](Image)
  - Click to save changes and close the dialog box.

- ![Cancel](Image)
  - 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 Defining Configuration Options.

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:

**Image: Execution Options dialog – Advanced Options tab**

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

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

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

The following fields are on the Advanced Options tab:

**Portal Name**

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

**Note:** The URL Type is saved in upper case.

**URL**

Enter the portal URL.

The portal URL is entered in the following format:

http://webserver/psp/domain/portalname/node
For example: http://myserver.us.oracle.com:8010/psp/ps/
EMPLOYEE/QE_LOCAL/

Note: The ending backslash / is optional

<table>
<thead>
<tr>
<th>Enable Persistent Variables</th>
<th>Select to enable saving and using persistent variables. Selecting this option enables the other fields in this group.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>See Using Persistent Variables</td>
</tr>
<tr>
<td>By Execution Option Name</td>
<td>Persistent variables are stored in the database keyed by execution option name. Persistent variables can also be keyed by User ID, machine name or both.</td>
</tr>
<tr>
<td>By User ID</td>
<td>Select to store and retrieve persistent variables by PTF user ID.</td>
</tr>
<tr>
<td>By Machine Name Used for Playback</td>
<td>Select to store and retrieve persistent variables by machine name.</td>
</tr>
</tbody>
</table>
PeopleTools Tab

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

Image: Execution Options dialog – PeopleTools tab

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

The 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.
**Log Export**

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

**Image: Execution Options dialog - Log Export tab**

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

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

- Log is accessible from any browser.
- PTF client is not required to verify test results.
- Logs will be available even after the environment or database is brought down or upgraded.
- Since the log is in XML format, you can write customized utilities to parse or interpret the logs as needed.
Use Export Log
Select Yes to activate the export log functionality.

Path
Specify the shared drive to store the log files.

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

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

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

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

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

\(<\text{TEST\_NAME}\>\-\<\text{LOG\_ID}\>-T\<\text{YYYYMMDD}\>\_-\<\text{HHMMSS}\>\)

Example: FSCM_INS_VER-LOG5-T20121128_163649.XML

Export Images
If set to Yes the Image will be saved in the same Log folder as the XML Log in the format below:

\(<\text{TEST\_NAME}\>\-\<\text{LOG\_ID}\>-\<\text{OPTIONAL\_TIME\_STAMP}\>-\<\text{LINE\_NUMBER\_AS\_SHOWN\_IN\_LOG}\>

Example: FSCM_INS_VER-LOG5-T20121128_163649-Line30.JPEG
This is an example of a formatted log opened in the browser:

**Image: Sample export log displayed in the browser**

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

![PTF Execution Log for the test "CHART_TYPE"](image)

**Log Info**

- Log Caption: 2. PTFTRN 2012-12-18 11:38
- Case Name: DEFAULT
- Machine Name: MYSERVER
- UserID: PTFTRN

**Execution Options**

- Execution Option Name: DEMO
- Database Name: DEMO
- Database Platform: 
- Database Server: 
- Tools Path: D:\PT8.53
- PIA URL: http://MYSERVER.us.oracle.com:8010/bsp/ps/?cmd=logout
- User ID: PTFTRN

**System Info**

- Screen resolution: 1024 X 768 pixels
- Processor: Intel 586.11.266x2

**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 **Execution**.

**Related Links**

- Use Execution Options
Configuring Execution Options in PeopleSoft Internet Architecture

This section describes how to define execution options using the Define Execution Options component in PIA.

You can also define and modify execution options using the PTF client.

The Define Execution Options component has 4 pages:

- Options
- Advanced Options
- PeopleTools
- Export Log

**Note:** Only a user with the PTF Administrator role is able to access the Define Execution Options component.

Configuring Execution Options

Use the Execution options (PSPTTSTEXEOPTIONS) page to define PTF configuration options.

**Navigation**

PeopleTools, Lifecycle Tools, Test Framework, Define Execution Options
This example shows the Execution Options component - Options page in PIA:

**Image: Define Execution Options Page**

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

<table>
<thead>
<tr>
<th>Options</th>
<th>Advanced Options</th>
<th>PeopleTools</th>
<th>Export Log</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>DEMO</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Prompt</th>
<th>Nb</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Application URL</th>
<th><a href="http://myserver.us.oracle.com:3010/pspp/QEDMO/?cmd=logout">http://myserver.us.oracle.com:3010/pspp/QEDMO/?cmd=logout</a></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>User ID</th>
<th>QEDMO</th>
</tr>
</thead>
</table>

**Password**

Password cannot be modified in PIA. It can only be modified from the Test Framework client.

<table>
<thead>
<tr>
<th>Process Server</th>
<th>PSNT</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Date Format</th>
<th>MM/DD/YYYY</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Log Folder</th>
<th>PBOOKS</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>'Verbose'</th>
<th>Yes</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>'Skip Save'</th>
<th>Nb</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>'Skip Run Request'</th>
<th>Nb</th>
</tr>
</thead>
</table>

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.

**Prompt**

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 ID**

Enter a valid user name for the application database.

**Process Scheduler**

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

**Date Format**

Select a date format.

**Log Folder**

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.

**Skip Save**

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

**Skip Run Request**

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

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

### Configuring Advanced Options

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

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

**Navigation**

PeopleTools, Lifecycle Tools, Test Framework, Define Execution options

Select the Advanced Options tab.

**Image: Define Execution Options-Advanced Options Page**

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

The fields on the Advanced Options page are the same as the fields on the Advanced Options tab in the PTF client.
**Portal Name**
Enter a valid portal name or select from a list of previously added portal names.

*Note:* The URL Type is saved in upper case.

**URL**
Enter the portal URL.

The portal URL is entered in the following format:

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

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

*Note:* The ending backslash / is optional

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

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

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

**By Machine Name**
Select to store and retrieve persistent variables by machine name.

See Configuring Execution Options in PTF Client

**Configuring PeopleTools Options**
Use the PeopleTools page (PSPTSTTOOLOPTIONS) to supply the information required to connect to DataMover.

**Navigation**
PeopleTools, Lifecycle Tools, Test Framework, Define Execution options

Select the PeopleTools tab.
This example shows the Execution Options component – PeopleTools page:

**Image: Define Execution Options-PeopleTools Page**

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

<table>
<thead>
<tr>
<th><strong>Tools Path</strong></th>
<th>D:\PT8.53</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Connection Type</strong></td>
<td>ORACLE</td>
</tr>
<tr>
<td><strong>Server Name</strong></td>
<td>DEMO</td>
</tr>
<tr>
<td><strong>Database Name</strong></td>
<td>DEMO</td>
</tr>
<tr>
<td><strong>User ID</strong></td>
<td>QEDMO</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Password cannot be modified in PIA, it can only be modified from the Test Framework client.</td>
</tr>
</tbody>
</table>

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

**Tools Path (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.

**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.
Configuring Export Log Options

Navigation

PeopleTools, Lifecycle Tools, Test Framework, Define Execution options

Select the Export Log tab.

Use the Export Log page (PSPTTSTLOGOPTIONS) to automatically archive the result logs to a file system in XML + XSL format at the end of each test. This option provides the following:

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

This example shows the Execution Options component – Export Log page.

**Image: Define Execution Options - Export Log page**

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

<table>
<thead>
<tr>
<th>Options</th>
<th>Advanced Options</th>
<th>PeopleTools</th>
<th>Export Log</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>DEMO</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Use Log Export</strong></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Path</strong></td>
<td>D:\pt\logs%test%</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Style Sheet File</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Add Time Stamp</strong></td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Export Images</strong></td>
<td>No</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Use Export Log** Select Yes to activate the export log functionality.

**Path** Specify the shared drive to store the log files.

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

**Style Sheet File** (Optional) Specify the path to the stylesheet. If specified, the exported XML log will be saved with the stylesheet.
Note: Ensure that the shared drive path allows users full read/write access. The path can be customized using existing system variables. See System Variables

Add Time Stamp

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

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

Example: FSCM_INS_VER-LOG5-T20121128_163649.XML

Export Images

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

<TEST_NAME>-<LOG_ID>-<OPTIONAL_TIME_STAMP>-<LINE_NUMBER_AS_SHOWN_IN_LOG>

Example: FSCM_INS_VER-LOG5-T20121128_163649-Line30.JPG
Chapter 3

Using PeopleSoft Test Framework

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:

**Image: Example of the PTF Explorer User interface for the PB_Demo Environment**

This example illustrates 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.
- Rename tests and test cases.
- 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.
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:

<table>
<thead>
<tr>
<th>Test Menu Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Opens the selected test or test case.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the selected test.</td>
</tr>
<tr>
<td>Rename</td>
<td>Select to rename the selected test or test case.</td>
</tr>
<tr>
<td>Refresh Selection</td>
<td>Refreshes the view for the selected test.</td>
</tr>
<tr>
<td>Expand Selection</td>
<td>Expands all the branches in the selected node.</td>
</tr>
<tr>
<td>Collapse Selection</td>
<td>Collapses all the branches in the selected node.</td>
</tr>
</tbody>
</table>

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

This table lists the additional PTF Explorer Edit menu command:

<table>
<thead>
<tr>
<th>Edit Menu Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy Link to Clipboard</td>
<td>Copies the link to the selected test to the clipboard. You can use this information in conjunction with the Quick Open command.</td>
</tr>
</tbody>
</table>

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

This table describes the Create menu commands:

<table>
<thead>
<tr>
<th>Create Menu Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Folder</td>
<td>Creates a new folder within the selected folder.</td>
</tr>
<tr>
<td>Test</td>
<td>Creates a new test in the selected folder.</td>
</tr>
<tr>
<td>Shell Test</td>
<td>Creates a shell test in the selected folder.</td>
</tr>
</tbody>
</table>

This table describes the PTF Explorer Tools menu commands:
### Tools Menu Command

<table>
<thead>
<tr>
<th><strong>Tools Menu Command</strong></th>
<th><strong>Usage</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>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.</td>
</tr>
<tr>
<td>Log Manager</td>
<td>Opens the Log Manager tool, which enables you to delete logs from tests.</td>
</tr>
</tbody>
</table>

This table describes the PTF Explorer Window menu command:

<table>
<thead>
<tr>
<th><strong>Window Menu Command</strong></th>
<th><strong>Usage</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Quick Open</td>
<td>Opens a test, test case or a text execution log with data that was copied to the clipboard using the Copy Link to Clipboard command.</td>
</tr>
</tbody>
</table>

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.

### PTF Explorer Tree Functionality

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

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

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

### Using the Test Editor

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

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

This example shows the PTF Test Editor:

**Image: PTF Test Editor**

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

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:
<table>
<thead>
<tr>
<th><strong>PTF Menu Commands</strong></th>
<th><strong>Usage</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Options</td>
<td>Open the Local Options dialog.</td>
</tr>
<tr>
<td>Execution Options</td>
<td>Open the Execution Options dialog.</td>
</tr>
</tbody>
</table>

This table describes the Test Editor Test menu commands:

<table>
<thead>
<tr>
<th><strong>Test Menu Command</strong></th>
<th><strong>Usage</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Save</td>
<td>Saves the current test.</td>
</tr>
<tr>
<td>Save As</td>
<td>Creates a copy of the current test with a new name.</td>
</tr>
<tr>
<td>Test Case Save As</td>
<td>Creates a test case as a copy of the current test case.</td>
</tr>
<tr>
<td>Export</td>
<td>Export test cases and test case values to a character-delimited text file, such as comma-separated values (csv) file.</td>
</tr>
<tr>
<td>Import</td>
<td>Import test cases and test case values from a character-delimited text file, such as comma-separated values (csv) file.</td>
</tr>
<tr>
<td>Copy Link to Clipboard</td>
<td>Copies a link to the test to the clipboard.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Validate Test</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>See Creating Test Maintenance Reports.</td>
</tr>
<tr>
<td>Check Syntax</td>
<td>Validates the parameter syntax and values.</td>
</tr>
<tr>
<td></td>
<td>See PTF Test Language</td>
</tr>
<tr>
<td>Run</td>
<td>Runs the current test.</td>
</tr>
<tr>
<td>Pause</td>
<td>Pauses execution of the test.</td>
</tr>
<tr>
<td>End</td>
<td>Stops execution of the test.</td>
</tr>
<tr>
<td>Open Test Recorder</td>
<td>Launches the Test Recorder.</td>
</tr>
</tbody>
</table>
The Test Editor Edit menu contains standard Microsoft edit commands, such as Cut, Copy, Paste, and Delete, and the following PTF commands:

This table lists the additional PTF commands:

<table>
<thead>
<tr>
<th>Edit Menu Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find</td>
<td>Finds occurrences of a specific text string in the current test. Select the List all MatchingSteps check box to create a list of matches.</td>
</tr>
<tr>
<td>Again</td>
<td>Searches for the Find string again.</td>
</tr>
</tbody>
</table>

This table describes Test Editor Debug menu commands:

<table>
<thead>
<tr>
<th>Debug Menu Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step Into</td>
<td>Executes the current step of the test and advances to the next step.</td>
</tr>
<tr>
<td>Step Over</td>
<td>Executes the current step of the test and advances to the next step, unless the next step calls another test. Steps over called tests.</td>
</tr>
<tr>
<td>Toggle Break</td>
<td>Sets a break point at the selected step or removes an existing breakpoint.</td>
</tr>
<tr>
<td>Clear All Breaks</td>
<td>Removes all break points.</td>
</tr>
<tr>
<td>Stop on Error</td>
<td>Stops execution if the test encounters an error.</td>
</tr>
<tr>
<td>Disable Screen Shots</td>
<td>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.</td>
</tr>
<tr>
<td>Highlight Errors</td>
<td>Highlights errors in the log in yellow.</td>
</tr>
<tr>
<td>Overwrite Log</td>
<td>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.</td>
</tr>
</tbody>
</table>

This table discusses the Test Editor Window menu commands:

<table>
<thead>
<tr>
<th>Window Menu Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quick Open</td>
<td>Using information from the Copy Link to Clipboard command, quickly opens a test without having to navigate to the log in PTF Explorer.</td>
</tr>
<tr>
<td>Window Menu Command</td>
<td>Usage</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------</td>
</tr>
<tr>
<td>Again</td>
<td>Searches for the Find string again.</td>
</tr>
</tbody>
</table>

**Test Editor Toolbar**

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

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

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

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

Image: Example of a Test Editor Test Window

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

Test Window Fields

The following fields appear in the test window:

Test Information Fields

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

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

Test Information Functions

Test Save As.
Click to access the Message Recognition dialog box.

Use the Message Recognition dialog box to define how PTF will respond to messages that are encountered during execution of the test. The Message Recognition icon contains a check mark when the message recognition is enabled for the test.
Open the Test Properties dialog. In the Language field, select the language for the PeopleSoft application. The default is English.

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

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

Select the Library Test checkbox to make the test a library test. See Using Library Tests.

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

**Test Case Information Fields**

- **Name**: Displays the name of the current test case. You can select a different test case using the drop-down list. When you create a test, the system automatically associates it with a test case named DEFAULT.
- **Description**: Brief description of the test case. You can enter a detailed description in the comments.

**Test Case Information Functions**

- **Test Case Save As**.
- **Create a new test case**.
- **Delete the selected test case**.
- **Open the Test Case Properties dialog**.
- **Open the Test Case Comments dialog**. You can enter a long description of the test case using Rich Text and images.
Test Window Toolbar

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

- **Active All**
  Selects or deselects the Active check box for all steps.

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

- **Delete Step**
  Deletes the current step.

- **Step Information**
  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:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq (sequence)</td>
<td>A system-generated sequence number. Test steps execute according to Seq order. When you move, add, or delete a step, Seq is refreshed.</td>
</tr>
<tr>
<td>ID</td>
<td>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.</td>
</tr>
<tr>
<td>Comment</td>
<td>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.</td>
</tr>
<tr>
<td>Active</td>
<td>Deselect this field to inactivate a step. PTF will skip inactive steps when the test runs. PTF syntax check is also skipped on inactive steps. Each step is active by default. This field is grayed for inactive steps.</td>
</tr>
<tr>
<td>Scroll ID</td>
<td>This field is only required for scroll handling.</td>
</tr>
<tr>
<td>Type</td>
<td>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.</td>
</tr>
</tbody>
</table>
**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.

---

**Related Links**
- PTF Test Language
- Creating Tests

---

**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.

Image: Test RecorderToolbar

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

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

The Test Recorder toolbar 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.
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 Verify.

Open field pop ups or organization chart menu items.

See MouseOver.

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 Get_Property.

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 Exists.

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.
Chapter 3 Using PeopleSoft Test Framework

See **Conditional**.

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 **Variable, System Variables**.

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 **Incorporating Scroll Handling**.

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.

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

Return the label value. Drag and drop over the HTML table label. You are prompted for return variable that will contain the HTML label text.

