Converting Classic PIA Components to PeopleSoft Fluid User Interface
Disclaimer

This material has not been submitted to any formal Oracle test and is published as is. It has not been the subject of rigorous review. Oracle assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by Oracle for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Information in this red paper was developed in conjunction with use of the products specified and is limited in application to those specific hardware and software products and levels.

Oracle may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents.
# Table of Contents

- Introduction 1
- Structure of This Red Paper 1
- Related Materials 1
- Overview 1
  - Who Should Read This Paper? 1
  - Before You Begin 1
- Converting Classic PIA Applications for Fluid User Interface Deployment 1
  - Understanding Classic PIA to Fluid Conversion 2
  - Step 1: Create the Fluid Component 3
  - Step 2: Create the Fluid Page 4
  - Step 3: Add Fluid Pages to the Fluid Component 8
  - Step 4: Update PeopleCode References 8
  - Step 5: Update, Remove, or Redesign Unsupported Controls 8
    - Help Text 8
    - Scroll Area 9
    - Classic Grid 9
    - Field Style Property 9
    - HR Core Navstack 10
    - HGrid 10
    - Toolbar 10
  - Step 6: Enable Search Pages 10
  - Step 7: Adjust and Finalize Layout 11
Introduction

This red paper is a practical guide for technical users, installers, system administrators, and programmers who implement, maintain, or develop applications for your PeopleSoft system. In this red paper, we discuss guidelines on how to convert existing classic PIA components for fluid deployment.

Although this document does not address every conceivable scenario that one could encounter while porting existing PeopleSoft pages and components for fluid deployment the content that appears in this document is intended to provide general steps, applying to a broad range of situations.

Structure of This Red Paper

This red paper provides guidance for converting existing PeopleSoft components for fluid mode deployment. There is an overview of conceptual information, followed by a set of steps used for converting classic pages to fluid pages, with some additional information provided as appendixes.

Oracle updates this document as needed so that it reflects the most current feedback from the field. Therefore, the structure, headings, content, and length of this document may vary with each posted version. To see if the document has been updated since you last downloaded it, compare the date of your version to the date of the version that is posted on My Oracle Support.

Related Materials

This paper is not a general introduction to PeopleSoft Fluid User Interface and is written for experienced IT professionals with a good understanding of PeopleTools development tools, such as Application Designer, PeopleCode, and the PeopleSoft Internet Architecture. To take full advantage of the information in this document, you should have a basic understanding of system administration, basic Internet architecture, integration technologies, relational database concepts and SQL, and how to use PeopleSoft applications.

This document does not replace the PeopleTools 8.5x product documentation. Before you read this document, you should become familiar with the PeopleTools product documentation to ensure that you have a well-rounded understanding of the technology.

Much of the information in this document may eventually be incorporated into subsequent versions of the product documentation.

The following product documentation discusses many of the fundamental concepts that are related to the PeopleSoft Fluid User Interface:
» PeopleTools 8.54: PeopleSoft Fluid User Interface Developer’s Guide
» PeopleTools 8.54: PeopleSoft Application Designer Developer’s Guide
» PeopleTools 8.54: PeopleCode API Reference
» PeopleTools 8.54: PeopleCode Language Reference
Overview

This paper describes the general approach an organization can use to make existing PeopleSoft classic components available to users accessing the system in fluid mode. A “classic” PeopleSoft component is one that contains pages built using the techniques of page development for the PeopleSoft Internet Architecture (PIA), which is the “pure internet architecture” used for deploying PeopleSoft applications for well over a decade.

While the PeopleSoft Fluid User Interface uses the same physical architectural elements as the classic PIA applications (web server, application server, and so on), the technique used for designing pages and components is different. Also different, the run time behavior of a fluid page can respond to the size of the device (form factor) accessing the system and display an interface that seems natural for that device, be it smartphone, tablet, or PC.

PeopleSoft Fluid User Interface moves away from pixel-perfect page layout page design and provides greater flexibility by enabling greater use of CSS 3 and the features of HTML 5. You create the structure of a page in Application Designer, and CSS handles the positioning of the elements at run time.

Fluid components are special in that they add the ability for a single component to be displayed in multiple ways. The appropriate visual representation will be picked and rendered depending on the device being used to view it. This enables fluid components to deliver an experience that is natural across a wide range of device types, including mobile phones and tablets running iOS and Android operating systems.

