

Harmony Viewer Technical Content Development Guide

Author: Razorfish Healthware
Release: January 2013

TABLE OF CONTENTS

About this Document.....	4
Harmony Viewer Platform	5
Anatomy of a Presentation.....	6
HTML Slides.....	6
Navigation Layer	6
Supplemental Media Controllers	6
External Data Assets	7
Presentation Assets	7
File Structure and Naming Conventions	9
Modular Asset Library	9
Naming Conventions	10
Presentation Slides	11
Modal Slides.....	11
Slide Configuration File	15
HTML Guidelines.....	19
Base HTML Template.....	20
Optimal Font Utilization.....	24
Retina Display Support.....	25
Orientation Support.....	27
Touch Event Support	28
Video Support	30
Animation Support	31
Third Party App Integration.....	32
SystemBridge JavaScript Library.....	34
Navigation Layer	43
Default Navigation layer	43
Optional Customized Navigation Layer	47
Presentations	48
Presentation Configuration File	49
Slides Array	50
Active Call Storyflow.....	52
Active Call Branching	56
Per Presentation Properties	58
Presentation independent Data Assets.....	60
Tracking	62
Automated Tracking Events	63

Custom Tracking Messages	70
Custom Tracking Events	71
Surveys	72
Dynamic Presentation Customization	75
Briefcase Assets	76
Testing and Deployment	77
Content Templates.....	78
Content Checklist.....	79
SystemBridge Function Reference	79

About this Document

This Content Development Guide is designed as a technical resource for developers of presentation content for the AstraZeneca Harmony Viewer (HarVie) iPad application. It provides technical information about the structure and capabilities of the Harmony Viewer framework, as well as suggested best practices for iDetail presentation content development. All development standards promoted in this document should be adhered to unless AstraZeneca officially sanctions deviations based on specific market needs.

Harmony Viewer iDetail content is primarily composed of HTML5 conformant HTML. This document is not intended to provide generalized instruction for HTML, CSS, or JavaScript development, but it does include relevant sample code snippets where appropriate to highlight framework functionality. The guidance provided in this document is relatively agnostic of particular development environments and tools, although Apple OSX and iOS environments do provide some workflow advantages. Content creators are assumed to possess prior knowledge of industry standard tools to design, develop, configure and deploy HTML content.

This document should be used in conjunction with the “AstraZeneca iDetailing Content Style Guide.” As such, this guide does not discuss the design or creation of multimedia assets, user interface elements, screen layouts, or other largely aesthetic concerns. Instead, this document provides guidelines as to how these design elements can be most effectively integrated into a highly interactive, responsive, and effective iDetail.

The content included in this guide is compulsory and the required guidelines are designed to meet and fulfill AstraZeneca objectives with regard to content development and reuse. AstraZeneca evaluates agencies based on application of these guidelines and adherence to development methodology; as such, deviation from the Content Development Guide should be previously agreed with the central or local team and always be communicated to the central team.

Harmony Viewer Platform

The Harmony Viewer is an iPad app used by pharmaceutical sales representatives in the field. Reps primarily use the application to present interactive content and supplemental data during live calls with health care professionals.

The Harmony Viewer's presentation platform is built using a combination of high-performance native Obj-C code and flexible web presentation technologies. The majority of the graphical content that is presented to the end user is created using a combination of HTML5, CSS3, and JavaScript because it is visually flexible, creatively robust, and simpler to develop.

The Harmony Viewer supports all web technologies that are supported by the iPad's Mobile Safari web browser. This includes almost all common web based file formats like HTML, mp4 video, PDF, and various image formats; however, no Flash content is supported on iPad devices.

Since web based content doesn't always perform optimally under the hardware constraints of mobile devices, certain key performance intensive features are instead developed using custom native Obj-C code. In this way, the Harmony Viewer can display highly engaging content, while retaining a high level of performance that is necessary for the sales environment encountered by AstraZeneca's sales representatives. These native modules include custom gesture based navigation schemas, optimized dynamic content loaders, a highly customizable persistent navigation overlay, a robust PDF Viewer, multiple video controllers, and various supporting back end data management libraries. Many of these native modules are accessible to content creators through our SystemBridge JavaScript library.