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

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

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

  Use explicit menu navigation.

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

  See Page.

- **Message Recognition**

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

See Handling Application Messages.

• Run Control Recognition

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

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

See Run

• File Download Prompt

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

See Download

**Related Links**

Recording Tests

**Using the Log Viewer**

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

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

You can also view a log by opening it from PTF Explorer.
This example shows the Log Viewer:

**Image: Log Viewer**

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

The Log Viewer has three panes:

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

**Log Viewer Menus**

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

This table describes the Log menu commands:

<table>
<thead>
<tr>
<th>Log Menu Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find</td>
<td>Find text within a log.</td>
</tr>
<tr>
<td>Expand All</td>
<td>Expands all the selections in the log.</td>
</tr>
</tbody>
</table>
**Log Menu Command** | **Usage**
---|---
Collapse All | Collapses all the selections in the log.
Change Log View | Toggles between a view that shows log results as colored flags and a view that shows log results as shaded labels.
Highlight Errors | Highlights log entries with errors in yellow.
Copy Link to Clipboard | Copies a link to the log to the clipboard. You can send this information to another user, who can use the Window, Quick Open feature to open the log without having to navigate to it in PTF Explorer.

This table describes the Detail menu commands:

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

This table describes the Window menu commands:

<table>
<thead>
<tr>
<th><strong>Window Menu Command</strong></th>
<th><strong>Usage</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Quick Open</td>
<td>Using information from the Copy Link to Clipboard command, quickly opens a log without having to navigate to the log in PTF Explorer.</td>
</tr>
<tr>
<td>Close All</td>
<td>Closes all open windows.</td>
</tr>
</tbody>
</table>

**Related Links**
- Reviewing Test Logs
- Interpreting Logs
Chapter 4

Creating Tests and Test Cases

Creating Tests and Test Cases

This chapter discusses how to:

• Create tests.
• Record tests.
• Create test cases.
• Export and import test cases.
• Execute tests.
• Review test logs.
• Export test logs to XML.
• Organize tests.

Creating Tests

This section discusses how to:

• Create a new folder.
• Create a new test.
• Name tests.
• Copy a test.
• Rename 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.

Renaming a Test

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

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

The test is renamed.

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:

**Image: Example of Test Steps**

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

<table>
<thead>
<tr>
<th>Seq</th>
<th>ID</th>
<th>Comment</th>
<th>Action</th>
<th>Scroll ID</th>
<th>Type</th>
<th>Action</th>
<th>Recognitions</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>Browser</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>Link</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>Pued</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>Button</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>Link</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>Link</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>Browser</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td>Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td>Pued</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td>Pued</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td>Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 [Using the Test Editor](#).

**Test Action Tools**

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

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

**Related Links**

[Using the PTF Test Recorder](#)
PTF Test Language

Recording a Test

To record a test:

1. Create a new test or open an existing test.

2. If you are recording within an existing test, highlight a test step. The system inserts the steps in the new recording following the highlighted step. If you are recording a new test, recording typically begins at Step 2. By default, Step 1 is Browser.Start.

3. With a test open, select Test, Open Test Recorder or click the Show Test Recorder button in the toolbar.

4. The PTF Test Recorder toolbar appears.

See Using the PTF Test Recorder.

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

To hook a browser, you can either:

- Click the Start Web Client icon on the recorder toolbar. This starts the web browser with the default test application URL from the current execution option.

- Select a browser window with the test application open and hook the application by dragging the Select Browser icon to the browser window.

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

6. Click the Start Recording icon in the Test Recorder toolbar to begin recording.

7. Perform the test steps in the PeopleSoft application.

8. As needed, insert additional steps using the Test Recorder test action tools.

9. Modify steps or add step comments using the Test Recorder step modification tools.

10. Click the Stop Recording icon in the Test Recorder toolbar to end the recording and write the steps to the test.

11. Save the test.

Related Links

Using the PTF Test Recorder
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 Properties icon to enter a long description of the test case.
   It is a good practice to provide a good description for a test case. Test names are limited in length and provide the user with little information.
5. (Optional) Click the Test Case Comment icon to add a comment for the test case.
6. The new test case provides a set of new, empty Value fields for all the steps of the test.
   Enter a new value for each step that requires a value.
7. Save the test case.

Image: New test case with blank values

This example illustrates a new test case with blank values.

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

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

- Comma
- Semicolon
- Vertical pipe
- Tab

Exporting Test Cases

To export test cases:

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

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

4. Specify whether to export one or all test cases.
5. Click Start.
The bar along the bottom of the dialog box denotes the progress of the export.

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

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

Importing Test Cases

To import test cases.

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

**Image: Test Case Import dialog box**

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

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

5. Select the test cases to import.

6. Click Start.

---

**Executing Tests**

This section describes how to execute tests and test cases.

**Executing a Test**

To execute a test:

1. With a test open in PTF, select Test, Run.
   
   Alternatively, you can press F5 or click the Run button.

2. If Yes is specified in the Prompt for Options field for the default execution option, then the Execution Options dialog box appears.
Select an execution option from the list and click Accept.

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

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

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

**Related Links**
Reviewing Test Logs

**Executing a Test Case**

To execute a test case:

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

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

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

2. Select Test, Run.

3. Review the log in the Log Viewer.

**Related Links**
Reviewing Test Logs

**Executing a Test from 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**

*–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 Creating a Connection to a PTF Environment.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>–CP=</td>
<td>Specify the user password.</td>
</tr>
<tr>
<td>–TST=</td>
<td>Specify the test name.</td>
</tr>
<tr>
<td>–TC=</td>
<td>Specify the test case name.</td>
</tr>
<tr>
<td>–PFX=</td>
<td>(Optional) Specify the prefix.</td>
</tr>
<tr>
<td>–EXO=</td>
<td>Specify the execution option to be used in the execution.</td>
</tr>
</tbody>
</table>

**Note:** The name of the execution option must not contain the following characters: space & ? / * < > ' "

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>–LOG=</td>
<td>(Optional) Specify the name for the log. The default is unattended.log.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>–CS=</td>
<td>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.</td>
</tr>
<tr>
<td>–CNO=</td>
<td>Specify the node name.</td>
</tr>
<tr>
<td>–CO=</td>
<td>Specify the user name.</td>
</tr>
<tr>
<td>–PS=</td>
<td>(Optional) Specify the ProxyServer:Port.</td>
</tr>
<tr>
<td>–PU=</td>
<td>(Optional) Specify the proxy user. If you use network authentication, use the DOMAIN\USER format.</td>
</tr>
<tr>
<td>–PP=</td>
<td>(Optional) Specify the proxy password.</td>
</tr>
</tbody>
</table>
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
-e -P=pwd123 -C=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.

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

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

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

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

**Image: Example of Log Entries in a Test Log**

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

```
| Frame set to " |
| Ignore step 2 |
| Ignore step 3 |
| Ignore step 4 |
| Ignore step 5 |
| Ignore step 6 |
| Ignore step 7 |
| Frame set to 'TargetContent' |
| Ignore step 9 |
| PagePrompt |
| Goto URL |
| PosPanelPrompt: Trying to do the following action = update |
| Click : #ICAdvSearch [HTMLLink] |
| Text Set_Value [Name=PSOPRDEFN_SRCH_OPRID] -> QEDMO |
| Click : #ICSearch [PushButton] |
| Text Set_Value [Name=DERIVED_CLONE_OPRID] -> JGDMO |
| Text Set_Value [Name=DERIVED_CLONE_OPRDEFNDDESC] -> Jim's ID |
| Pwd Set_Value [Name=DERIVED_CLONE_OPERPSWD] -> ******** |
| Pwd Set_Value [Name=DERIVED_CLONE_OPERPSWDCONF] -> ******** |
| Page.Save |
| Click : #ICSave [PushButton] |
| LogScreenShot |
| ================ PageSave: ERROR ================ |
```

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:

- ![Info](image)
  - Information message only.

- ![Info](image)
  - Information message only.

- ![Pass](image)
  - Step was successful.

- ![Warning](image)
  - Step was successful, but with a warning.
Step failed.

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

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

Related Links
Interpreting Logs
Reviewing Test Logs

Exporting Test Logs to XML

PTF provides the ability to export test log to XML. Once the logs have been exported to XML, the user can compare the results in a text editor, such as NotePad or EditPlus. The user can select which fields to exclude from the XML.

To export a test log to XML:

1. In PTF Explorer, open the logs folder.
2. Right-click on a specific log.
3. Select Export.
The Export Log dialog box is displayed:

**Image: Export Log Dialog Box**

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

![Export Log Dialog Box](image)

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

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

**File Path**

Enter the file path. The file name is automatically generated based on the test name and log number. For example PB_RUN_PROCESS-LOG-6.xml.

**Select Elements to Export Column**

By default all items are selected, click the check box to deselect specific items.

**Preview**

Select this check box to preview the xml.

**Export Images**

Select this check box to include images in the exported file.
Organizing Tests In PTF Explorer

This section discusses how to:

- Cut and paste multiple tests or folders.
- Delete multiple tests or folders.
- Expand or collapse multiple tests or folders.

Cutting and Pasting Multiple Tests or Folders

PTF Administrator and PTF Editors can cut and paste tests and folders. A user possessing only the PTF User role is not be able to cut folders residing outside of the myFolder folder. PTF User role can paste folders or tests outside of the myFolder folder.

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

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

All contents are moved to the target folder location.

Deleting Multiple Tests or Folders

PTF Administrator and PTF Editors can delete tests and folders. A user possessing only the PTF User role is not be able to delete folders residing outside of the myFolder folder.

To delete multiple tests or folders:

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

The folders or tests are deleted.

Expanding or Collapsing Tests and Folders

To expand or collapse multiple tests or folders:

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

PTF expands or collapses all selected tests and folders at the level of selection in the tree.
Chapter 5

Developing and Debugging Tests

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 test strings as parameters for functions.
- 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 Tools, Message to open the Message tool.
Click and drag the Object Properties icon and hover over a browser object to view details about that object in the Message window.

**Image: Message Tool ID Values**

This example illustrates Message tool ID values.

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

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

**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:

**Image: Example of the HTML Browser**

This example illustrates the HTML Browser Object Properties window.

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.

**Capturing index for non-unique objects using the HTML Browser**

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

1. In PTF Client, select Tools > Message to open the Message tool.
2. Start your recording.

3. Open the tree to the desired location.

4. Click the Object Properties icon in the Message Toolbar and drag it to the leaf image on the tree.

5. Access the Message tool and select Menu > HTML Browser > Show.

6. Right-click on the nameProp in the Object Properties window and select Get Index for.

7. The index will be added to the object Properties.

**Image: Object properties Dialog Box**

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

---

**Using Reserved Words**

This section discusses how to use reserved words.

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

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

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

**Image: Example of a Test Step with Static Data**

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

| Text | Set_Value | Name=PA_PROP_VOUCH_CREAT_DT | 06/30/2010 |

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:

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

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

| Text | Set_Value | Name=PA_PROP_VOUCH_CREAT_DT | #TODAY |

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

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

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

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

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

**Image: Example of a Step that Sets a Value**

This example illustrates a step that sets a value.

| Text | Set_Value | PA_CALCULATION_DESCR | Sample pension calc |

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

**Image: Example of a Step that Verifies a Value**

This example illustrates a step that verifies a value.

| Text | Set_Value | PA_CALCULATION_DESCR | #CHECK#Sample pension calc |

**Related Links**

- Creating Test Cases
- Reserved Words
Using Variables

This section discusses how to use variables.

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

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

Store a value for a variable either by placing the `ret=&varname` parameter in the 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.

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

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

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

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

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

**Image: Example of Using a Variable**

This example illustrates using a variable to verify the value of a property on a page.
Variable string length is limited to 254 characters. For any variable longer than 254 characters, you will need to wrap the string into multiple variables and concatenate them for usage in verification or set value steps.

**Image: Example Using Multiple Variables for String over 254 Characters**

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Set Value</th>
<th>ivar1</th>
<th>First 254 characters string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Set Value</td>
<td>ivar2</td>
<td>The remaining characters string</td>
</tr>
<tr>
<td>Variable</td>
<td>Set Value</td>
<td>ivar3</td>
<td>concat(ivar1,ivar2)</td>
</tr>
<tr>
<td>LongText</td>
<td>Verify</td>
<td>None+PSU INSTL EXP DESCRLNG</td>
<td>&amp;ivar3</td>
</tr>
</tbody>
</table>

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

**Wrapping Variables in Quotes**

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

Quotes are only necessary in the following situations:

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

**Assigning Variable in the Recognition Field**

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

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

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

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

<table>
<thead>
<tr>
<th>Recognition</th>
<th>Expect Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;MyVar=&amp;PriorVar</td>
<td>Contents of the &amp;PriorVar variable</td>
</tr>
<tr>
<td>&amp;MyVar=&quot;&amp;PriorVar&quot;</td>
<td>&amp;PriorVar</td>
</tr>
<tr>
<td>&amp;MyVar=&quot;Hello&quot;</td>
<td>Hello</td>
</tr>
</tbody>
</table>
Assigning Variables in the Value field

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

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

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

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

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

<table>
<thead>
<tr>
<th>Recognition</th>
<th>Value</th>
<th>Expected Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;MyVar</td>
<td>&amp;PriorVar</td>
<td>Contents of &amp;PriorVar variable</td>
</tr>
<tr>
<td>&amp;MyVar</td>
<td>CONCAT(&quot;&amp;PriorVar&quot;)</td>
<td>&amp;PriorVar</td>
</tr>
<tr>
<td>&amp;MyVar</td>
<td>Hello</td>
<td>Hello</td>
</tr>
<tr>
<td>&amp;MyVar</td>
<td>“Hello”</td>
<td>“Hello”</td>
</tr>
<tr>
<td>&amp;MyVar</td>
<td>““Hello”“”</td>
<td>““Hello”“”</td>
</tr>
</tbody>
</table>

Related Links
Parameters
Variable
Conditional

Using Text Strings as Parameters in Functions

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

...
• Text strings supplied as parameters for a function must always be wrapped in quotes, regardless of whether they appear in the Recognition field or the Value field.

• The CONCAT() function should be used to return a text string containing special characters such as ampersands that could be interpreted as a variable reference.

• Double quotes ("""”) should be used to return one quotes character. Consequently, when a final text string result is expected to include leading and trailing quotes characters, the parameter supplied should be lead and trailed by three quotes characters, for example, """"Hello"""".

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

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

<table>
<thead>
<tr>
<th>Recognition</th>
<th>Expected Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONCAT(&quot;&amp;MyVar = &quot;</td>
<td>&amp;MyVar = Hello</td>
</tr>
<tr>
<td></td>
<td>&amp;MyVar)</td>
</tr>
<tr>
<td>SUBSTR(&quot;Hello&quot;</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Hello</td>
</tr>
<tr>
<td>SUBSTR(&quot;&quot;&quot;&quot;Hello &quot;&quot;&quot;&quot;</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>&quot;Hello&quot;</td>
</tr>
<tr>
<td>INSTR(&quot;Hello&quot;</td>
<td>&quot;e&quot; )</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>INSTR(&quot;&quot;&quot;&quot;Hello&quot;&quot;&quot;&quot;</td>
<td>&quot;e&quot; )</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td>INSTR(&quot;&quot;&quot;&quot;Hello&quot;&quot;&quot;&quot;</td>
<td>&quot;&quot;e&quot;&quot;&quot;&quot; )</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>INSTR(Hello</td>
<td>e )</td>
</tr>
<tr>
<td></td>
<td>PTF returns error message</td>
</tr>
</tbody>
</table>

**Using Persistent Variables**

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

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

There are two key elements required to use persistent variables:

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

**Setting Variable Option in the Test**

To set the variable option in the test, use the Test Property icon.
1. Create a new test or open an existing test.
2. Click on the Test Properties icon.

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

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

**Setting Persistent Variable Options**

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

See Configuring Execution Options in PTF Client or Configuring Advanced Options

**Managing Persistent Variables**

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

**Navigation**

PeopleTools, Lifecycle Tools, Test Framework, Manage Persistent Variables

**Image: Manage Persistent Variables page**

This example illustrates the fields and controls on the Manage Persistent Variables page. You can find definitions for the fields and controls later on this page.

<table>
<thead>
<tr>
<th>Name</th>
<th>User ID</th>
<th>Machine Name</th>
<th>Personal</th>
<th>Find</th>
<th>View</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1B65301</td>
<td>PTFTRN</td>
<td>&amp;ROLE</td>
<td></td>
<td></td>
<td></td>
<td>ADS Designer</td>
</tr>
<tr>
<td>T1B65301</td>
<td>PTFTRN</td>
<td>&amp;USERID</td>
<td></td>
<td></td>
<td></td>
<td>PTINT</td>
</tr>
</tbody>
</table>

This page will list all the persistent variables stored in the database. The variable is stored when the test writes the variable to the database.
The variable is always stored with the name of the execution option. The User ID and Machine Name are dependent on the selection you made for the persistent variable on the Advanced Options page of the Execution Options.