The PeopleSoft Fluid User Interface does not completely replace classic PIA pages. Classic pages and components remain fully supported and fulfill an important role for components that do not need to respond to device screen size. There is also an effort underway to bring the appearance of classic PIA in line with the modern visual appearance of PeopleSoft Fluid User Interface. While this will make classic components more modern (visually), it will take additional design and development effort to enable the component to adapt to device size and take advantage of device features.

This document establishes a pattern that can be followed to port an existing classic component and enable it to take advantage of fluid features.

This section includes the following topics:

- Who should read this paper?
- Before you begin
- Common terms

Who Should Read This Paper?
Developers and technical experts involved with building or implementing fluid pages for PeopleSoft applications.

Before You Begin

Prior to beginning any fluid development project, make sure you:

- Are working with a database on at least the PeopleTools 8.54 version.
- Use browsers that support fluid applications (see My Oracle Support > Certifications).
- Are familiar with the content in the product documentation for PeopleTools 8.54: PeopleSoft Fluid User Interface Developer’s Guide.

Converting Classic PIA Applications for Fluid User Interface Deployment

This section contains an overview and describes these steps:

- Step 1: Create the Fluid Component
» Step 2: Create the Fluid Page
» Step 3: Add Fluid Pages to the Fluid Component
» Step 4: Update PeopleCode References
» Step 5: Update, Remove, or Redesign Unsupported Controls
» Step 6: Enable Search Pages
» Step 7: Adjust and Finalize Layout
» Step 8: Implement Responsive and Adaptive Behaviors

Understanding Classic PIA to Fluid Conversion

Before attempting to port a classic page for fluid deployment, the designer must carefully rethink the user experience of the classic page you are porting. Many user experience decisions that worked well in classic mode do not fit into fluid patterns for navigation and design. While there may be some straightforward cases when a classic component can be converted using the exact steps mentioned below, you are likely to encounter several cases where redesign is preferable to a direct port.

Redesigning may be the most suitable option because the fluid user experience differs from the classic user experience in significant ways (see the following table).
TABLE 1. DIFFERENCES BETWEEN CLASSIC AND FLUID

<table>
<thead>
<tr>
<th>Difference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layout</td>
<td>The layout of a component developed for the fluid deployment needs to display correctly when viewed from any supported device. Since fluid is supported on mobile phones and tablets, this works out to a very large set of devices (see Appendix). To make best use of the available space, a fluid layout must automatically reposition page controls depending on the available space.</td>
</tr>
<tr>
<td>User interaction patterns</td>
<td>The pattern for user interaction on a mobile device is different from the pattern for interaction with a keyboard and a mouse. Additional factors have to be taken into consideration such as touch screens in addition to mouse clicks, and slide in addition to drag. An obvious example is that buttons need to be larger in order to provide a suitable touch target. In addition, the location of the back button, menus and actions in the UI need to be consistent with patterns followed by mobile operating systems to provide the expected user experience.</td>
</tr>
<tr>
<td>User expectation</td>
<td>The user expectation from mobile devices is different. The changed style of usage must be taken into consideration when designing for a mobile device. For example, a mobile application may be used while on the go, and therefore must be designed for a shorter attention span.</td>
</tr>
</tbody>
</table>

To cater to the changed user expectations, the patterns for navigation and even page constructs for fluid pages are simplified in comparison with classic pages. Navigation in the fluid interface is often driven by global search, context, and related actions, rather than from a search page. Once a fluid component is built, it is available to be added as a tile on the end user’s homepage. Ideally, the end user should be able to start from a home page tile and end on the transaction page without having to use a keyboard.

Mobile devices have additional hardware capabilities such as GPS, accelerometer, sensors, and cameras. These are accessible to the fluid developer familiar with HTML5 API’s.

*The screenshots provided in the steps use the “Business Unit” component as an example for direct port from classic PIA to fluid.*

**Step 1: Create the Fluid Component**

To create a fluid component:
1. Open the classic component you want to port to fluid, select File > Save As, and enter a new name.

![Save As dialog box]

Though not mandatory, it is recommended to adopt a naming convention for fluid components (and pages). In this case, because it is the HR_BUSINESS_UNIT component being ported to fluid, and FL is added to the name to indicate "fluid." For example, HR_BUSINESS_UNIT_FL.