The Harmony Viewer's presentation functionality is bolstered by the HarVie Publisher, a server based backend framework that provides user/group management, content deployment, and interaction tracking support. Additional healthcare professional data management and HCP specific customization of presentation content is possible through Harmony Viewer integrations with external sales force automation applications.

Anatomy of a Presentation

Mobile devices, like the iPad, are quite processor and memory limited in comparison to current desktop and laptop computer systems. With this in mind, the Harmony Viewer's presentation system has been designed to minimize memory and processor footprint, while maximizing creative flexibility.

HTML Slides

Harmony Viewer presentations are built in a very modular way. Individual content screens, called "slides," are created and stored as distinct HTML files. This allows the Harmony Viewer to dynamically load individual slides only when they are useful to the user and/or required by the prescribed presentation configuration rules. Likewise, the Harmony Viewer can dynamically unload slides when they become unnecessary or are blocked by presentation configuration rules, thereby maintaining a small memory footprint at all times, regardless of the overall size of the presentation.

Navigation Layer

Although the HTML based presentation screens form the majority of the presentation, an additional user interface layer, called the Navigation Layer, sits on top of this content. The Navigation Layer is intended to house all of the persistent screen elements for the presentation. Typically this includes functional elements like navigation or menu buttons, as well as purely aesthetic branding elements.

Due to technical limitations on the iPad, the Navigation Layer functionality could not be achieved entirely with HTML. Instead, a custom Navigation Layer class was created using native Obj-C code. This class contains a user interface template for xCode's Interface Builder (Apple's Obj-C software development environment), which allows the Navigation Layer's graphics to be fully customized. A single default Navigation Layer skin is embedded in the Harmony Viewer app, but additional skins can be supplied along with presentation content. In this way, every single Harmony Viewer presentation can have a completely unique user interface, with the performance and capabilities of native code, without actually having to write any Obj-C code.

Supplemental Media Controllers

Most of the visual and interactive content elements displayed by the Harmony Viewer framework are HTML, CSS, and JavaScript based. In some cases, these HTML based interactive elements need additional support from native Obj-C code to accomplish their desired behaviors due to the functional and performance limitations of HTML based views. To facilitate better performance, several supplemental media controllers have been incorporated into the Harmony Viewer application. Many of these are modal dialog boxes that can be launched directly from within HTML content using minimal JavaScript integration code. Notable examples include:

- **Pop-Up:** Allows a full screen HTML asset to be launched on top of the existing Harmony Viewer content.
- **PDF Viewer:** Allows PDF content to be launched in a full screen modal window above the existing Harmony Viewer content. Offers a more robust user experience than the standard web based PDF view offered on the iPad (UI is similar in functionality to the iBooks PDF Viewer).
- **Fullscreen Web Viewer:** Allows HTML content or external web page content (if an active internet connection is available) to be launched in a full screen modal window above the existing Harmony Viewer content.
- **Briefcase:** Allows reps to preview supplemental content (PDFs, videos, etc.) with an HCP. This supplemental content can be emailed to the HCP at their request.