You can delete persistent variables or modify the variable value.

**Example of a Test that uses Persistent Variables**

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

**Image: Example of using persistent variables in a test**

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

<table>
<thead>
<tr>
<th>Browser</th>
<th>Test_Name</th>
<th>Start_Login</th>
<th>Prompt</th>
<th>PromptOK</th>
<th>Message</th>
<th>Step 1: Read persistent variable from persistent variable table</th>
<th>Step 2: Write persistent variable to persistent variable table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page</td>
<td>Page</td>
<td>Page</td>
<td>Log</td>
<td>Test</td>
<td>Page</td>
<td>Log</td>
<td>Test</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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: Example of Step that Requires a Conditional Construct**

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

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

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

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

**Image: Example of Using Conditional Logic**

This example illustrates the steps necessary for the conditional logic.

![Example of Using Conditional Logic](image)

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

**Related Links**

Conditional

---

**Handling Application Messages**

This section discusses how to handle application messages.

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

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

12. Clear the Calculate all Plans checkbox. If you get the following warning, click OK:
   Warning - Remember all plans must be selected to ensure an accurate limit calculation (48,17)
When you record the test, the step that triggers the message and the step that clicks OK might look like this:

**Image: Example of steps that trigger and dismiss a warning**

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

<table>
<thead>
<tr>
<th>CheckBox</th>
<th>Set Value</th>
<th>Name</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button</td>
<td>Click</td>
<td>Name</td>
<td>#OK</td>
</tr>
</tbody>
</table>

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

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

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

1. With a test open, click the Message Recognition icon.
2. Enter the text of the message, or a portion of the text, in the Message field.
   - The following example uses the message catalog number rather than the full message text.
   - You can define message definitions at either the test or test case level. Message definitions defined at the test case level take precedence over message definitions defined at the test level.
3. Select which button the step should click.
4. Specify what action PTF should take after the button is clicked.
5. To delete a message, select the message line and press the Delete key.

This example shows the Message Recognition dialog box. In this example when the application message (1040,3) error occurs, PTF will click the `Cancel` button and abort the test execution.
Note: The Test Case Message Definitions takes precedence over the Test Message Definition.

Image: Message Recognition dialog box

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

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

<table>
<thead>
<tr>
<th>Message</th>
<th>Button</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1040.3]</td>
<td>OK</td>
<td>Continue</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Message</th>
<th>Button</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1040.3]</td>
<td>Cancel</td>
<td>Abort</td>
</tr>
<tr>
<td>[ ]</td>
<td>OK</td>
<td>Continue</td>
</tr>
</tbody>
</table>

**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**
Interpreting Logs

This section discusses how to interpret logs.

This table lists and describes common log messages:

<table>
<thead>
<tr>
<th>Status</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fail</td>
<td>Access is denied. Please check browser settings.</td>
<td>PTF is not able to read information from your browser. Check browser settings to make sure your browser is configured properly. See Configuring the Browser Security Settings.</td>
</tr>
</tbody>
</table>
### Status | Message | Description
--- | --- | ---
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 [Configuring the Browser Security Settings](#).

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 [Using the Message Tool](#).

- 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 [Interpreting Test Maintenance Reports](#).

---

### Incorporating Scroll Handling

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

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

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

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

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

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

**Image: Example of the Roles page**

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

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

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

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

<table>
<thead>
<tr>
<th>Scroll ID</th>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>Verify</td>
<td>Name=PSROLEUSER_VW_ROLENAME$0</td>
<td>PTF Administrator</td>
<td></td>
</tr>
</tbody>
</table>
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 Using the PTF Test Recorder.

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

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

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

Image: Example of the Message tool

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

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

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

<table>
<thead>
<tr>
<th>Scroll ID</th>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Name=PSROLE_USER_VW_ROLENAME[0]</td>
<td>PTF Administrator</td>
</tr>
</tbody>
</table>

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.

**Image: Example of a Scroll.Key_Set step**

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

<table>
<thead>
<tr>
<th>Scroll ID</th>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Scroll</td>
<td>Key_Set</td>
<td>Type=Text;Name=PSROLE_USER_VW_ROLENAME</td>
<td>PTF Administrator</td>
</tr>
</tbody>
</table>

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

**Creating an Action Step**

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

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

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

**Image: Example of an Action step**

This example illustrates an action step.

<table>
<thead>
<tr>
<th>Scroll ID</th>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Scroll</td>
<td>Action</td>
<td>ret=&amp;Scroll1</td>
<td>000</td>
</tr>
</tbody>
</table>

**Using the Index Variable**

Now that you have the row position stored in the index variable, you can use that value to reference other fields on that row.
For instance, suppose you want to verify the Dynamic checkbox. You can use the PTF Recorder or the Message tool to get its name and position and create a step similar this one:

**Image: Example of step using positional reference**

This example illustrates a step with a positional reference.

<table>
<thead>
<tr>
<th>CheckBox</th>
<th>Verify</th>
<th>Name=PSROLEUSER_VW_DYNAMIC_SW60</th>
<th>N</th>
</tr>
</thead>
</table>

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

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

**Image: Example of a step using an index variable**

This example illustrates a step using an index variable.

<table>
<thead>
<tr>
<th>1</th>
<th>Scroll</th>
<th>Key_Set</th>
<th>Type=Text;Name=PSROLEUSER_VW_ROLENAME</th>
<th>PTF_Administrator</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Scroll</td>
<td>Action</td>
<td>ret=&amp;Scroll1</td>
<td>PTF_Administrator</td>
</tr>
<tr>
<td></td>
<td>CheckBox</td>
<td>Verify</td>
<td>Name=PSROLEUSER_VW_DYNAMIC_SW60&amp;Scroll1</td>
<td>N</td>
</tr>
</tbody>
</table>

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

**Specifying the Scroll ID**

Assign a Scroll ID to group Scroll actions for each scroll. Use a different scroll ID and scroll variable for each different scroll area. You can assign any integer you like, as long as it is unique.

Scroll ID is a required field for Scroll actions.

If you are taking multiple actions in the same scroll, using the same scroll ID and scroll variable improves performance over using a new scroll ID.

In this example, the test defines two Action steps that both act on the same scroll. The first action verifies that the user profile does not contain the PTF Administrator role. The second action verifies that the user profile does contain the PTF User role. You would assign the same Scroll ID number to all four of the Scroll steps because they all act on the same scroll.

**Image: Example of assigning Scroll IDs to Scroll steps**

This example illustrates assigning scroll IDs to scroll steps.

<table>
<thead>
<tr>
<th>Scroll ID</th>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page</td>
<td>Go_To</td>
<td>Roles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Scroll</td>
<td>Key_Set</td>
<td>Type=Text;Name=PSROLEUSER_VW_ROLENAME</td>
<td>PTF_Administrator</td>
<td></td>
</tr>
<tr>
<td>1 Scroll</td>
<td>Action</td>
<td>ret=&amp;Scroll1</td>
<td>PTF_Administrator</td>
<td></td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>Scroll1 index variable = &amp;Scroll1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Scroll</td>
<td>Key_Set</td>
<td>Type=Text;Name=PSROLEUSER_VW_ROLENAME</td>
<td>PTF_User</td>
<td></td>
</tr>
<tr>
<td>1 Scroll</td>
<td>Action</td>
<td>ret=&amp;Scroll1</td>
<td>PTF_User</td>
<td></td>
</tr>
</tbody>
</table>
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.
Developing and Debugging Tests Chapter 5

The following example shows a parameter list:

**Image: Test Properties dialog box with a parameter list**

This example illustrates the fields and controls on the Test Properties dialog box with a parameter list.

![Test Properties dialog box with a parameter list](image)

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.

**Image: Test.Exec step with parameters**

This example illustrates a Test.Exec step with parameters.

![Test.Exec step with parameters](image)
In the library test, place a %param.parameter_name% construct wherever you want to use a parameter. The parameter is replaced at runtime by the text assigned in the calling test.

**Image: Library test steps with parameters**

This example illustrates library test steps with parameters.

<table>
<thead>
<tr>
<th>Test</th>
<th>Set Value</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td></td>
<td>%param.Name%</td>
</tr>
<tr>
<td>Pwd</td>
<td></td>
<td></td>
<td>%param.Pwd%</td>
</tr>
<tr>
<td>Variable</td>
<td></td>
<td></td>
<td>%OrderNo%=%param.OrderNo%</td>
</tr>
</tbody>
</table>

**Note:** All library parameters values need to be set in the calling test, variables are not automatically initialized. If the user makes a call to the library without passing a value for the parameter, the parameter value will be the parameter name. For example, if the parameter is UserName and the value is not set, the value be %param.UserName% not blank.

**Using Shell Tests**

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

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

- Execution actions - modify the behavior of tests during execution. Execution actions include Skip_PageSave, Skip_RunRequest, Skip_Login, 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
- 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

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 toolbar.
• Use Log Manager fields.
• 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 Manager 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.

**Image: Example of PTF Log Manager**

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

**Using Log Manager Toolbar**

The log manager toolbar has the following options:

**Retrieve**

Select to populate the selection pane based on the specified criteria.

**Note:** The user can cancel the retrieval process at any time by pressing the Abort button, and thus getting a partial result.

**Delete**

Select to delete the selected logs.

**Undelete**

Select to cancel logs marked for deletion. When you click Delete, the selected logs are only marked for deletion. They are not removed from the database until you click Purge Log. Until then, you can undelete logs.

**Purge**

Select to remove selected logs from the database if they are marked for deletion.
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.

By Log Status
When you select this check box, the statuses are available. Select the statuses to list and click the Refresh button in the Selection pane. The results are updated in the Selection pane.

**Extra Options**

Bypass Selection
Select to perform the selected action on all log entries listed according to the criteria, regardless of user selections.

Use Start Time
Select and enter a value to start the process at the designated time.

Using the Selection Pane

The selection pane displays the logs based on the selection criteria. By default all the logs are selected.

When you click Retrieve, the system populates this pane with logs based on the criteria you specified. Using the check boxes, select the logs that will be processed when you click Delete, Undelete, or Purge.
The selection pane contains a status bar that displays the number of logs selected, processed and errors, as well as buttons for:

- **Unselect All**: Click to unselect all logs.
- **Clear**: Click to clear the selection list.
- **Refresh**: Click to refresh the selection after changing the criteria.

**Note:** The Refresh is based on the new criteria only. If you change the folder path or test, you must use a new Retrieve.

All of the columns in the selection section are resizable.

**Example Purge**

When you select logs in the selection pane and then select Purge from the toolbar, the selection pane status is updated and the focus is set on the log lines being processed.

**Image: Example of PTF Log Manager after Purge**

This example illustrates PTF Log Manager after Purge.

**Using the Trace Pane**

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

**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.

**Related Links**

"Inserting Definitions Into Projects (PeopleTools 8.53: PeopleSoft Application Designer Developer's Guide)"
Chapter 7

Identifying Change Impacts

These topics provide an overview of change impacts and discuss how to:

- Define analysis rules.
- Create test maintenance reports.
- Interpret test maintenance reports.
- Understand test coverage reports.
- Create test coverage reports.
- Interpret test coverage reports.
- Run test details reports.
- Create a test compare report.
- 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.

**Defining Analysis Rules**

Use the Define Analysis Rules page (PSPTTSTANLMENU) to define the priority for each of the attribute checks in a test maintenance report.

**Navigation**

PeopleTools, Lifecycle Tools, Test Framework, Define Analysis Rules

**Image: Define Analysis Rules page**

This example illustrates the fields and controls on the Define Analysis Rules page. You can find definitions for the fields and controls later on this 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:

<table>
<thead>
<tr>
<th>Analysis Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M01</td>
<td>Menu does not exist</td>
</tr>
<tr>
<td>M02</td>
<td>Component deleted from Menu</td>
</tr>
<tr>
<td>M03</td>
<td>Component added to Menu</td>
</tr>
<tr>
<td>C01</td>
<td>Page deleted from Component</td>
</tr>
<tr>
<td>C02</td>
<td>Page added to Component</td>
</tr>
<tr>
<td>C03</td>
<td>Search Record changed on Component</td>
</tr>
<tr>
<td>P01</td>
<td>Field deleted on Page</td>
</tr>
<tr>
<td>P02</td>
<td>Required Field added to Page</td>
</tr>
<tr>
<td>P03</td>
<td>Fieldname changed on Page</td>
</tr>
<tr>
<td>P04</td>
<td>Field Type changed on Page</td>
</tr>
<tr>
<td>P05</td>
<td>Field Label changed on Page</td>
</tr>
<tr>
<td>P06</td>
<td>Non-Required Field added to Page</td>
</tr>
<tr>
<td>P07</td>
<td>Recname changed on Page</td>
</tr>
<tr>
<td>P08</td>
<td>Recname &amp; Fieldname changed on Page</td>
</tr>
<tr>
<td>R01</td>
<td>RecordField now required on Page</td>
</tr>
<tr>
<td>R02</td>
<td>RecordField no longer required on Page</td>
</tr>
<tr>
<td>R03</td>
<td>RecordField is now a Search Key</td>
</tr>
<tr>
<td>R04</td>
<td>RecordField no longer a Search Key</td>
</tr>
<tr>
<td>R05</td>
<td>RecordField is now a List Box Item</td>
</tr>
<tr>
<td>R06</td>
<td>RecordField no longer a List Box Item</td>
</tr>
</tbody>
</table>
Creating Test Maintenance Reports

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


The wizard consists of four steps:

- Step 1: Manual Tasks
- Step 2: Analyze Compare Data
- Step 3: Select an Analyzed Project
- Step 4: Generate Report

**Step 1 of 4: Manual Tasks**

For a given change project the tasks for Step 1 only need to be executed once.

The first step in creating a test maintenance report comprises five manual tasks that you will complete in Application Designer to create a compare report that PTF will use as a basis for the maintenance report.

Perform these tasks to create a compare report from a project file.

<table>
<thead>
<tr>
<th>Analysis Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F01</td>
<td>Field Type changed</td>
</tr>
<tr>
<td>F02</td>
<td>Field Length changed</td>
</tr>
<tr>
<td>F03</td>
<td>Field Format changed</td>
</tr>
<tr>
<td>F04</td>
<td>Field Decimal Positions changed</td>
</tr>
<tr>
<td>X01</td>
<td>Translate Value does not exist</td>
</tr>
<tr>
<td>X02</td>
<td>Translate Value added</td>
</tr>
</tbody>
</table>

**Image: Create Test Maintenance Report Wizard: Step 1 of 4**

This example illustrates the fields and controls on the Create Test Maintenance Report Wizard: Step 1 of 4. You can find definitions for the fields and controls later on this page.

---

**Create Test Maintenance Report**

Create a Test Maintenance Report based on compare data and test metadata.

---

**Manual Tasks**

Using App Designer, follow these tasks to create a compare report for the test maintenance process:

<table>
<thead>
<tr>
<th>Task Completed</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>Task 1: Import only the project definition for the change project.</td>
</tr>
<tr>
<td>✔️</td>
<td>Task 2: Rename the change project imported in Task 1 to create the snapshot project.</td>
</tr>
<tr>
<td>✔️</td>
<td>Task 3: Create a snapshot of the original metadata definitions by exporting the snapshot project created in Task 2 to file.</td>
</tr>
<tr>
<td>✔️</td>
<td>Task 4: Import the change project (definition and objects).</td>
</tr>
<tr>
<td>✔️</td>
<td>Task 5: Compare the snapshot project (from file) to the database, saving the compare output to tables.</td>
</tr>
</tbody>
</table>

---

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.
5. 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:

2. Highlight the snapshot project, SA_PROJ_SNAP, and click the Select button. The Replace existing SA_PROJ_SNAP? dialog box appears.

3. Click Yes. The Compare and Report From File dialog box appears.

4. Click Options to access the Upgrade Options dialog box.

5. Select the Report Options tab.

6. Select the Generate Output to Tables check box.

7. Select the Report Filter tab.

8. Click Select All.

9. Click OK.

10. Click Compare.

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

**Step 2 of 4: Analyze Compare Data**

This example shows Step 2 of 4:

**Image: Create Test Maintenance Report Wizard: Step 2 of 4**

This example illustrates the fields and controls on the Create Test Maintenance Report Wizard: Step 2 of 4. You can find definitions for the fields and controls later on this page.

For a given change project the tasks for Step 2 only need to be executed once. This analysis is based only on the compare data, not on the test data, so changes to tests do not affect the result. The relationship between this analysis and test data is calculated in Step 3: Generate Impact Report. You may generate multiple Impact Reports based on a single analysis as tests change.