2. Select Yes on the dialog box when prompted to save a copy of component PeopleCode from the classic component.

![Application Designer dialog box]

Since fluid is mainly a UI update, existing PeopleCode logic, records, and application packages ported from a classic component will continue to work with the fluid component.

3. Open the Component Properties dialog box, and on the Fluid tab, select Fluid Mode.

![Component Properties dialog box]

4. You will also want to enable the Fluid header toolbar options as per your requirements.

Refer to PeopleSoft Online Help for further details.


http://docs.oracle.com/cd/E17566_01/epm91pbr0/eng/psbooks/psft_homepage.htm

Step 2: Create the Fluid Page

To create a fluid page:
1. Select File > New, and select *Page (Fluid)* from the New Definition dialog box, and click OK.

2. Select a layout page from the Choose Layout Page dialog box.

To minimize your effort, pick a layout page that is the best match for the page that you are porting.

In our example, the page being created is a simple, one-column layout, so using the layout: *PSL_APPS_CONTENT* is appropriate.

For more information on layout pages, see Appendix 3: Layout Pages.

3. Save the fluid page.

You will be prompted to save the fluid page. It's recommended to adopt a naming convention that will indicate from which classic page this fluid page is being ported. In this case, the page is named *BUS_UNIT_TBL_HR_FL* since the classic page is *BUS_UNIT_TBL_HR*.

Some layout pages provide PeopleCode in the page Activate event. When prompted, you should choose to save a copy of PeopleCode with the newly created page.

4. Clone the classic page you are porting, setting the page properties so that the page is now a subpage.

Select File > Save As. Select a page name indicating which page classic has been ported. In this case, the subpage is named *BUS_UNIT_TBL_H_SBP*. 


When prompted, do not copy the PeopleCode to a subpage. You can copy the PeopleCode manually at a later point into the main page. Page Activate PeopleCode on subpages is not executed.

Open Page Properties, and set the Page Type for the newly created page as Subpage.

It may be required to create multiple subpages from a single source page. Some layouts may have multiple areas into which fields can be added. If, for example, you use a two-panel layout, you would have two separate areas on which to place content - one to populate the folding left panel and a second area for the actual content of the page. This will force you to split the source page into two subpages and distribute the page controls as desired. In this case you can either clone the page several times and delete the non-required fields, or move the fields into newly created subpages manually as needed.

5. Insert the subpage(s), into the main layout page.

The layout page will have one or more areas into which fields can be added. Insert the subpages into the appropriate areas.
Copy the page Activate PeopleCode from the original page into the fluid page.

Now that all the fields of the original page have been moved into the fluid page, move the PeopleCode from the original page to the fluid page Activate event. This step must be repeated for every page that is in the component. If there are subpages on the page it may be required to clone the subpages as well so that the fluid component has no dependency on any classic component. It’s not recommended to share a subpage between a fluid and a classic page since some page field level properties are not cross-compatible between fluid and classic mode.

If your component used the PeopleTools toolbar action buttons for save, correct history, and so on, these do not automatically appear on the fluid converted component. You will have to add buttons to the bottom of your pages for each of these actions and assign toolbar actions to them as shown in the screenshot below.

Creating a footer page is an easy way to add the required buttons to every page in the component. A footer page is a new page type with PeopleTools 8.54 that, if used, is automatically shown at the bottom of all pages in a fluid component. For example, adding the footer page PT_FOOTER_SAVEONLY to your component will add the Save button on the bottom of every page in the component when it is rendered.
Step 3: Add Fluid Pages to the Fluid Component

In this step you add the new fluid pages to your cloned component, and remove the old ones.