In this step, PTF analyzes data from the compare you performed in Wizard Step 1, Task 5.

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

   This will launch the Application Engine program PSPTANALYSIS.

3. When the analysis is complete a pop-up appears. Click Next to continue.

   **Note:** A project name can appear more than once in the list because the Compare Output to Table feature stores data by both project name and compare date/time. The most recent compare will appear first in the list.

### Step 3 of 4: Select an Analyzed Project

This example shows Step 3 of 4:

**Image: Create Test Maintenance Report Wizard: Step 3 of 4**

This example illustrates the fields and controls on the Create Test Maintenance Report Wizard: Step 3 of 4. You can find definitions for the fields and controls later on this page.

**Create Test Maintenance Report**

Create a Test Maintenance Report based on compare data and test metadata.

**Select an Analyzed Project**

Select an analyzed project to report its impact on the PeopleSoft Test Framework metadata.

Use this page to select the analysis data PTF will use to generate the report.

Sets of compare data are listed by project name, the date and time the compare was generated, and the date and time the analysis was run.

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

### Step 4 of 4: Generate Report

This example shows Step 4 of 4:
Image: Create Test Maintenance Report Wizard: Step 4 of 4

This example illustrates the fields and controls on the Create Test Maintenance Report Wizard: Step 4 of 4. You can find definitions for the fields and controls later on this page.

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 through 3.

Follow these steps to generate a report:

1. Select the test or tests to use to generate the report.
   - Click Select All to select all tests.
2. Select the report format.
3. Click Generate Report to have PTF create the test maintenance report.
Running a Test Report from PTF Client

As you modify a test based on the results of a maintenance report, you can validate the test by running a test report from the PTF client. The report is generated for a single test, using analyses generated in Step 2 of the Test Maintenance Wizard.

1. In PTF Explorer, right-click on a test.
2. From the context menu, select Validate Test.
3. The Validate Test – Analysis Project list dialog box appears.
4. Select an analysis from the list and click Open.

   The test maintenance report opens in a new window in the PTF client.
Interpreting Test Maintenance Reports

A test maintenance report lists the tests that have been impacted by changes to menus, components, records, pages, and fields, and details the changes that impacted those tests. You can choose to output the report to PIA or to BI Publisher.

Image: Example of a Test Maintenance Report in BI Publisher Format

This example illustrates a test maintenance report in PI Publisher format.

<table>
<thead>
<tr>
<th>Test Name: RESERVED WORDS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2 - Medium</strong>: Component</td>
</tr>
<tr>
<td><strong>2 - Medium</strong>: Component</td>
</tr>
<tr>
<td><strong>2 - Medium</strong>: Component</td>
</tr>
<tr>
<td><strong>2 - Medium</strong>: Component</td>
</tr>
</tbody>
</table>

Image: Example of a Test Maintenance Report in PIA Format

This example illustrates 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:

**Image: Example of a Test Maintenance Report in Spreadsheet Format**

This example illustrates a test maintenance report that has been downloaded in spreadsheet format.

The following columns are on a Test Maintenance report (test data may be positioned differently depending on the report format):

<table>
<thead>
<tr>
<th><strong>Column Name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Name</td>
<td>The test that is impacted by a change.</td>
</tr>
<tr>
<td>Priority</td>
<td>A test maintenance report is sorted by test name. Within each test name grouping, the report items are sorted by priority, according to the values specified on the Define Analysis Rules page. See Defining Analysis Rules.</td>
</tr>
</tbody>
</table>
### 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.

<table>
<thead>
<tr>
<th><strong>Column Name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Category</strong></td>
<td>Which category the change belongs to, as detailed on the Define Analysis Rules page. See Defining Analysis Rules.</td>
</tr>
<tr>
<td><strong>Step Type</strong></td>
<td>The type of definition that was changed:</td>
</tr>
<tr>
<td></td>
<td>• Menu</td>
</tr>
<tr>
<td></td>
<td>• Component</td>
</tr>
<tr>
<td></td>
<td>• Page</td>
</tr>
<tr>
<td></td>
<td>• Record</td>
</tr>
<tr>
<td></td>
<td>• Field</td>
</tr>
<tr>
<td><strong>Market</strong></td>
<td>For components, the market; for instance, GBL.</td>
</tr>
<tr>
<td><strong>Command ID</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Seq. Nbr</strong></td>
<td>The sequence number for the test step reflects the relative run order position of the step within the test.</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Indicates whether, within the test, the step is active or inactive.</td>
</tr>
<tr>
<td><strong>Object Action</strong></td>
<td>Indicates whether the object was:</td>
</tr>
<tr>
<td></td>
<td>• Changed</td>
</tr>
<tr>
<td></td>
<td>• Added</td>
</tr>
<tr>
<td></td>
<td>• Deleted</td>
</tr>
<tr>
<td><strong>Pre Object Value</strong></td>
<td>The value before the object was changed.</td>
</tr>
<tr>
<td><strong>Post Object Value</strong></td>
<td>The value after the object was changed.</td>
</tr>
</tbody>
</table>
Test Coverage reports leverage PTF integration with PeopleTools to provide this information to you in a usable form.

Creating Test Coverage Reports

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

When used in conjunction with Usage Monitor, test coverage correlations can be extended to include information on 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 wizard consists of two steps:

• Step 1: Select a Project
• Step 2: Generate Report

Step 1 of 2: Select a Project

Use the Create Test Coverage Report page (PSPTSTSTCOVERAGESELS) to select the projects to include in the test coverage report.
Navigation

PeopleTools, Lifecycle Tools, Test Framework, Create Test Coverage Report

Image: Create Test Coverage Report Wizard: Step 1 of 2

This example illustrates the fields and controls on the Create Test Coverage Report Wizard: Step 1 of 2. You can find definitions for the fields and controls later on this page.

Create Test Coverage Report

Create a Test Coverage Report based on a project definition and test metadata.

Please select a Project

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Project Description</th>
<th>Last Update Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTRUCTOR</td>
<td></td>
<td>05/08/2012 9:23:22AM</td>
</tr>
<tr>
<td>PTF_ROLES</td>
<td></td>
<td>05/08/2012 8:21:41AM</td>
</tr>
<tr>
<td>USER_PRF_SO</td>
<td></td>
<td>06/07/2012 6:50:48PM</td>
</tr>
<tr>
<td>PO_PROJ_SNAP</td>
<td></td>
<td>05/07/2012 5:32:15PM</td>
</tr>
</tbody>
</table>

Select a change project to be correlated with PTF metadata and click Next.
Step 2 of 2: Generate Report

This example shows step 2 of 2:

Image: Create Test Coverage Report Wizard: Step 2 of 2

This example illustrates the fields and controls on the Create Test Coverage Report Wizard: Step 2 of 2. You can find definitions for the fields and controls later on this page.

The following fields are on the Generate Impact Report page:

Test Name
Select the tests to include in the report. You can select either all tests or a maximum of 10.

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 Using Usage Monitor Data with PTF.

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 be executed to test the change project.

For example, suppose you have 100 PTF tests in your test repository. You receive a bundle from PeopleSoft that consists of updates to ten PeopleCode objects. Based solely on the data stored by PTF, you cannot determine which PTF tests you need to run to test the bundle, because PTF does not track references to PeopleCode.

You can use Usage Monitor in conjunction with PTF to address this issue. Usage Monitor tracks references to 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 PeopleTools: Performance Monitor

Generating Usage Monitor Data

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

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

```sql
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:

```sql
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:

**Image: Example of a Test Coverage Report**

This example illustrates a test coverage report in PIA format.

![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):

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Name</td>
<td>The test that is impacted by a change.</td>
</tr>
<tr>
<td>Object</td>
<td>The type of definition that was in the change project.</td>
</tr>
<tr>
<td>Object Name</td>
<td>The primary key of the object.</td>
</tr>
<tr>
<td>Object Detail</td>
<td>The additional keys (as applicable) required to uniquely identify the object.</td>
</tr>
<tr>
<td>Test Metadata</td>
<td>If the object is referenced in a PTF test, this value will be <strong>Active</strong> or <strong>Inactive</strong>, reflecting whether within the test, the referenced step is active or inactive.</td>
</tr>
<tr>
<td></td>
<td>If the object is not referenced in a PTF test, this value will be <strong>No Data</strong>. This scenario can occur when the Include Usage Monitor Data checkbox is selected and the value in the Usage Monitor Data column is <strong>Yes</strong>.</td>
</tr>
<tr>
<td>Usage Monitor Data</td>
<td>When the Include Usage Monitor Data checkbox is selected this value will be <strong>Yes</strong> or <strong>No</strong>. When the Include Usage Monitor Data checkbox is deselected the value will be set to <strong>No Data</strong>.</td>
</tr>
</tbody>
</table>

**Running Test Details Reports**

A test details report is an HTML file with details for a PTF test and its associated test cases, including comments in rich text format with images.
Use the Create Test Details Report page (PSPTTSTDTLRUNCNTL) to run a test details report.

**Navigation**

PeopleTools, Lifecycle Tools, Test Framework, Create Test Details Report

**Image: Create Test Details Report Page**

This example illustrates the fields and controls on the Create Test Details Report page.

**Create Test Details Report**

Create a Test Details Report based on test metadata.

**Search For Tests By** Name: po

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PO_ITEM8</td>
<td>Test for PO Items</td>
</tr>
<tr>
<td>PO_PURCH_ORDER</td>
<td>Test Purchase Order fields</td>
</tr>
<tr>
<td>PO_TRANSPORTATION</td>
<td>Transportation vendors</td>
</tr>
<tr>
<td>PO_VENDORS</td>
<td>Test vendors component</td>
</tr>
</tbody>
</table>

To run the Test Detail report:

1. Access the Create Details Report page (PeopleTools, Lifecycle Tools, Test Framework, Test Details Report)
2. Search for tests by name or description.
3. Select up to 10 tests to include in the report.

**Note:** At least one test must be selected.

Note: The run control is automatically generated by the system and the report is scheduled. You can use the Process Monitor link to check the report status and the Report Manager link to view the report.

Image: Example Test Details Report

This example illustrates a test details report.

PeopleSoft Test Framework

Test Details Report

<table>
<thead>
<tr>
<th>Test Name:</th>
<th>PB_USER_PROFILE_SHORT</th>
<th>Type:</th>
<th>Test</th>
<th>Message Recognition:</th>
<th>Y</th>
</tr>
</thead>
</table>

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

Test Case Information

<table>
<thead>
<tr>
<th>Test Case Name:</th>
<th>CASE_01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comments:</td>
<td></td>
</tr>
</tbody>
</table>

Test Steps

<table>
<thead>
<tr>
<th>Active</th>
<th>Page Prompt</th>
<th>MAINTAIN_SECURITY.USER_SAVEAS.GBL</th>
<th>Command Id:</th>
<th>Language: ENG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Object Properties</td>
<td>Menu: MAINTAIN_SECURITY</td>
<td>Components: USER_SAVEAS.GBL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Page: (Search)</td>
<td>PageField:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Record:</td>
<td>Field:</td>
<td></td>
</tr>
</tbody>
</table>

Comments:
The step uses the Page Prompt construct instead of explicit navigation.

Test Case Details:

<table>
<thead>
<tr>
<th>CASE_01</th>
<th>Value: update</th>
</tr>
</thead>
</table>

Creating a Test Compare Report

Use the Create Test Compare Report page (PSPTTSTCOMPARESEL) to run a test compare report based on a project file.

See Creating Test Maintenance Reports
Navigation

PeopleTools, Lifecycle Tools, Test Framework, Create Test Compare Report

**Image: Create Test Compare Report Page 1 of 2**

This example illustrates the fields and controls on the Create Test Compare Report Page 1 of 2. You can find definitions for the fields and controls later on this page.

![Create Test Compare Report](image-url)
Select the project you want to compare and click Next.

**Image: Create Test Compare Report Page 2 of 2**

This example illustrates the fields and controls on the Create Test Compare Report Page 2 of 2. You can find definitions for the fields and controls later on this page.

```
Create Test Compare Report
Create a Test Compare Report based on output data from previous project compares.

Project Name: COMPARE_TESTS
Run Date/Time: 07/22/11 10:20:31PM

Generate report for either all Tests, or only select a maximum of 10 Tests

Select Test(s) to Create Compare Report

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Find</th>
<th>View All</th>
<th>First</th>
<th>Last</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑ INSTRUCTORS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>☑ STUDENTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Select All  Clear All

Report Format

- [ ] PIA
- [ ] Excel

Generate Report
```
Select the tests and report format, then Generate the report.

**Image: Example of a test compare report in PIA format**

This example illustrates a test compare report in PIA format.

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Test Case Name</th>
<th>Command ID</th>
<th>Attribute</th>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>STUDENTS</td>
<td></td>
<td>4</td>
<td>LangCd</td>
<td>EN</td>
<td>EN</td>
</tr>
<tr>
<td>STUDENTS</td>
<td></td>
<td>1</td>
<td>UserEnter</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>STUDENTS</td>
<td></td>
<td>1</td>
<td>LangCd</td>
<td>EN</td>
<td>ENG</td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>22</td>
<td>LangCd</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>12</td>
<td>LangCd</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>14</td>
<td>LangCd</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>16</td>
<td>LangCd</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>29</td>
<td>LangCd</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>18</td>
<td>LangCd</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>26</td>
<td>Component</td>
<td>PSU_INSTR</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>21</td>
<td>LangCd</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>23</td>
<td>LangCd</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>24</td>
<td>CmdType</td>
<td></td>
<td>Test</td>
</tr>
</tbody>
</table>

**Image: Example of a test compare report in Excel format**

This example illustrates a test compare report in Excel format.

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Test Case Name</th>
<th>Command ID</th>
<th>Attribute</th>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>1</td>
<td>UserEnter</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>12</td>
<td>LangCd</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>14</td>
<td>LangCd</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>16</td>
<td>LangCd</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>18</td>
<td>LangCd</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>20</td>
<td>LangCd</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>21</td>
<td>LangCd</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>22</td>
<td>LangCd</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>23</td>
<td>LangCd</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>INSTRUCTORS</td>
<td></td>
<td>24</td>
<td>CmdType</td>
<td></td>
<td>Test</td>
</tr>
</tbody>
</table>

**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.
• PSPTSTCOVRGRPT

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

Related Links
Recording Tests

Document Tests

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

You can find description fields in:

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

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

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

Clean Up Tests

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

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

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

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

**Image: Example of using Browser.Start_Login**

This example illustrates 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:

<table>
<thead>
<tr>
<th>No.</th>
<th>ID</th>
<th>Comment</th>
<th>Active</th>
<th>Special ID</th>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>Browser</td>
<td>Start_Login</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>Link</td>
<td>Click</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>Text</td>
<td>Set_Value</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>Push</td>
<td>Set_Value</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>Button</td>
<td>Click</td>
<td>✔️</td>
<td></td>
</tr>
</tbody>
</table>

Related Links

Configuring Execution Options

Start Login

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 administrator on the Define Configuration Options page and can be overridden during recording using the recorder utility tools.

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

**Related Links**

- Prompt
- PromptOK
- Using the PTF Test Recorder
- Defining Configuration Options

### Use the Process Step Type

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

When you select the *Use Process Run* check box before you record a process request all actions in the run request standard page will be replace by the step action Process.Run and any Process Monitor actions that are recorded will be ignored.

You can set the *Use Process Run* check box using PIA (select PeopleTools> Lifecycle Tools > Test Framework > Define Configuration Options) for all tests, or you can select the check box from the Toolbar when you are ready to record a test (Configure Recording Settings icon).

The Process.Run step will handle the execution and return the process status. You can modify the parameters on the Process.Run step to indicate the expected results.

Suppose you have the following test scenario:

> From the Request Calculation page, click the Process Monitor link.  
> In Process Monitor, click the Refresh button until either Success or Error appears in the Run Status column for your process instance.

The Process.Run step will capture all of the process information and you can modify the step parameters in PTF.

**Related Links**

- Process

### Make Tests Dynamic

Many tests involve values and conditions that are not known until a test runs. Advanced functionality in PTF enables you to build tests that adapt to the application when necessary.

These 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.

Related Links
Using Variables
Using Conditional Logic
Incorporating Scroll Handling
Using Reserved Words

Reduce Duplication

Avoid creating tests and test steps that are duplicated elsewhere. When multiple tests validate similar functionality, it increases the complexity of test maintenance and the amount of manual work necessary to add or change test functionality later.

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.

Related Links
Creating Test Cases
Calling Tests
Chapter 9

Using the PTF Test Language

This topic 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.

Image: Example of a PTF test

This example illustrates the fields and controls on for a simple PTF test. You can find definitions for the fields and controls later on this page.

<table>
<thead>
<tr>
<th>Seq</th>
<th>ID</th>
<th>Comment</th>
<th>Active</th>
<th>Scroll ID</th>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Field Label</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 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:

**Image: Example of a PTF test step**

This example illustrates the step to enter a value in an edit box.

<table>
<thead>
<tr>
<th>Seq</th>
<th>ID</th>
<th>Comment</th>
<th>Active</th>
<th>Scroll ID</th>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>✔️</td>
<td>✔️</td>
<td>Test</td>
<td>Text</td>
<td>Set_Value</td>
<td>Name-EMPLID</td>
<td>KU0001</td>
</tr>
</tbody>
</table>

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 topic, “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
• Syntax Check
• Context sensitive help within grid for function parameter details

Validation

Each step type allows only certain actions. Similarly, each action can only be used with certain step types. The PTF Test Editor automatically limits your choice of actions based on the step type selected.

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

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

**Image: Action Drop-Down List for Text**

This example illustrates the drop-down list showing available actions for the step type:Text.

<table>
<thead>
<tr>
<th>Text</th>
<th>Set_Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link</td>
<td>Exists</td>
</tr>
<tr>
<td></td>
<td>GetLabel</td>
</tr>
<tr>
<td></td>
<td>Get_Property</td>
</tr>
<tr>
<td></td>
<td>Set_Value</td>
</tr>
<tr>
<td></td>
<td>Verify</td>
</tr>
</tbody>
</table>

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 Radio.

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

For example:

```
ret=&chart_val;
```

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

The PTF Test Editor provides context sensitive help for parameters. Double-click in the Recognition column to display and enter the parameters. For each parameter, an explanation and example are displayed.

For examples see Context Sensitive Help within Grid for Function Parameter Details

Variables

Variables enable you to work with steps in which the information or values are not known when you create the test or the information or values for a step change each time the test executes.

You prefix variables with an ampersand (&).

In this example, the first step stores the value in the OPRID field to the variable &OPRID. The second step populates the field OPRDEFNDESC with the value in the variable &OPRID.

Image: Example of Using Variables in Test Steps

This example illustrates how variables can be used in test steps.

<table>
<thead>
<tr>
<th>Text</th>
<th>Get_Property</th>
<th>Name=OPRID,ret=&amp;OPRID</th>
</tr>
</thead>
</table>

| Text | Set_Value | Name=OPRDEFNDESC | &OPRID |

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

Related Links

Variable
Using Variables
System Variables

Reserved Words

Similar to variables, you use reserved words to supply information that is not known until a test executes.

Prefix reserved words with a pound sign (#) and use them in the Value field of a test to verify a condition, change the value in an application field, or both. In this example, the #TODAY reserved word sets the EFFDT_FROM field to the current date.

Image: Example of using the #TODAY reserved word

This example illustrates how the reserved word #TODAY can be used in a test step.

| Text | Set_Value | Name=EFFDT_FROM | #TODAY |

Related Links

Using Reserved Words
Reserved Words
System Variables

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

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

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

**Image: Example of using system variables**

This example illustrates 2 methods for assigning system variables to user-defined variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Set_Value</th>
<th>&amp;User</th>
<th>%oo.dbuser%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Set_Value</td>
<td>&amp;Test=%test%</td>
<td></td>
</tr>
</tbody>
</table>

Related Links

System Variables

Check Syntax

If the syntax or value in a test step is not correct, the test will produce an error when the test is executed. The Check Syntax option allows a user to validate the parameters provided in their steps either on save or on demand prior to execution.

Check Syntax will:

- Display an error message for any invalid or missing required parameter values.
- Display a error message for any invalid optional parameter values.
- Display a warning message when the return value of a step has not been entered.
- Display an error window listing all errors and warnings found in sequential order for a test.

**Note:** There may be multiple errors in a single step.

- Display an error when syntax is incorrect or missing, such as verifying that all Conditional.If steps are paired with Conditional.End_if steps.

**Note:** Syntax check only validates active steps. Only syntax is validated, not data.

Check Syntax does not validate:

- PTF Functions
- System variables
- Reserved words
- Condition sign
• File and folder names
• Duplicated parameters
• Missing values
• Parameter that is not available for the action
• Missing parameter name

**Running On Demand Syntax Check**

After you have edited your test, select Test, Check Syntax or click the Check Syntax icon. The errors and warnings will be displayed. When you click on the error or warning in the Test Check Syntax window, the step is highlighted in the test.

**Image: Test Syntax Check Showing 2 Errors**

This example illustrates the check syntax dialog box with 2 errors.
If the syntax check does not detect any errors or warnings, the dialog box will not return any rows of data:

**Image: Test Syntax Check Showing No Errors or Warnings**

This example illustrates the check syntax dialog box with no errors.

**Automatically Running Syntax Check on Save**

To have Syntax Check run automatically every time you save a test:

1. In PTF Explorer, click on your environment name and select Local Options.
2. In the Auto-Check Syntax field select Yes and click OK.
3. When you save a test, you will be prompted to run the syntax check.
Context Sensitive Help within Grid for Function Parameter Details

For any step type/action including parameters, you can double-click in the Recognition column to display and enter the parameters. For each parameter, an explanation and example are displayed. This is an example of the step Process.Run:

**Image: Parameter Dialog Box for Step:Process.Run**

This example illustrates the fields and controls on the Parameter dialog box for Step:Process.Run. You can find definitions for the fields and controls later on this page.

![Parameter Dialog Box for Step:Process.Run](image)

For step type Query, you can also double-click in the Value column to display and enter Query parameters.
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 substitute ## for the number, for example ptModFrame_## and PTF will search through all frames starting with ptModFrame_ and use the one with the highest number.
Set_URL

Description
Sets the portal type. Uses the URL for the selected portal type defined in the execution option to access the component. This is the url that is used with Page.Prompt.

Example

Image: Example of Set_URL action

This example illustrates the Set_URL action.

<table>
<thead>
<tr>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser</td>
<td>Start_Login</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browser</td>
<td>Set_URL</td>
<td>EMPLOYEE</td>
<td></td>
</tr>
</tbody>
</table>

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 Click.

Exists

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

See Exists.

Get_Property

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

See Get_Property.

---

Chart

These are the actions associated with the Chart step type.

ChartClick

**Description**
Performs a click in a chart section.

**Parameters**

- `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.

GetChartType

**Description**
Returns the chart type chart type from a displayed chart on the PIA page.
Parameters

chart=value;  The index for the chart image on the page.
ret=&variable;  The return value.

GetText

Description
Returns the text value for the specified chart section.

Parameters

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 step type:

Image: Example of GetText action for a Chart step type

This example illustrates the GetText action for a Chart step type.

<table>
<thead>
<tr>
<th>Chart</th>
<th>GetText</th>
<th>chart=0,ret=&amp;chart_val</th>
<th>idx=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log</td>
<td>Message</td>
<td>The value for index 3 is &amp;chart_val;</td>
<td></td>
</tr>
<tr>
<td>Chart</td>
<td>GetText</td>
<td>chart=0,ret=&amp;chart_val</td>
<td>idx=2;</td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>The value for index 2 is &amp;chart_val;</td>
<td></td>
</tr>
</tbody>
</table>

SectionCount

Description
Counts the number of regions in a chart.

Parameters

chart=value  The index for the chart image on the page.
ret=&variable  The return value.

The Chart_SECTIONCount returns the real value (not zero-based). If you are using this variable in a For Loop, you will need to subtract 1.
Example

This is an example of the SectionCount action for a Chart step type:

Image: Example of SectionCount Action for a Chart Step Type

This example illustrates using the SectionCount action with a loop to retrieve the text for each section.

<table>
<thead>
<tr>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser</td>
<td>Start_Login</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browser</td>
<td>FrameSet</td>
<td>TargetContent</td>
<td></td>
</tr>
<tr>
<td>Page</td>
<td>Prompt</td>
<td>QE_CHART_MENU QE_CHART2.GBL update</td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>Set_Value</td>
<td>Name=QE_CHART_RECORD QE_CHART_CATEGORIES</td>
<td>CAR SALES</td>
</tr>
<tr>
<td>Page</td>
<td>PromptOk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chart</td>
<td>SectionCount</td>
<td>chart+0=+chart_count</td>
<td></td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>ChartCount =&gt; &amp;chart_count</td>
<td></td>
</tr>
<tr>
<td>Loop</td>
<td>For</td>
<td>$view=0 to Subtotal[&amp;chart_count]</td>
<td></td>
</tr>
<tr>
<td>Chart</td>
<td>GetText</td>
<td>chart$+0=+chart_value</td>
<td></td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>ChartValue = &amp;chart_value</td>
<td></td>
</tr>
<tr>
<td>Loop</td>
<td>Next</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 Exists.

GetLabel

Description

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

See GetLabel.

Get_Property

Description

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

See Get_Property.
Set_Value

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

Verify

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

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 Exists.

Get_Property

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

GetLabel

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

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.

**Image: Example of using the Set_Value action with the #LIST# reserved word**

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

| ComboBox | Set_Value | Name=PS0PR0DFN_LANGUAGE_CD | #LIST#English|[French][Finnish] |

**Related Links**

#LIST#

**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:

**Image: Example of Verify action**

This example illustrates the step to verify that the field value is set to Two and that the values One and Three exist.

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

**Command**

Use the Command step to run a command line program. The command step has one associated action Exec.
Exec

Description
Executes a command line program.

Parameters

name=&commandname
The command name as defined on the Define Configuration Options page.
See Defining Configuration Options.

timeout=&seconds
The number of seconds before the program times out.

ret=&variable
The return value.

Example

This is an example of a command line program that requires multiple parameters.

Image: Example of Command.Exec with Parameters Defined in the Value Column
This example illustrates a command line program that requires multiple parameters.

Example

If the command line program is used in multiple tests, you should create a library test for the command. This example shows a test library created for the FILEKILL command. The command has one parameter FILENAME.

Note: FILEKILL and FILECHECK commands are examples of possible utilities to call from PTF, they are not delivered with PTF. Customers are responsible to create their own command line programs.

Image: Example Command in a Library Test
This example illustrates the FILEKILL command.exec step, it has one parameter FILENAME. This command will look for a specific file and delete it if it exists. If the file cannot be deleted it returns a 1.
This is an example of the calling test.

**Image: Example of the Calling Test**

This example illustrates how the command can be called from a test. This test will delete the file before running a process to create a new file.

<table>
<thead>
<tr>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser</td>
<td>Start_Login</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Val_Value</td>
<td>Filename</td>
<td>c:\Lab\MyForm.pdf</td>
</tr>
<tr>
<td>Test</td>
<td>Exec</td>
<td>FILEKILL_LIB\FILENAME = filename</td>
<td>DEFAULT</td>
</tr>
<tr>
<td>Browser</td>
<td>FrameSet</td>
<td>TargetContent</td>
<td></td>
</tr>
<tr>
<td>Page</td>
<td>Prompt</td>
<td>PROCESS_SCHEDULER\PRCSMULTI.GBL</td>
<td>edit update</td>
</tr>
<tr>
<td>Test</td>
<td>Set_Value</td>
<td>Name=PRCSMULTI\RUN\CNTL_ID</td>
<td>test</td>
</tr>
<tr>
<td>Page</td>
<td>PromptOk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process</td>
<td>Run</td>
<td>process=40;WIN\pretype=II Publisher\autotype=File\outformat=PDF\outfile=filename</td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>Exec</td>
<td>FILECHECK_LIB\FILENAME = filename</td>
<td>DEFAULT</td>
</tr>
</tbody>
</table>

---

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

If_Then supports these logical operators:

\(<\), \(\leq\), \(\geq\), \(=\), \(<\), \(\neq\)

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:

**Image: Example of the If_Then conditional construct**

This example illustrates the use of multiple conditions and nested conditionals.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Set_Value</th>
<th>&amp;Integer</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditional</td>
<td>If_Then</td>
<td>&amp;Integer&gt;=1 AND &amp;Integer&lt;=10 OR &amp;Integer=0</td>
<td></td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>Determine whether Integer is odd or even</td>
<td></td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>Enter first nested conditional</td>
<td></td>
</tr>
<tr>
<td>Conditional</td>
<td>If_Then</td>
<td>&amp;Integer = 0</td>
<td></td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>Do not divide by 0.</td>
<td></td>
</tr>
<tr>
<td>Conditional</td>
<td>Else</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Set_Value</td>
<td>&amp;Half=Divide(&amp;Integer2[dec=1])</td>
<td></td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>Enter second nested condiional</td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Set_Value</td>
<td>&amp;Odd=Substr(&amp;Half[3])</td>
<td></td>
</tr>
<tr>
<td>Conditional</td>
<td>If_Then</td>
<td>&amp;Odd = 5</td>
<td></td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>The number is odd</td>
<td></td>
</tr>
<tr>
<td>Conditional</td>
<td>Else</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>The number is even</td>
<td></td>
</tr>
<tr>
<td>Conditional</td>
<td>End_If</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conditional</td>
<td>End_If</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conditional</td>
<td>End_If</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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:

**Image: Example of the DataMover Exec action**

This example illustrates the use of the Data Mover.Exec action

| DataMover | Exec | C: \temp\emp_imp.dns |

---

**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 Click.

---

**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. Specify False to execute the Process.Run steps.

This action overrides the Skip RunRequest setting in Execution Options.

Skip_Login

**Description**

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

StopOnError

**Description**

Specify *True*, *False* or *ALL* in the Recognition column.

Specify True to stop execution on the current called Test when a Fail is logged.

Specify False to continue execution when the called test encounters a Fail.

Specify ALL to stop shelltest execution when any called test encounters a Fail.

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

**Related Links**

Configuring Execution Options

---

**File**

The File step type corresponds to the object in PIA that allows users to upload and download files to the PeopleSoft application.

These are the actions associated with the File step type.

**Download**

**Description**

Used to download a file.

The File.Download step needs to be preceded by the appropriate click (such as Image.Click, Link.Click, or Button.Click).Click. In the Value column specify a full file path name.
Select the File Download Prompt check box in the Settings dialog box to be prompted for a file download path when recording the test.

To set the file download prompt check box:
1. Click the Show Test Recorder icon.
2. Click the Configure recording settings icon.
3. Select File Download Prompt check box and click OK.
4. Record your test and you will be prompted for the file download path when you record a download step.

**Example**

These examples show the use of File.Download.

**Image: Example of File Download action for downloading to Excel**

This example illustrates the File Download action for downloading a Query to Excel.

<table>
<thead>
<tr>
<th>Link</th>
<th>Click:</th>
<th>Name:</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Download</td>
<td>C:\temp\excel_test.xlsx</td>
<td></td>
</tr>
</tbody>
</table>

**Image: Example of File Download action from a button**

This example illustrates the File Download action from a button.

<table>
<thead>
<tr>
<th>Button</th>
<th>Click:</th>
<th>Name:</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Download</td>
<td>C:\temp\excel_test.xlsx</td>
<td></td>
</tr>
</tbody>
</table>

**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 path name.

**Example**

This example shows the use of the Upload action:

**Image: Example of the File Upload action**

This example illustrates the File.Upload step.

<table>
<thead>
<tr>
<th>File</th>
<th>Upload</th>
<th>name:</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#OrigFileName</td>
<td>C:\Temp\MyFile.txt</td>
</tr>
</tbody>
</table>
HTMLTable

These are the actions associated with the HTMLTable step type.

CellClick

**Description**

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

**Parameters**

```
index=I/R/C:  The table, row, column index string.
```

For example:

```
index=&CellIndex
index=1/2/3;
```

In the second example, CellClick clicks on the third column of the second row of the first table.

CellClickOnChkB

**Description**

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

**Parameters**

```
index=I/R/C:  The table, row, column index string.
```

For example:

```
index=&CellIndex
index=1/2/3;
```

In the second example, this function clicks on a checkbox within the third column of the second row of the first table.

```
chkidx=value;  The CheckBox object index inside the cell.
check=value;   check=Y – Select the checkbox.
               check=N – Clear the checkbox.
```

This parameter is optional. The default value is Y.
CellClickOnImage

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

Parameters
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
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
index=I/R/C: The table, row, column index string.
For example:

\[ \text{index} = \&\text{CellIndex} \]
\[ \text{index} = 1/2/3; \]

In the second example, this function verifies whether a cell exists within the third column of the second row of the first table.

\[ \text{ret} = \&\text{variable} \]

The return value.

True – the cell exists.

False – the cell does not exist.

**CellGetIndex**

**Description**

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

Use the GetCellIndex button on the recorder toolbar to capture the text value and return a variable for the index.

**Parameters**

| text=value; | The text to look for on the page. |
| index=value; | Optional. If a value is entered it is used to start the search for the text. If the value blank, PTF will start to look for the text in the index 1/1/1. |
| equal=value; | equal=true performs an exact match on the text to search for. This is the default for this optional parameter. |
| equal=false uses a LIKE statement when performing the search. |
| 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:

\[ \text{ret} = \&\text{CellIndex} \]
Example

This is an example of several of the HTMLTable actions:

Image: Example of CellGetIndex action

This example illustrates using the CellGetIndex to return a variable for the index, then create a variable for the html cell that is to the left of the original cell using the sum function. In this example PTF will start to look for the text in table 7, row 1, column 1.