If the previous step was performed correctly, the component buffer structure will remain unchanged for all your existing PeopleCode, apart from page names and component name references. For example:

```
Step 4: Update PeopleCode References

If there is PeopleCode in which the classic page names or the classic component name is referenced, update the references to point to the new fluid names to avoid PeopleCode errors.

Step 5: Update, Remove, or Redesign Unsupported Controls

Several controls and constructs that are valid in classic PIA are not intended for use in a fluid experience. You will need to remove or redesign these controls. This section discusses common considerations with such controls.

Help Text

Currently, the popup help text does not display in fluid mode.

However, a similar appearance can be generated using the popup group box type. Set on the Fluid tab for a group box, selecting *Popup* for Group Box Type.
Scroll Area
Currently, scroll buttons for next and previous are not yet supported in fluid scroll areas.

However the component buffer has the data and developers can implement their own mechanism to navigate the data.

Classic Grid
Putting a classic grid in a fluid mode component will result in a JavaScript error. Replace classic Grid Layouts with the fluid Original Flex Grid Layout.

PeopleTools 8.54 introduces several new grid styles, but the Original Flex Grid Layout option is closest to the classic grid appearance.

Update classic grids to flex grids by changing the Grid Layout on the Use tab of Grid Properties dialog box.

Field Style Property
The classic page field Style property on the Type tab should be moved to the Default Style Name page field property on the Fluid tab.

Style (Type tab)
Default Style Name (Fluid tab):

HR Core Navstack

(Enterprise or Shared Component controls)

The HR Core navstack is used in several self-service pages to return to the referring page.

* Required Field

Return

Navstack is not supported in fluid mode.

Remove the control and all PeopleCode referring to it. The Back button on the fluid banner provides the functionality of HR Core NavStack.

HGrid

The HGrid is used in manager self-service pages to represent the reporting structure of employees.

The HGrid is not supported in fluid mode.

Remove the widget or redesign it.

Toolbar

The toolbar is used in products such as Recruiting and ELM to present page level actions to self-service users.

Pages using the toolbar may require you to write CSS to realign the content depending on the icons used.

Step 6: Enable Search Pages

By default, fluid mode allows pivot grids and PeopleSoft Search Framework to be used to provide search features for a component.


http://docs.oracle.com/cd/E17566_01/epm91pbr0/eng/psbooks/psft_homepage.htm
Alternatively you can create your own custom fluid page to provide the search functionality.

Step 7: Adjust and Finalize Layout
At this point, the component is usable in fluid mode. The next step is to optimize the look and behavior.
You first want to identify and correct any layout inconsistency. Fine adjustments to the layout can be made by using CSS.
The PeopleSoft delivered stylesheet, PSSTYLEDEF_FMODE, provides numerous commonly used styles so that most of the time developers can adjust the layout without having to create their own stylesheet. An extract of some of the commonly used styles in the fluid stylesheet is provided in the appendices of this document.

For example, to align the "OR" in the screenshot above to a more desired place above the second input box, you can apply the provided style _psc_label_filler_ to the control using page-field properties.

It is also possible to attach your own stylesheet to the component using the PeopleCode construct:

AddStylesheet(Stylesheet.<STYLESHEETNAME>);

More information on the AddStylesheet function is provided in the PeopleCode Language Reference:


Step 8: Implement Responsive and Adaptive Behaviors
The final step in the process is to make the page adapt to the available display area. There are a couple of ways to accomplish this.
The first method is applying responsive design principles using media queries. PeopleTools provides a few media queries in the default fluid style-sheet, however it's also possible to write your own media queries and to attach them to the component using the AddStylesheet function.
The second method is by using *adaptive design* features provided by the PeopleTools system. PeopleTools categorizes devices under small, medium, large and extra large form factors based on the detected device width (measured in pixels). Fluid pages and page controls have an additional set of properties which allows styles to be added conditionally based on device type.

You can use these options to override the styles applied at the page field level, according to the width of the device as detected by the server. If needed, you can have different styles applied to a control depending on what device accesses a component.

In addition, controls or groups of controls can be suppressed entirely for a particular a form factor by selecting the appropriate checkbox in the Suppress On Form Factor group box, allowing different sections on a page to be served depending on device type.

For more information, see the production documentation for *PeopleTools 8.54: Fluid User Interface Developer's Guide*, "Creating Pages for PeopleSoft Fluid Applications," Setting Properties for Fluid Pages
Appendices

These appendices contain additional information that may be helpful for certain steps in this document.

This document contains these appendices:

» Appendix 1: Fluid Target Devices
» Appendix 2: External Resources
» Appendix 3: Layout Pages
» Appendix 4: Delivered PeopleSoft Style Classes
» Appendix 5: Common Terms Related to PeopleSoft Fluid User Interface
Appendix 1: Fluid Target Devices

When building a Fluid page, it's important to understand that there is a wide range of target devices accessing a page.

Mobile devices running Android and iOS are supported targets for fluid components in addition to the desktop operating systems that were supported by the previous versions of PeopleTools.

The primary target device for classic pages is desktop PC's. In the past, a majority of PC's had a resolution of 1024 x 768 or 800 x 600, a physical keyboard, and pointing device (mouse).

See the following website for screen resolution statistics for metrics regarding the percentages of website visitors using a screen resolution of at least 1024 x 768.

http://www.w3schools.com/browsers/browsers_display.asp

Note that the Apple iPad does follow this trend with a device independent resolution of 1024x968, though its default viewport is standardized differently.

This makes development considerations relatively simple. A single layout that looks and behaves in an acceptable way on devices with these common resolutions for PC was sufficient to cater to all supported devices for PeopleSoft. There would be some scrolling and whitespace inconvenience on the minimal set of devices having other resolutions.

Pointing devices such as a mouse and track pad which are common to PC devices are accurate in comparison with touch screens, so the UI for the PC can use small icons and links which take up less screen real estate. This allows designers to fit more information on the screen.

Over the years, higher resolution displays, both small and large have started to become more common and as technology evolves, productivity tablets running PC OS's but with a smaller screen and touch interface are becoming increasingly available. In addition, personal devices such as phones and tablets running iOS and Android and Windows Phone are increasingly being used to access company applications.

iOS

Apple manufactured mobile devices (iPhone, iPad, iPod touch) run iOS.

As such, there are a fixed number of devices, which means a fixed number of resolutions to support. The following are some screen dimensions (in DIPs):

- iPhone and iPod touch: 320 x 480
- iPad: 768 x 1024

Android

Android is a mobile OS which is maintained and developed by Google. Unlike Apple, Google doesn’t control the hardware on which Android is installed. As such, the device types, screen size, and hardware on which Android can be installed varies greatly.

The following links provide details on common screen sizes and resolutions for Android devices:

http://developer.android.com/about/dashboards/index.html#Screens

http://opensignal.com/reports/fragmentation.php
Appendix 2: External Resources

A good understanding of HTML, CSS and JavaScript is recommended in order to make the best use of fluid functionality. PeopleSoft Fluid User Interface is built on these languages and it fully exposes these underlying technologies to the developer.

A lot has been written on how to design high quality mobile friendly user interfaces. HTML and CSS are commonly used technologies for delivering native-like apps on mobile devices.

This section lists a selection of articles relating to the topic of designing for mobile devices and designing pages that look and behave like native apps across multiple devices sizes.

Information on HTML5, Responsive Web Design, and Adaptive Design
http://en.wikipedia.org/wiki/Responsive_web_design
http://en.wikipedia.org/wiki/Adaptive_web_design
http://www.techrepublic.com/blog/web-designer/what-is-the-difference-between-responsive-vs-adaptive-web-design/
http://alistapart.com/article/dao
http://www.webdirections.org/resources/john-allsopp-the-dao-of-web-design-revisited/
http://www.alistapart.com/articles/responsive-web-design
http://www.html5rocks.com/en/
http://http://www.w3schools.com/css/css3_intro.asp

iOS User Experience Guidelines
CH2-SW15

Android User Experience Guidelines
http://developer.android.com/guide/practices/screens_support.html
http://opensignal.com/reports/fragmentation.php
http://developer.android.com/about/dashboards/index.html#Screens

Windows User Experience Guidelines