<table>
<thead>
<tr>
<th>Browser</th>
<th>Start_Login</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser</td>
<td>FrameSet</td>
<td></td>
</tr>
<tr>
<td>Link</td>
<td>Click</td>
<td>id=pitacpercent</td>
</tr>
<tr>
<td>Browser</td>
<td>FrameSet</td>
<td>plModFrame_###</td>
</tr>
<tr>
<td>HTMLTable</td>
<td>CellGetIndex</td>
<td>teste</td>
</tr>
<tr>
<td>Variable</td>
<td>Set_Value</td>
<td>index=index</td>
</tr>
<tr>
<td>HTMLTable</td>
<td>CellClick</td>
<td>index=lHtmlIndex</td>
</tr>
<tr>
<td>Page</td>
<td>Server</td>
<td>Name=PORTAL_HFWPRK_HTMLAREA</td>
</tr>
</tbody>
</table>

CellGetValue

Description

Returns the contents of an HTMLTableCell.

Use the CellGetValue button on the recorder toolbar to capture the index value and return a variable for the value.

Parameters

index=I/R/C: The table, row, column index string.

For example:

index=&CellIndex

index=1/2/3;

In the second example, this function returns the contents of the third column of the second row of the first table.

ret=&variable 

The return value.

ColCount

Description

Returns the number of columns for the HTMLTable row.
Parameters

\[ \text{table}=\text{value}; \]  
The table index.

\[ \text{row}=\text{value}; \]  
The row index.

\[ \text{index}=\text{I/R/C}; \]  
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:

\[ \text{index}=\&\text{CellIndex}; \]

\[ \text{ret}=\&\text{variable} \]  
The return value.

RowCount

Description

Returns the number of rows for the HTMLTable.

Parameters

\[ \text{table}=\text{value}; \]  
The table index.

\[ \text{index}=\text{I/R/C}; \]  
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:

\[ \text{index}=\&\text{CellIndex}; \]

\[ \text{ret}=\&\text{variable} \]  
The return value.

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 Click.

**Exists**

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

See Exists.

**Get_Property**

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

See Get_Property.

**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 Click.

**Exists**

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

See Exists.
Get_Property

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

SaveTargetAs

Description
Provides the ability to do a right-click and then save as on a link. Enter the link name in the recognition column and the file path in the value column.

Example

Image: Example of SaveTargetAs action
This example illustrates saving a report to a target location. From the View Log/Trace page (PMN_CDM_INDEX) in the Process Monitor, click on the report name, verify it is the correct report and then do a SaveTargetAs to save the report to a file location.

<table>
<thead>
<tr>
<th>Link</th>
<th>Click</th>
<th>Name=PMN_DERIVED_INDEX_BTN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link</td>
<td>Verify</td>
<td>Name=URL10</td>
</tr>
<tr>
<td>Link</td>
<td>SaveTargetAs</td>
<td>Name=URL10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C:TEMP\DDAUIDT PDF</td>
</tr>
</tbody>
</table>

Verify

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

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:

**Image: Example of Log action steps**

This example illustrates Log actions for fail, Pass and Warning.
This log example shows how text from the Log actions appears in the Log Viewer:

**Image: Example of a PTF log**

This example illustrates the PTF log generated when the above test is executed.

---

**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 **Exists**.

**Get_Property**

**Description**

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

See **Get_Property**.
GetLabel

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

See GetLabel.

Set_Value

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

See Set_Value.

SetValue_InModal

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

Verify

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

See Verify.

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

Image: Example of a Loop.Do construct

This example illustrates a Loop.Do construct.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Set_Value</th>
<th>&amp;Var = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop</td>
<td>Do</td>
<td></td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>The variable is &amp;Var</td>
</tr>
<tr>
<td>Conditional</td>
<td>If_Then</td>
<td>&amp;Var &gt; 3</td>
</tr>
<tr>
<td>Loop</td>
<td>Exit_Loop</td>
<td></td>
</tr>
<tr>
<td>Conditional</td>
<td>End_If</td>
<td></td>
</tr>
<tr>
<td>Loop</td>
<td>End_Loop</td>
<td></td>
</tr>
</tbody>
</table>

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

Image: Example of Loop.While

This example illustrates a Loop.While construct.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Set_Value</th>
<th>&amp;Var = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop</td>
<td>While</td>
<td>&amp;Var &lt;=3</td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>The variable is &amp;Var</td>
</tr>
<tr>
<td>Variable</td>
<td>Set_Value</td>
<td>&amp;Var = add(&amp;Var,1)</td>
</tr>
<tr>
<td>Loop</td>
<td>End_Loop</td>
<td></td>
</tr>
</tbody>
</table>
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

&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 Add.

Example

Image: Example of a Loop.For Construct

This example illustrates using the Loop For with Loop Next to terminate the loop.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Set_Value</th>
<th>&amp;Var = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop</td>
<td>For</td>
<td>&amp;Var = 1 to 5</td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>The variable is &amp;Var</td>
</tr>
<tr>
<td>Loop</td>
<td>Next</td>
<td></td>
</tr>
</tbody>
</table>

MsgBox

Use the MsgBox step to insert manual steps in your test for steps that cannot be automated in PTF. For example, if your test includes a step to drag and drop on a page, you will not be able to automate the drag and drop feature. By using the MsgBox step, the message will be displayed allowing the user to perform the action manually, once the user dismisses the message box PTF execution will continue.

All of the actions associated with MsgBox step type use the same parameters.
Actions for MsgBox Step Type

The following actions are associated with the MsgBox step type, all of the actions use the same parameters.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbortIgnoreRetry</td>
<td>Creates a message box that contains 3 buttons- Abort, Ignore, and Retry.</td>
</tr>
<tr>
<td>OKCancel</td>
<td>Creates a message box that contains 2 buttons- OK and Cancel.</td>
</tr>
<tr>
<td>OKOnly</td>
<td>Creates a message box that contains 1 button- OK.</td>
</tr>
<tr>
<td>RetryCancel</td>
<td>Creates a message box that contains 2 buttons- Retry and Cancel.</td>
</tr>
<tr>
<td>YesNo</td>
<td>Creates a message box that contains 2 buttons- Yes and No.</td>
</tr>
<tr>
<td>YesNoCancel</td>
<td>Creates a message box that contains 3 buttons- Yes, No, and Cancel.</td>
</tr>
</tbody>
</table>

Parameters

**prompt=value**

The text string displayed in the message dialog box.

**title=value**

The text string splayed in the title bar of the message dialog box.

**ret=&MyBtnClick**

The variable name that will store the integer indicating which button the user clicked.

Return Values

The return value is based on the button that was pressed by the user during test execution:

<table>
<thead>
<tr>
<th>Button Pressed</th>
<th>Return Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>1</td>
</tr>
<tr>
<td>Cancel</td>
<td>2</td>
</tr>
<tr>
<td>Abort</td>
<td>3</td>
</tr>
<tr>
<td>Retry</td>
<td>4</td>
</tr>
<tr>
<td>Ignore</td>
<td>5</td>
</tr>
<tr>
<td>Yes</td>
<td>6</td>
</tr>
<tr>
<td>Button Pressed</td>
<td>Return Value</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
</tr>
<tr>
<td>No</td>
<td>7</td>
</tr>
</tbody>
</table>

**Example**

**Image: Example of MsgBox Steps**

In this example variables are created to store the text for the prompt and title. The message box is created as a YesNo message box. If the Yes button is pressed by the user, the return value will be 6.

<table>
<thead>
<tr>
<th>Variable</th>
<th>$Sel_Value</th>
<th>$Prompt</th>
<th>Manually drag and drop</th>
</tr>
</thead>
<tbody>
<tr>
<td>MsgBox</td>
<td>YesNo</td>
<td>Prompt</td>
<td>Was manual test step successful?</td>
</tr>
<tr>
<td>Conditional</td>
<td>If_Then</td>
<td>&amp;=6</td>
<td></td>
</tr>
<tr>
<td>Log</td>
<td>Pass</td>
<td>Manual Step &amp; TestStep Passed</td>
<td></td>
</tr>
<tr>
<td>Conditional</td>
<td>Else</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>Manual Step &amp; TestStep Failed</td>
<td></td>
</tr>
<tr>
<td>Conditional</td>
<td>End_II</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Image: Example of the Message Box displayed in test execution**

Based on the test steps in the example above, the message box is created.

**MultiSelect**

MultiSelect allows you to select multiple value. The values are separated by a pipe (|). These are the actions associated with the MultiSelect step type.

**Exists**

**Description**

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

See Exists.

**Get_Property**

**Description**

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

GetLabel

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

See GetLabel.

Set_Value

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

See Set_Value.

Verify

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

See Verify.

MultiSelect Example

This is an example of using MultiSelect on the Portal Layout page. In this example, you select 3 values in the first column (col0) and move them to the second column (col1), then verify the values.

**Image: Example of MultiSelect Steps**

This example illustrates using MultiSelect with Set_Value to indicate the multiple values and then move the values to a new column, the verify the values in the new column.

<table>
<thead>
<tr>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser</td>
<td>Start_Login</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browser</td>
<td>FrameSet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Link</td>
<td>Click</td>
<td>d-probeclayout</td>
<td></td>
</tr>
<tr>
<td>Browser</td>
<td>FrameSet</td>
<td>pbf/probe_39</td>
<td></td>
</tr>
<tr>
<td>MultiSelect</td>
<td>Set_Value</td>
<td>Name=col0</td>
<td>Calculation/Calendar/Dictionary</td>
</tr>
<tr>
<td>Image</td>
<td>Click</td>
<td>Name=naming</td>
<td></td>
</tr>
<tr>
<td>MultiSelect</td>
<td>Verify</td>
<td>Name=col1</td>
<td>Dictionary/Calendar/Calculator</td>
</tr>
<tr>
<td>Page</td>
<td>Save</td>
<td>Name=PORTAL_HPwK.HTMLAREA$7%</td>
<td></td>
</tr>
</tbody>
</table>
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
Image: Example of the Go_To action
This example illustrates the Go_To action for a page step.

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

Image: Example of the Prompt and PromptOK actions

This example illustrates the use of the Prompt and PromptOK actions.

<table>
<thead>
<tr>
<th>Page</th>
<th>Prompt</th>
<th>SQA_MENU SQA_SIMPLECOMP.GBL</th>
<th>add update</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Set_Value</td>
<td>name=SQA_SIMPLERECD_SQA_DATAID</td>
<td>US001</td>
</tr>
<tr>
<td>Page</td>
<td>PromptOk</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Save

Description

Attempts to save the current page. This action checks for the SkipSavePage flag in the execution options. For non-standard pages, the non-standard save object is recorded in the recognition column.

Example

Image: Non-Standard Page Save Step

This example illustrates a non-standard page save where the non-standard save object is defined in the recognition field.

| Page | Save | Name=PORTAL_HTMLAREA |

SecPage_Close

Description

Closes the secondary page.

SecPage_Open

Description

Opens a secondary page.

Parameters

page= value Specify the name of the secondary page.

Example

Image: Example SecPage_Open

This example illustrates the SecPage_Open action.

| Page | SecPage_Open | Name=SQA_SIMPLEREC; page=SQA_SIMPLE_SUBPAGE; |
Process

The Process actions run processes through Process Scheduler.

Run

Description

Runs a Process Scheduler process.

Parameters

Parameter | Description |
---|---|
prcname=\textit{value}; | The process name. |
prctype=\textit{value}; | The process type. |
wait=\textit{value}; | True - the test waits for the process to finish. |
| False - the test does not wait for the process to finish. |
outtype=\textit{value}; | The process output type. |
outformat=\textit{value}; | The process output format. |
outfile=\textit{value}; | The process output file. |
expected=\textit{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: |
| \texttt{expected=Success;} |
| \texttt{expected=No Success;} |
| If the final status equals the expected status, then a Pass is logged; if not, a Fail is logged. |
ret=&\textit{variable} | The return value. |
| True - the process completed successfully. |
| False - the process did not complete successfully. |
Example

Image: Example of Run Action

This example illustrates an example of the run action, followed by a conditional If_Then to verify that the process ran to success.

<table>
<thead>
<tr>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser</td>
<td>Start_Login</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browser</td>
<td>Frameset</td>
<td>TargetContent</td>
<td></td>
</tr>
<tr>
<td>Page</td>
<td>Prompt</td>
<td>APPMONITOR_RUN_AFNSDARCH &lt;SEL&gt;</td>
<td>add update</td>
</tr>
<tr>
<td>Test</td>
<td>Set_Value</td>
<td>Name=PROCESSRUNCTL_RUN_CNTL_ID</td>
<td>archive</td>
</tr>
<tr>
<td>Page</td>
<td>PromptOk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CheckBox</td>
<td>Set_Value</td>
<td>Name=ARCH_RUNCNTL_IB_STATUS_DCHECK</td>
<td></td>
</tr>
<tr>
<td>Process</td>
<td>Run</td>
<td>pname=AFNSDARCH片刻type=ApplicationEnginetype=Webautomatic=TXT,expected=Successset=status</td>
<td>Y</td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>status is Status</td>
<td></td>
</tr>
<tr>
<td>Conditional</td>
<td>If_Then</td>
<td>Status = True</td>
<td></td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>Method</td>
<td></td>
</tr>
<tr>
<td>Conditional</td>
<td>End_It</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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.

Related Links

Configuring Local Options
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 Exists.

GetLabel

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

Set_Value

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

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.

Note: Context sensitive help is available by double-clicking in the Value column to display and enter the query parameters.

Parameters

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.

**Exec_Private**

**Description**

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

**Parameters**

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.

**Note:** Query steps do not support simultaneous usage of both 0rows= and Nrows= parameters in the same step.

Use the following formats to specify query parameters:

<table>
<thead>
<tr>
<th><strong>Page Control</strong></th>
<th><strong>Format</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>param=value</td>
</tr>
<tr>
<td>Radio button</td>
<td>param={RB}value</td>
</tr>
</tbody>
</table>
### Page Control

<table>
<thead>
<tr>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Combo box</td>
</tr>
<tr>
<td>Check box</td>
</tr>
<tr>
<td>Text box</td>
</tr>
</tbody>
</table>

### Example

This example shows the Query.Exec step type, with the dialog box for the values open. In this example, the query has 3 prompts: a text field, a date field and a check box. The query is expected to return rows.

**Image: Example of Query.Exec Action**

This example illustrates entering the value parameters for the Query.Exec action.

---

### Radio

These are the actions associated with the Radio step type.

### Exists

**Description**

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

See \textit{Exists}.  

---

*Image: Example of Query.Exec Action*
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.

Image: Example of setting the value for a Radio step type

This example illustrates setting the value for a Radio step type.

<table>
<thead>
<tr>
<th>Radio</th>
<th>Exists</th>
<th>name=SQA_SIMPLEREC_SQAOPPTION; ret=&amp;RadioEx1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radio</td>
<td>Exists</td>
<td>name=SQA_SIMPLEREC_SQAOPPTION; ret=&amp;RadioEx2; value=3</td>
<td></td>
</tr>
</tbody>
</table>

Get_Property

Description

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

See Get_Property.

GetLabel

Description

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

See GetLabel.

Set_Value

Description

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

See Set_Value.

Verify

Description

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

See Verify.

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 GetLabel.

Set_Value

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

Scroll

These are the actions associated with the Scroll step type.
The Scroll ID field is required for all Scroll action types.

Related Links
Incorporating Scroll Handling

Action

Description
Takes an action based on the row specified by Key_Set.

Parameters

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.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ins</td>
<td>Insert a row. If the current row is the first row, then use the existing row.</td>
</tr>
<tr>
<td>ins+</td>
<td>Always insert a row.</td>
</tr>
<tr>
<td>delins</td>
<td>Delete all rows, and then insert into the first row.</td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>del</td>
<td>Delete the row specified by the Key_Set action.</td>
</tr>
<tr>
<td>delall</td>
<td>Delete all rows. No further processing.</td>
</tr>
<tr>
<td>delsel</td>
<td>Look for the row and delete it. Do not add a fail if the row does not exist.</td>
</tr>
<tr>
<td>upd</td>
<td>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.</td>
</tr>
<tr>
<td>upd+</td>
<td>Find a specified row to work with. If is not found, always insert a new row.</td>
</tr>
<tr>
<td>sel</td>
<td>Find a row specified by the Key_Set action.</td>
</tr>
<tr>
<td>find</td>
<td>Use the scroll Find link. Format: find=\textit{text_to_find}</td>
</tr>
<tr>
<td>not</td>
<td>Check that the row is not in the scroll. Add a Fail to the log if the row is found.</td>
</tr>
</tbody>
</table>

**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**

\textbf{def=\textit{value};}

One of the definition values. Example:

\texttt{def=parent;}

The following table gives valid values for the def parameter.

\textbf{type=\textit{value};}

The step type for the new action.

Example 1:

\texttt{type=Image;}

Example 2:

\texttt{type=PushButton;}

\textbf{value=\textit{value};}

The scroll parent variable or the new object recognition string.

Example 1:

\texttt{value=&Scr1;}
Example 2:

```
value=Name=$ICField3$newm$0$$img$0;
```

This table lists the valid values for the def parameter:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>Override the Add new row button.</td>
</tr>
<tr>
<td>del</td>
<td>Override the Delete row button.</td>
</tr>
<tr>
<td>next</td>
<td>Override the Next button.</td>
</tr>
<tr>
<td>prev</td>
<td>Override the Previous button.</td>
</tr>
<tr>
<td>first</td>
<td>Override the First button.</td>
</tr>
<tr>
<td>last</td>
<td>Override the Last button.</td>
</tr>
<tr>
<td>parent</td>
<td>Reassign the parent for a specific scroll area.</td>
</tr>
</tbody>
</table>

**Examples**

This example uses the Definition action with the `def=Add` parameter to assign the Add action to the image $ICField3$newm$0$$img$0:

**Image: Example of a Definition action with def=Add parameter**

This example illustrates using the Definition action with def=Add parameter.

<table>
<thead>
<tr>
<th>1</th>
<th>Scroll</th>
<th>Definition</th>
<th>def=Add; Type=Image; Value=$ICField3$newm$0$$img$0</th>
</tr>
</thead>
</table>

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

**Image: Example of the Definition action with the def=parent parameter**

This example illustrates using the Definition action with the def=parent parameter.

```
1 | Scroll | Key_Set | Type=Text; Name=PSU_EMP_REVIEW_REVIEW_YEAR | 2009
1 | Scroll | Action  | Ret=8Scr1                                     | upd
1 | Text   | Set_Value | Name=PSU_EMP_REVIEW_REVIEW_DAYS&Scr1         | 365
2 | Scroll | Definition | def=parent; value=8Scr1                      | upd
2 | Scroll | Key_Set | Type=ComboBox; Name=PSU_EMP_RWV_RWV_REVIEWER_ID | 00013
2 | Scroll | Action  | rel=5Scr2                                     | upd
2 | ComboBox | Set_Value | Name=PSU_EMP_RWV_RWV_REVIEW_TYPE&Scr2         | S
```

**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**

- **type=value;** The step type for the key field.
  
  Example:
  
  type=Text;

- **name=value;** The name of the key field.

**Example**

**Image: Example of the Key_Set and Action actions**

This example illustrates using Key_Set and Action

<table>
<thead>
<tr>
<th>Scroll ID</th>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Scroll</td>
<td>Key_Set</td>
<td>type=Text; Name=PSE_SCR_REC01_PSE_KEY_1.V.1</td>
<td>ROW1</td>
</tr>
<tr>
<td>1</td>
<td>Scroll</td>
<td>Action</td>
<td>RET=Scr1</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td></td>
<td>Verify</td>
<td>Name=PSE_SCR_REC01_PSE_KEY_1.V.1&amp;Sscr1</td>
<td>ROW1</td>
</tr>
<tr>
<td>ComboBox</td>
<td></td>
<td>Set_Value</td>
<td>Name=PSE_SCR_REC01_PSE_COMBOScr1</td>
<td>second</td>
</tr>
<tr>
<td>Text</td>
<td></td>
<td>Set_Value</td>
<td>Name=PSE_SCR_REC01_CON_CHAR_01&amp;Sscr1</td>
<td>updated</td>
</tr>
</tbody>
</table>

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

Image: Example of RowCount action

This example illustrates using the RowCount action to perform a loop.

<table>
<thead>
<tr>
<th>Span ID</th>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Browser</td>
<td>Start_Login</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Browser</td>
<td>FrameSet</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Page</td>
<td>Prompt</td>
<td>MAINTAIN_SECURITY_USERMAINT.GBL</td>
<td>update</td>
</tr>
<tr>
<td></td>
<td>Text</td>
<td>Set_Value</td>
<td>Name=PSDFROEN_SRCH_GRPID</td>
<td>TEST1</td>
</tr>
<tr>
<td></td>
<td>Page</td>
<td>PromptOk</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Page</td>
<td>Get_To</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scroll</td>
<td>Key_Set</td>
<td>type=FontSize=PSROLEUSER_VW_ROLENAME</td>
<td>Dummy</td>
</tr>
<tr>
<td>1</td>
<td>Scroll</td>
<td>FlexCount</td>
<td>ret_Account</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Variable</td>
<td>Set_Value</td>
<td>bend_count</td>
<td>Add([count]+1)</td>
</tr>
<tr>
<td></td>
<td>Loop</td>
<td>For</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Text</td>
<td>Set_Property</td>
<td>Name=PSROLEUSER_VW_ROLENAME bend_count=value</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Log</td>
<td>Message</td>
<td>The role of user number bend_count is value</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Loop</td>
<td>Next</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 Click.

Exists

**Description**

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

SeeExists.

Get_Property

**Description**

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

See Get_Property.
GetLabel

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

MouseOver

Description
Fires the mouseover event to show a popup page. Enter the page name in the Recognition field.
See "Using Pop-up Pages (PeopleTools 8.53: PeopleSoft Application Designer Developer's Guide)"

MouseOverClose

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

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

To record a verification of a field value in a mouseover popup window:

1. From the recorder tool bar, click and drag the 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:

   **Image: Example of MouseOver with Span.Verify**

   This example illustrates the recorded steps.

<table>
<thead>
<tr>
<th>Browser</th>
<th>Start</th>
<th>id=QE_EMPLOYEE_EMPLID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Span</td>
<td>MouseOver</td>
<td>on</td>
</tr>
<tr>
<td>Span</td>
<td>Verify</td>
<td>Comment=QE_EMPLOY2_2DEPTID 22000</td>
</tr>
<tr>
<td>Span</td>
<td>MouseOverClose</td>
<td>id=QE_EMPLOYEE_EMPLID</td>
</tr>
</tbody>
</table>

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:

**Image: Example of MouseOver with Link.Click**

This example illustrates the recorded steps.

<table>
<thead>
<tr>
<th>Browser</th>
<th>WaitForNew</th>
<th>Span</th>
<th>MouseOver</th>
<th>id=OE_EMPLOYEEEMPLID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link</td>
<td>Click</td>
<td></td>
<td></td>
<td>Name=OE_EMPLOYEEEMPLID</td>
</tr>
<tr>
<td>Span</td>
<td>MouseOverClose</td>
<td></td>
<td>id=OE_EMPLOYEEEMPLID</td>
<td></td>
</tr>
</tbody>
</table>

**Related Links**
"Using Pop-up Pages (PeopleTools 8.53: PeopleSoft Application Designer Developer's Guide)"

**Verify**

**Description**

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

See Verify.

**Example**

This step validates a Span object that contains informational text:

**Image: Example of the Span.Verify action**

This example illustrates a Span.Verify step.

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

**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 ellipsis icon and select a test case to automatically populate the Recognition and Value fields with the test name and the test case name.
Test.Exec supports the use of parameters that enable you to pass dynamic values from a calling test to a library test.

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

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

See Using Parameters with Library Tests.

**Example**

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

**Image: Example of the Test Exec action**

This example illustrates a step that calls the test COPY_USERPROFILE with the DEFAULT, CASE_01, and CASE_02 test cases, then skips the call to PROCESS.

<table>
<thead>
<tr>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>Exec</td>
<td>CHILD_ONE</td>
<td>CASE_01</td>
</tr>
<tr>
<td>Test</td>
<td>Exec</td>
<td>CHILD_TWO</td>
<td>#IGNORE</td>
</tr>
</tbody>
</table>

**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 Exists.

**Get_Property**

**Description**

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

See Get_Property.

**GetLabel**

**Description**

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

See GetLabel.
Set_Value

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

See Set_Value.

Verify

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

See Verify.

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:

Image: Example of the Set_Value action

This example illustrates a test case that sets a variable, which is then used in other steps.

<table>
<thead>
<tr>
<th>Log</th>
<th>Message</th>
<th>This is a test of variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Set_Value</td>
<td>&amp;Var1=This is the value for Var1</td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>The value in Var1 is &amp;Var1</td>
</tr>
<tr>
<td>Variable</td>
<td>Set_Value</td>
<td>&amp;Var2</td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>The value in Var2 is &amp;Var2</td>
</tr>
<tr>
<td>Variable</td>
<td>Set_Value</td>
<td>&amp;Var3= &amp;Var1</td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>Var3 is a done of Var1. The value is &amp;Var3</td>
</tr>
</tbody>
</table>

Related Links
Variables
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

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:

Image: Example of the Wait actions

This example illustrates various uses of Wait.

<table>
<thead>
<tr>
<th>Wait</th>
<th>For_Seconds</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wait</td>
<td>For_Value</td>
<td>Type=Link;Name=STATUS;</td>
</tr>
<tr>
<td>Wait</td>
<td>For_Value</td>
<td>Type=Link;Name=STATUS; Refresh=Name=R3_REFRESH BTN; timeout=90;</td>
</tr>
</tbody>
</table>

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.
Chapter 10 Test Language Reference

Click

**Description**
Performs a mouse click on the specified object.

This is the list of objects that support this action:

- Button
  
  See Button.

- Image
  
  See Image.

- Link
  
  See Link.

- Span
  
  See Span.

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

**Image: Example of the Click action**
This example illustrates the use of the Click action with a Button object and a Link object:

<table>
<thead>
<tr>
<th>Button</th>
<th>Click</th>
<th>Name=PB_FILTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link</td>
<td>Click</td>
<td>Name=LAST_NAME$0</td>
</tr>
</tbody>
</table>

Exists

**Description**
Checks whether the object exists on the page.

This is the list of objects that support this action:

- Button
  
  See Button.

- CheckBox
  
  See CheckBox.

- ComboBox
  
  See ComboBox.
Parameters

\[ ret=\&\text{variable}; \]

- \( ret=true \) – 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:

Image: Example of the Exists action

This example illustrates 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 Using the Message Tool.

Some objects have properties that are different from what you might expect.

For example:

- The value property for a check box returns Y for selected, N for deselected.
- Combo boxes return the translate value of the selection for the value property.
  The full text of the selected item is available as the text property.
- Radio buttons return the translate value of the selection for the value property.
- The label of the selected item is a separate label object.

This is the list of objects that support this action:

- Button
  See Button.
- CheckBox
  See CheckBox.
- ComboBox
  See ComboBox.
- Image
  See Image.
- Link
  See Link.
- LongText
  See LongText.
- MultiSelect
  See MultiSelect.
- Radio
  Refer to the Radio step type entry for details.
See Radio.

- Span
  
  See Span.

- Text
  
  See Text.

**Parameters**

prop=value;  

The property name.

ret=&variable;  

The return value.

**Example**

**Image: Example of the Get_Property action**

This example illustrates the Get_Property action.

<table>
<thead>
<tr>
<th>Text</th>
<th>Get_Property</th>
<th>Name=submit; ret=&amp;statusProp; prop=Status;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log</td>
<td>Get_Property</td>
<td>Name=submit; ret=&amp;statusProp; prop=Status;</td>
</tr>
<tr>
<td>Button</td>
<td>Get_Property</td>
<td>Name=submit; ret=&amp;statusProp; prop=TagLabel;</td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>This is the TagLabel for the button: &amp;statusProp;</td>
</tr>
<tr>
<td>Button</td>
<td>Get_Property</td>
<td>Name=submit; ret=&amp;statusProp; prop=Valuel;</td>
</tr>
<tr>
<td>Log</td>
<td>Message</td>
<td>This is the Value for the button: &amp;statusProp;</td>
</tr>
</tbody>
</table>

**GetLabel**

**Description**

Gets the label for the specified HTML object.

This is the list of objects that support this action:

- CheckBox
  
  See CheckBox.

- ComboBox
  
  See ComboBox.

- LongText
  
  See LongText.

- MultiSelect
  
  See MultiSelect.
• Pwd
  See Pwd.
• Radio
  Refer to the Radio step type entry for details.
  See Radio.
• RichText
  Refer to the RichText step type entry for details.
  See RichText.
• Span
  See Span.
• Text
  See Text.

**Set_Value**

**Description**
Sets the field value in the browser object.

This is the list of objects that support this action:

• CheckBox
  See CheckBox.
• ComboBox
  See ComboBox.
• LongText
  Use the SetValue_InModal action to set the value of a long text field on a modal page.
  See SetValue_InModal.
  See LongText.
• MultiSelect
  See MultiSelect.
• Pwd
  See Pwd.
• Radio
Refer to the Radio step type entry for details.

See Radio.

- Text
  
  See Text.

**Example**

**Image: Example of the Set_Value action**

This example illustrates the Set_Value action.

<table>
<thead>
<tr>
<th>LongText</th>
<th>Set_Value</th>
<th>name=SQALE_SQA_DESCRIPTION</th>
<th>long</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Set_Value</td>
<td>name=SQALE_SQA_PSE_DESCR</td>
<td>ebox</td>
</tr>
<tr>
<td>CheckBox</td>
<td>Set_Value</td>
<td>name=SQALE_SQA_CHECKBOX</td>
<td>Y</td>
</tr>
<tr>
<td>ComboBox</td>
<td>Set_Value</td>
<td>name=SQALE_SQA_COMBO_OPTION</td>
<td>2</td>
</tr>
<tr>
<td>Radio</td>
<td>Set_Value</td>
<td>name=SQALE_SQA_OPTION</td>
<td>2</td>
</tr>
</tbody>
</table>

**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 CheckBox.

- ComboBox
  
  See ComboBox.

- LongText
  
  See LongText.

- MultiSelect
  
  See MultiSelect.

- Radio
  
  Refer to the Radio step type entry for details.

  See Radio.

- Span
  
  See Span.
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:

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

This example illustrates 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:

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

This example illustrates 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:

**Image: Example of using #DIS#**

This example illustrates using #DIS# to verify that a field is display only and contains a specific date.

| Text | Set Value | Name=PA_CALCULATION_BEN_CMT_DATE | #DIS#07/12/2000 |

**#DTTM**

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

**Related Links**
See Also:#TODAY

**#EXIST#**

**Description**
Verifies the existence of a field.

If the field exists in the application, a Pass is logged. If the field is not found, a Fail is logged.

If a value is passed after the closing # and the field exists, PTF tries to set the field to that value.
Example

In this example, the first step checks for whether the Benefit Plan field exists in the application and logs
a Fail if it is not found. The second step not only checks for the existence of the field, it attempts to enter
the value **KUHP** into it:

**Image: Example of using #EXIST#**

This example illustrates using #EXIST#, 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.

<table>
<thead>
<tr>
<th>Text</th>
<th>Set_Value</th>
<th>Name=PA_CLC_CLN_INPT_BENEFIT_PLAN</th>
<th>#EXIST#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Set_Value</td>
<td>Name=PA_CLC_CLN_INPT_BENEFIT_PLAN</td>
<td>#EXIST#KUHP</td>
</tr>
</tbody>
</table>

Related Links

#NOTEXIST#

#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:

**Image: Example of #FAIL#**

This example illustrates using #FAIL#, 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 |

Related Links

#WARN#

#IGNORE

**Description**

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

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

**Image: Example of the #IGNORE reserved word**

This example illustrates

<table>
<thead>
<tr>
<th>Type</th>
<th>Action</th>
<th>Recognition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>Exec</td>
<td>CHILD_ONE</td>
<td>CASE_01</td>
</tr>
<tr>
<td>Test</td>
<td>Exec</td>
<td>CHILD_TWO</td>
<td>#IGNORE</td>
</tr>
</tbody>
</table>

#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:

<table>
<thead>
<tr>
<th>Type of Match</th>
<th>Pattern</th>
<th>Match (Log a Pass)</th>
<th>No Match (Log a Fail or Warn)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple characters</td>
<td>a*a</td>
<td>aa, aBa, aBBBa</td>
<td>ABC</td>
</tr>
<tr>
<td>Multiple characters</td>
<td><em>ab</em></td>
<td>abc, AABB, Xab</td>
<td>aZb, bac</td>
</tr>
<tr>
<td>Multiple characters</td>
<td>ab*</td>
<td>abcdefg, abc</td>
<td>cab, aab</td>
</tr>
<tr>
<td>Special character</td>
<td>a[*]a</td>
<td>a*a</td>
<td>Aaa</td>
</tr>
<tr>
<td>Single character</td>
<td>a?a</td>
<td>aaa, a3a, aBa</td>
<td>ABBBa</td>
</tr>
<tr>
<td>Single digit</td>
<td>a#a</td>
<td>a0a, a1a, a2a</td>
<td>aaa, a10a</td>
</tr>
<tr>
<td>Range of characters</td>
<td>[a-z]</td>
<td>f, p, j</td>
<td>2, &amp;</td>
</tr>
<tr>
<td>Outside a range</td>
<td>![a-z]</td>
<td>9, &amp;, %</td>
<td>b, a</td>
</tr>
<tr>
<td>Not a digit</td>
<td>![0-9]</td>
<td>A, a, &amp;,</td>
<td>0, 1, 9</td>
</tr>
<tr>
<td>Combined</td>
<td>a[!b-m]#</td>
<td>~ An9, az0, a99</td>
<td>abc, aj0</td>
</tr>
</tbody>
</table>
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:

Image: Example of using #LIKEF#

This example illustrates using #LIKEF# to verify only the first several characters of a text entry.