http://blogs.msdn.com/b/b8/archive/2012/03/21/scaling-to-different-screens.aspx
Appendix 3: Layout Pages

Layout pages are a concept introduced with fluid user interface development. The use of layout pages is comparable to that provided by the “New Slide template” functionality of Microsoft PowerPoint.

By cloning a PeopleSoft delivered layout page, you are given a tested starting point so that you do not have to design every new page from scratch.

PeopleSoft will periodically provide new layout pages to satisfy other use cases, and customers can add to this library by creating new pages of type Layout Page in Application Designer.

This section describes some examples of delivered PeopleSoft layout pages.

For more information on layout pages, see PeopleSoft Online Help > PeopleTools 8.54 Fluid User Interface Developer’s Guide, “Creating Pages for PeopleSoft Fluid Applications,” Creating the Primary Fluid Page

PSL_APPS_CONTENT

PSL_APPS_CONTENT is a one column layout template.
HCSC_2PNL_LO_LFL
HCSC_2PNL_LO_LFL is a two panel layout with a left panel that can be hidden/shown by the user. (Delivered by HCM applications)

HR_ESS_LAYOUT_LFL
HR_ESS_LAYOUT_LFL is a layout with employee photo header for self service components in HCM. (Delivered by HCM applications)
Appendix 4: Delivered PeopleSoft Style Classes

This table describes some commonly used style classes delivered in the fluid style sheet (PS_STYLEDEF_FMODE).

### TABLE 2. COMMONLY USED DELIVERED STYLE CLASSES

<table>
<thead>
<tr>
<th>Style Class Name</th>
<th>Category</th>
<th>Construct</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ps_hidden</td>
<td>Display</td>
<td>Any</td>
<td>Set display attribute to none so element is not displayed.</td>
</tr>
<tr>
<td>psc_hidden</td>
<td>Display</td>
<td>Any</td>
<td>Sets display attribute to INLINE on element (preferred to use psc_display-inline).</td>
</tr>
<tr>
<td>ps_inline</td>
<td>Display</td>
<td>Any</td>
<td>Set display attribute to INLINE-BLOCK on element.</td>
</tr>
<tr>
<td>psc_inline</td>
<td>Display</td>
<td>Any</td>
<td>Set display attribute to BLOCK on element.</td>
</tr>
<tr>
<td>psc_display-inline</td>
<td>Display</td>
<td>Any</td>
<td>Floats the element left, right or none (respectively).</td>
</tr>
<tr>
<td>psc_display-inlineblock</td>
<td>Display</td>
<td>Any</td>
<td>Reserved turn surround field(s) which do not have labels on the left side of the control. Reserves space on the left to provide alignment of fields with labeled fields (default 33%).</td>
</tr>
<tr>
<td>psc_label_filler</td>
<td>Layout</td>
<td>Group Box (Layout)</td>
<td>Used turn surround field(s) which do not have labels on the left side of the control. Reserves space on the left to provide alignment of fields with labeled fields (default 33%).</td>
</tr>
<tr>
<td>psc_width&lt;num&gt;&lt;unit&gt;</td>
<td>Layout</td>
<td>Any</td>
<td>Sets the width of the element to num units, where num is a number between 1 and 100 and units is either px, em or pct.</td>
</tr>
<tr>
<td>psc_nolabel</td>
<td>Label</td>
<td>Field Grid</td>
<td>Forces all ps_box-label elements to be hidden from rendering (no space reserved).</td>
</tr>
<tr>
<td>psc_label-none</td>
<td>Label</td>
<td>Field Grid</td>
<td>Sets all children ps_box-label to INVISIBLE (space is still reserved).</td>
</tr>
<tr>
<td>psc_label-invisible</td>
<td>Label</td>
<td>Field Grid</td>
<td>Sets the width of ps_box-label to num pct, where num is a number between 1 and 100.</td>
</tr>
<tr>
<td>psc_label-width&lt;num&gt;pct</td>
<td>Label</td>
<td>Fields or group box set to .psc_label_filler</td>
<td>Sets the width of ps_box-label to num em, where num is a number between 1 and 50.</td>
</tr>
<tr>
<td>psc_label-width&lt;num&gt;em</td>
<td>Label</td>
<td>Fields or group box set to .psc_label_filler</td>
<td>Sets the width of the input control similar to psc_label-width. Num is a number between 1 and 100, unit is em, px or pct.</td>
</tr>
<tr>
<td>psc_control-width&lt;num&gt;&lt;unit&gt;</td>
<td>Control</td>
<td>Fields</td>
<td>Sets the height of the input control. Num is a number between 1 and 100, unit is em, px or pct.</td>
</tr>
</tbody>
</table>
### Appendix 5: Common Terms Related to PeopleSoft Fluid User Interface

The following table provides definitions for some of the common terms that are used in this paper as well as throughout the industry when discussing responsive design and adaptive design.

#### TABLE 3. COMMON TERMS

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive Design</td>
<td>A page design that uses properties published by the device's browser to serve a layout as appropriate for the device properties. Note that because of the way that CSS pixels, zooming, and viewport properties work on mobile devices, multiple fixed width layouts can be as effective as fluid layouts for building native-like websites for devices.</td>
</tr>
</tbody>
</table>
| CSS Pixel                 | A relative unit that is used when writing CSS. For example, the HTML below generates a div element of width 1 CSS Pixel:  

```html
<div style='background-color: red; height: 1px; width: 1px; '></div>
```

As a result of how density independent pixels behave, when a page containing the code above is opened on an original iPhone and on an iPhone 4, the size of the element is rendered visually exactly the same on both devices though the iPhone 4 has double the resolution.

1 CSS pixel = 1 DIP when zoom is 100%. |
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Height/Width</td>
<td>The height/width of the device in density independent pixels. Because content in PeopleSoft scrolls vertically, device width usually is the deciding factor for layout of responsive pages.</td>
</tr>
<tr>
<td>Density Independent Pixel (DIP)</td>
<td>A reference unit introduced and supported by mobile OS’s to reduce the complexity of coding for multiple device sizes. For example, the original iPhone has a physical resolution of 320 x 480 at 163 PPI. On the original iPhone, 1 DIP = 1 physical device pixel. The iPhone 4S has a physical resolution of 640 x 960 at 326 PPI. On the iPhone 4S, 1 DIP = 4 physical device pixels. The result is that though the iPhone 4S has four times the number of hardware pixels in comparison with the original iPhone, both phones have the same number of density independent pixels.</td>
</tr>
<tr>
<td>Device Pixel (or Hardware Pixel)</td>
<td>The smallest physical element of the display hardware.</td>
</tr>
<tr>
<td>Device Pixel Density</td>
<td>The number of hardware pixels represented by a density independent pixel on one axis. The first generation iPhone has a device pixel density of 1 (1 pixel = 1 DIP horizontally). The iPhone 4 has a device pixel density of 2 (1 pixel = 1 DIP). So while the iPhone 4 has twice the number of physical pixels in comparison with the first generation iPhone, both devices have the same number of density independent pixels.</td>
</tr>
<tr>
<td>Fixed Width Layout</td>
<td>The layout of the page doesn’t adjust itself when the browser is resized. For example, the visible content of the page wrapped within a container element that has its width set to a constant.</td>
</tr>
<tr>
<td>Fluid Layout</td>
<td>The page content flows to occupy the available space when the browser is resized.</td>
</tr>
<tr>
<td>Media Query</td>
<td>A CSS feature that enables developers to selectively enable and disable styles based on viewport, device, and browser properties.</td>
</tr>
<tr>
<td>Pixels Per Inch (PPI)</td>
<td>The number of physical pixels under one inch of the device on one axis. The same measure is referred to as dots per inch or DPI for printer devices. With CSS media queries, DPI may be used to refer to PPI on pixel screens as well.</td>
</tr>
<tr>
<td>Resolution</td>
<td>The resolution of a display is the number of pixels on each dimension. It is usually quoted as Width x Height. For example, a display with a maximum resolution of 1920 x 1080 is composed of 2073600 device pixels. Resolution may be quoted in physical pixels or density independent pixels.</td>
</tr>
<tr>
<td>Responsive Design</td>
<td>A page design that uses CSS media queries to alter its layout on the fly based on the properties of the viewport.</td>
</tr>
<tr>
<td>Viewport</td>
<td>The area of the device display that is available to the web page to show information. On mobile devices running a modern browser, developers can configure the default zoom and window size (including the size in CSS pixels) of the web browser’s viewable area by using the viewport meta tag. If a viewport is not specified, default properties are assigned to the viewport based on which OS is being used. Default viewport details for iOS and Android: <a href="http://developer.android.com/guide/webapps/targeting.html">http://developer.android.com/guide/webapps/targeting.html</a> <a href="http://developer.apple.com/library/ios/#documentation/AppleApplications/Reference/SafariWebContents/UsingtheViewport/UsingtheViewport.html#apple_ref/doc/uid/TP40006509-SW27">http://developer.apple.com/library/ios/#documentation/AppleApplications/Reference/SafariWebContents/UsingtheViewport/UsingtheViewport.html#apple_ref/doc/uid/TP40006509-SW27</a></td>
</tr>
</tbody>
</table>
Validation and Feedback

This section documents the real-world validation that this red paper has received.

The approach described in this document was tested on several delivered PeopleSoft components.

Customer Validation

Oracle is working with PeopleSoft customers to get feedback and validation on this document. Lessons that are learned from these customer experiences will be posted here.

Field Validation

Oracle Consulting Services has provided feedback and validation on this document. Additional lessons that are learned from field experience will be posted here.

Authors

Various representatives from PeopleTools and PeopleSoft application teams.

Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 24, 2015</td>
<td>Created document.</td>
</tr>
</tbody>
</table>