<table>
<thead>
<tr>
<th>Text</th>
<th>Set Value</th>
<th>Name=PA_CLC_CLN_INPT_BENEFIT_PLAN</th>
<th>#LIKEF#US*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Set Value</td>
<td>Name=PA_CLC_CLN_INPT_BENEFIT_PLAN</td>
<td>#LIKEF#US*1</td>
</tr>
</tbody>
</table>

#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:

Image: Example of using #LIKEF##TODAY*

This example illustrates using #LIKEF##TODAY to verify if a specific date is today’s date.

<table>
<thead>
<tr>
<th>Text</th>
<th>Set Value</th>
<th>Name=PA_PROP_YOUC_H_CREAT_DTTM</th>
<th>#LIKEF##TODAY*</th>
</tr>
</thead>
</table>

#LIKEW#

Description
The #LIKEW# reserved word is used the same as #LIKEF# except it logs a Warning rather than a Fail. For complete details for using #LIKEW# see #LIKEF#.

Related Links
#LIKEF#

#LIST#

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

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

Example
This example shows the use of the #LIST# reserved word:
In this example, the first step verifies the existence of items in the list. The second step verifies that the items exist and verifies that Individual is selected.

**Image: Example of using #LIST#**

This example illustrates using #LIST#, 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.

<table>
<thead>
<tr>
<th>ComboBox</th>
<th>Set_Value</th>
<th>Name=PA_CALCULATION_CALCULATION_TYPE</th>
<th>#LIST#Individual</th>
<th>Pre-Defined Group</th>
<th>Pre-Defined List</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComboBox</td>
<td>Set_Value</td>
<td>Name=PA_CALCULATION_CALCULATION_TYPE</td>
<td>#LIST#Individual</td>
<td>Pre-Defined Group</td>
<td>Pre-Defined List</td>
</tr>
</tbody>
</table>

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:

**Image: Example of using #LIST# with translate values**

This example illustrates using #LIST# with translate values.

<table>
<thead>
<tr>
<th>ComboBox</th>
<th>Set_Value</th>
<th>Name=PA_CALCULATION_CALCULATION_TYPE</th>
<th>#LIST#IGIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComboBox</td>
<td>Set_Value</td>
<td>Name=PA_CALCULATION_CALCULATION_TYPE</td>
<td>#LIST#IGIL</td>
</tr>
</tbody>
</table>

### #NOTEXIST#

**Description**

The opposite of the #EXIST# reserved word, #NOTEXIST# verifies that a field does not exist. If the field does not exist, a Pass is logged. If the field does exist, a Fail is logged.

**Related Links**

- #EXIST#

### #NOTHING#

**Description**

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

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

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

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

**Image: Example of using #NOTHING**

This example illustrates using #NOTHING to select a blank and to verify that the field is blank.

<table>
<thead>
<tr>
<th>ComboBox</th>
<th>Set Value</th>
<th>Name=PA_CALCULATION_CALC_REASON</th>
<th>#NOTHING</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComboBox</td>
<td>Verify</td>
<td>Name=PA_CALCULATION_CALC_REASON</td>
<td>#NOTHING</td>
</tr>
</tbody>
</table>

### #PREFIX#

**Description**

The #PREFIX# reserved word substitutes the text in the Prefix field in the Test Editor for the #PREFIX# string in the Value field.

This substitution is useful when developing a test that adds new data. It enables you to modify each new added record slightly so that the test is able to successfully add unique data each time the test is executed.

**Example**

Suppose you entered *add* in the Prefix field, as in this example:

**Image: Example of Prefix field**

This example illustrates the prefix field with the value *add*.

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

**Image: Example of using #PREFIX#**

This example illustrates a test step where the #PREFIX# is used in the value field.

<table>
<thead>
<tr>
<th>Text</th>
<th>Set Value</th>
<th>Name=DERIVED_CLONE_OPRID</th>
<th>#PREFIX#USER</th>
</tr>
</thead>
</table>

**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:

Image: Example of using #TODAY

This example illustrates using #TODAY.

<table>
<thead>
<tr>
<th>Text</th>
<th>Set Value</th>
<th>ID=PA_CLC_EMP_YW_EVENT_DT</th>
<th>#TODAY</th>
</tr>
</thead>
</table>

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:

Image: Example of using #TODAY+10

This example illustrates using #TODAY+10 to verify that the calculation event date is 10 days in the future.

<table>
<thead>
<tr>
<th>Text</th>
<th>Set Value</th>
<th>ID=PA_CLC_EMP_YW_EVENT_DT</th>
<th>#TODAY+10</th>
</tr>
</thead>
</table>

#WARN#

Description

This reserved word works the same as #CHECK# but does not update the value after performing the comparison. If a mismatch is found, a Warning is logged; otherwise, a Pass is logged.

You would use #WARN# rather than #CHECK# when you do not want to update the field if a mismatch exists.

Related Links

#FAIL#

System Variables

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%case%</td>
<td>Current test case name.</td>
</tr>
<tr>
<td>%component%</td>
<td>Current component name.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>%env.appserver%</td>
<td>Application server name.</td>
</tr>
<tr>
<td>%env.database%</td>
<td>Database name and database type. For example: T852U11 /DB2</td>
</tr>
<tr>
<td>%eo.dbname%</td>
<td>Execution option database name.</td>
</tr>
<tr>
<td>%eo.dbserver%</td>
<td>Execution option database server name.</td>
</tr>
<tr>
<td>%eo.dbuser%</td>
<td>Execution option database user name.</td>
</tr>
<tr>
<td>%eo_logfolder%</td>
<td>Execution option log folder.</td>
</tr>
<tr>
<td>%eo.name%</td>
<td>Current execution option name.</td>
</tr>
<tr>
<td>%eo.platform%</td>
<td>Execution option platform.</td>
</tr>
<tr>
<td>%eo.pserver%</td>
<td>Execution option process server.</td>
</tr>
<tr>
<td>%eo.pshome%</td>
<td>Execution option PS_HOME path.</td>
</tr>
<tr>
<td>%eo_url%</td>
<td>Execution option URL.</td>
</tr>
<tr>
<td>%env.toolsrel%</td>
<td>The PeopleTools version.</td>
</tr>
<tr>
<td>%eo.verbose%</td>
<td>Execution option verbose flag.</td>
</tr>
<tr>
<td>%menu%</td>
<td>Current menu name.</td>
</tr>
<tr>
<td>%frame%</td>
<td>The current frame name.</td>
</tr>
<tr>
<td>%log.id%</td>
<td>Current log number.</td>
</tr>
<tr>
<td>%page%</td>
<td>Current page name.</td>
</tr>
<tr>
<td>%pid%</td>
<td>The last process instance number.</td>
</tr>
<tr>
<td>%test%</td>
<td>Current test name.</td>
</tr>
</tbody>
</table>

**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
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.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add(10</td>
<td>-12</td>
</tr>
<tr>
<td>Add(10.43</td>
<td>10.55</td>
</tr>
</tbody>
</table>

Concat

Syntax
Concat(string1|string2|string3|...)

Description
Concatenates the strings in the parameters.

Parameters
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:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>concat(hello</td>
<td>and</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date()</td>
<td>07/04/2011</td>
</tr>
</tbody>
</table>

Day

Syntax
Day(date_value)

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

Parameters
date_value
A date value, such as the value returned by the Date() function.
Returns
Returns the day portion of the date value provided as a parameter.

Example
The following table presents examples of using the Day function:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day(February 13, 2012)</td>
<td>13</td>
</tr>
<tr>
<td>Day(Date())</td>
<td>13</td>
</tr>
</tbody>
</table>

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

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:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Divide(75</td>
<td>13.5)</td>
</tr>
<tr>
<td>Divide(-75</td>
<td>13.5</td>
</tr>
</tbody>
</table>
GetField

Syntax

GetField(string, segment, delimiter)

Description

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

Parameters

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.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetField(a/b/c</td>
<td>1</td>
</tr>
<tr>
<td>GetField(a/b/c</td>
<td>2</td>
</tr>
<tr>
<td>GetField(a/b/c</td>
<td>5</td>
</tr>
<tr>
<td>GetField(a/b/c</td>
<td>-1</td>
</tr>
</tbody>
</table>

Hour

Syntax

Hour(time_value)

Description

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

Parameters

time_value

A time value, such as the value returned by the Time() function.
Returns

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

Example

The following table presents examples of using the Hour function:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hour(13:07:25  )</td>
<td>13</td>
</tr>
<tr>
<td>Hour(Time())</td>
<td>13</td>
</tr>
</tbody>
</table>

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

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:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>instr(ABCDEFG</td>
<td>c)</td>
</tr>
<tr>
<td>instr(ABCDEFG</td>
<td>C)</td>
</tr>
<tr>
<td>instr(ABCDEFG</td>
<td>CDE)</td>
</tr>
<tr>
<td>instr(ABCDEFG</td>
<td>zz)</td>
</tr>
</tbody>
</table>
### LCase

**Syntax**

LCase(*string*)

**Description**

Converts a string to lowercase.

**Parameters**

- **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.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>lcase(Hello World 1234)</td>
<td>hello world 1234</td>
</tr>
</tbody>
</table>

### Left

**Syntax**

Left(*string*|*length*)

**Description**

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

**Parameters**

- **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.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>left(Hello World</td>
<td>5)</td>
</tr>
</tbody>
</table>

Len

Syntax
Len(string)

Description
Returns the length of string as an integer.

Parameters

string A text string.

Returns
Returns an integer indicating the length of string.

Example
The following table presents example of using Len function.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>len(Hello World)</td>
<td>12</td>
</tr>
</tbody>
</table>

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

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:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>MakeDate(Add(Year(Date())</td>
<td>1)</td>
</tr>
<tr>
<td>MakeDate(Add(Year(Date())</td>
<td>1)</td>
</tr>
</tbody>
</table>

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

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 MakeTime function:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>MakeTime(Add(Hour(Time()))</td>
<td>12)</td>
</tr>
<tr>
<td>MakeTime(23</td>
<td>Add(60</td>
</tr>
<tr>
<td>MakeTime(23</td>
<td>Add(60</td>
</tr>
<tr>
<td>MakeTime(23</td>
<td>Add(60</td>
</tr>
</tbody>
</table>

**Minute**

**Syntax**

Minute(time_value)

**Description**

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

**Parameters**

- time_value: A time value, such as the value returned by the Time() function.

**Returns**

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

**Example**

The following table presents examples of using the Minute function:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minute(13:07:25 )</td>
<td>7</td>
</tr>
</tbody>
</table>
Month

Syntax
Month(date_value)

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

Parameters
date_value A date value, such as the value returned by the Date() function.

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

Example
The following table presents examples of using the Month function:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Month(February 13, 2012)</td>
<td>2</td>
</tr>
<tr>
<td>Month(Date())</td>
<td>2</td>
</tr>
</tbody>
</table>

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
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.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiply(10.3</td>
<td>13.45)</td>
</tr>
<tr>
<td>Multiply(10.3</td>
<td>-13.45</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Now()</td>
<td>07/04/2011 12:20 PM</td>
</tr>
</tbody>
</table>

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**

- **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.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>replace(original text</td>
<td>i</td>
</tr>
</tbody>
</table>

**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**

- **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 Right function.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>right(Hello World</td>
<td>5)</td>
</tr>
</tbody>
</table>
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
- **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.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round(-130.456)</td>
<td>-130</td>
</tr>
<tr>
<td>Round(-130.456</td>
<td>dec=2)</td>
</tr>
<tr>
<td>Round(-130.455</td>
<td>dec=2)</td>
</tr>
</tbody>
</table>

Second

Syntax
Second(time_value)

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

Parameters
- **time_value**: A time value, such as the value returned by the Time() function.
Returns

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

Example

The following table presents examples of using the Second function:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Second(13:07:25 )</td>
<td>25</td>
</tr>
<tr>
<td>Second(Time())</td>
<td>25</td>
</tr>
</tbody>
</table>

SubStr

Syntax

SubStr(source_str|start_pos||length|)

Description

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

If length is not specified, SubStr returns the substring starting at the position specified in start_pos and continuing to the end of the string.

Parameters

- **source_str**: A string from which to extract a substring.
- **start_pos**: A number representing the character position in source_str where the substring starts, starting at 1.
- **length**: 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.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>substr(12345678</td>
<td>2</td>
</tr>
<tr>
<td>substr(12345678</td>
<td>2)</td>
</tr>
</tbody>
</table>
Subtract

**Syntax**

Subtract($number1$|$number2$|$number3$|...$)

**Description**

Use the Subtract function to subtract a series of numbers. Numbers can be decimal and negative.

**Parameters**

- **number1**
  - Initial number.
- **number2**
  - Number to be subtracted.
- **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.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subtract(10</td>
<td>2</td>
</tr>
<tr>
<td>Subtract(10</td>
<td>-2</td>
</tr>
</tbody>
</table>

Sum

**Syntax**

Sum(Index| value| delimiter)

**Description**

Sum works with the HTMLTable indexes.

**Note:** In PeopleTools releases prior to 8.53 commas were supported as a delimiter for the sum function parameters. If any tests exist in the database using that format, the commas will be converted to pipes in the upgrade.

**Parameters**

- **Index**
  - The HTMLTable index string, such as 2/5/4. An index string is the return value of CellGetIndex.
See CellGetIndex.

**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.
The character must be wrapped in quotes.

**Example**
The following table presents examples of using Sum function.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>`sum(&quot;2/5/4&quot;</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2 is added to the first section of the string.</td>
</tr>
<tr>
<td>`sum(&quot;2/5/4&quot;</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>1 is subtracted from the third section of the string.</td>
</tr>
<tr>
<td>`Sum(&amp;index</td>
<td>-4</td>
</tr>
<tr>
<td></td>
<td>4 is subtracted from the third section of the string in the variable &amp;index.</td>
</tr>
</tbody>
</table>

**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:
<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time()</td>
<td>13:07:25</td>
</tr>
<tr>
<td>Time()</td>
<td>1:07:25 PM</td>
</tr>
</tbody>
</table>

**Trim**

**Syntax**

Trim(string)

**Description**

Returns a string with all leading and trailing spaces removed.

**Parameters**

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.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>trim( Hello World )</td>
<td>Hello World</td>
</tr>
</tbody>
</table>

**UCase**

**Syntax**

UCase(string)

**Description**

Converts a string to uppercase.

**Parameters**

string  
The string to be converted.

**Returns**

Returns a string resulting from converting string to uppercase.
Example

The following table presents example of using UCase function.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucase(Hello World 1234)</td>
<td>HELLO WORLD 1234</td>
</tr>
</tbody>
</table>

Year

Syntax

Year(date_value)

Description

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

Parameters

date_value

A date value, such as the value returned by the Date() function.

Returns

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

Example

The following table presents examples of using the Year function:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year(February 13, 2012 )</td>
<td>2012</td>
</tr>
<tr>
<td>Year(Date())</td>
<td>2012</td>
</tr>
</tbody>
</table>
Appendix A

Reserved Words Quick Reference

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 #CHECK#.

#DIS#  Checks whether an object is display-only.

See #DIS#.

#DTTM#  Enters the current date and time into the application.

See #DTTM#.

#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 #EXIST#, #NOTEXIST#.

#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 #FAIL#, #WARN#.

#IGNORE#  Place the #IGNORE reserved word in the Value field of a Test. Exec step to skip the call to the child test.

See #IGNORE#.

#LIKEF# and #LIKEW#  Match strings using LIKE. If no match exists, PTF logs a Fail or Warning. PTF does not update the value.

See #LIKEF#, #LIKEW#.

#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 #LIST#.

#NOTHING

Deletes a value in the object or verifies that it is blank. If the object is a ComboBox and the action is Set, then PTF selects a blank item.

See #NOTHING.

#PREFIX#

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

See #PREFIX#.

#TODAY

Enters the current date into the application.

See #TODAY.