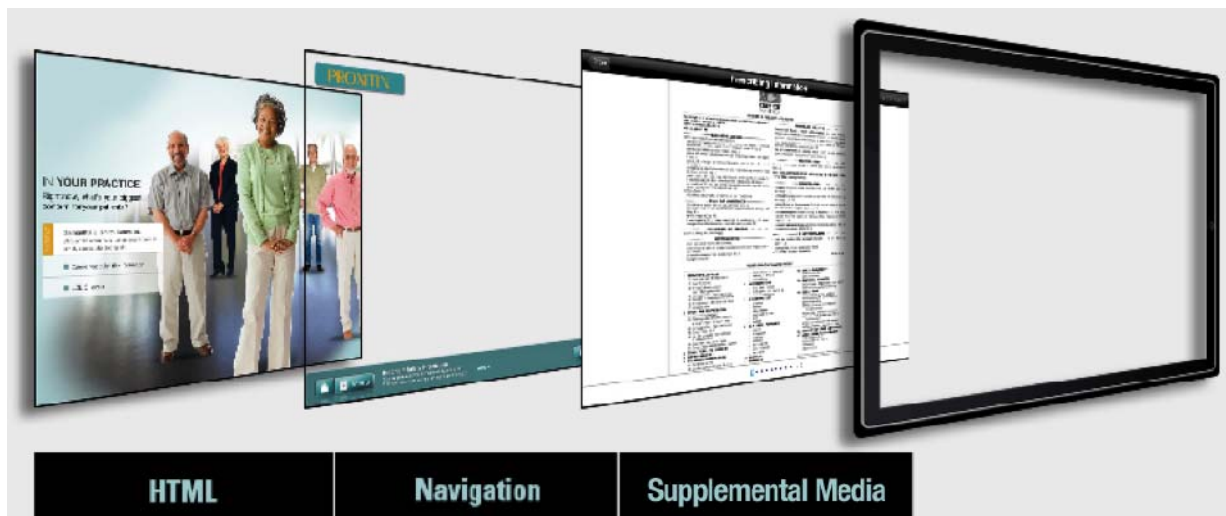


FIGURE 1.0: Diagram showing the visible “layers” of content that comprise a Harmony Viewer presentation – HTML Slide, Navigation Layer, Supplemental Media Controller (PDF Viewer)

External Data Assets

Data assets are an additional unstructured modular asset type supported by the Harmony Viewer. Data asset directories can contain any arbitrary files (PDF, video, HTML, XML, etc.). They are not accessed directly by the default Harmony Viewer functionality. For these data assets to be useful, they must be accessed and used by a separate slide asset.

Data assets are primarily intended to eliminate redundant file data by enabling file sharing across multiple presentations or slides, but can also be leveraged to dynamically populate presentation content.

Presentation Assets

The presentation asset is the root asset defining a single Harmony Viewer presentation. A presentation asset must minimally contain a configuration file that defines the overall structure of a presentation.

For particularly small presentations, the presentation asset directory can contain all assets necessary to display the presentation (HTML, images, etc.), but in the interests of content reuse, this strategy should be minimized. Nearly all presentation configurations should instead refer heavily to external slide, navigation layer, and data assets.

File Structure and Naming Conventions

Modular Asset Library

All file assets required to display Harmony Viewer presentations are stored in the app's Library/Content/iVisual directory. The iVisual directory adheres to the structure described in "Anatomy of a Presentation" and, as such, contains individual asset subdirectories for each slide, navigation layer, external data, and presentation asset.

Depending on the nature of a presentation's implementation, each "asset" directory can be as large as a fully self-contained presentation (e.g. one large presentation asset containing many files) or as small as a single file (e.g. a data asset containing one media file). Since the "asset" directory is the smallest level of content download currently afforded by the Harmony Viewer framework, it is recommended that presentations be logically divided into several different asset directories. The optimal level of asset granularity may vary widely from presentation to presentation, but some general guidelines are:

- Each presentation should have a single "Presentation Asset". This directory should contain a presentation configuration file that refers almost entirely to external "Slide Assets", "Navigation Layer Assets", and "Data Assets".
- Each slide of content in the presentation should be implemented as a self-contained "Slide Asset".
- Each presentation should either refer to an existing "Navigation Layer Asset" that is shared between many presentations or a single presentation specific "Navigation Layer Asset".
- Large file or data assets that are used by several different presentations or slides should be isolated into their own unique "Data Assets" (e.g. if several slides include a link to the same large PDF file, that PDF file should be placed in its own "Data Asset" where it can be shared between the relevant slides).

The internal structures and necessary configuration properties of each asset type are detailed in subsequent sections of this document.

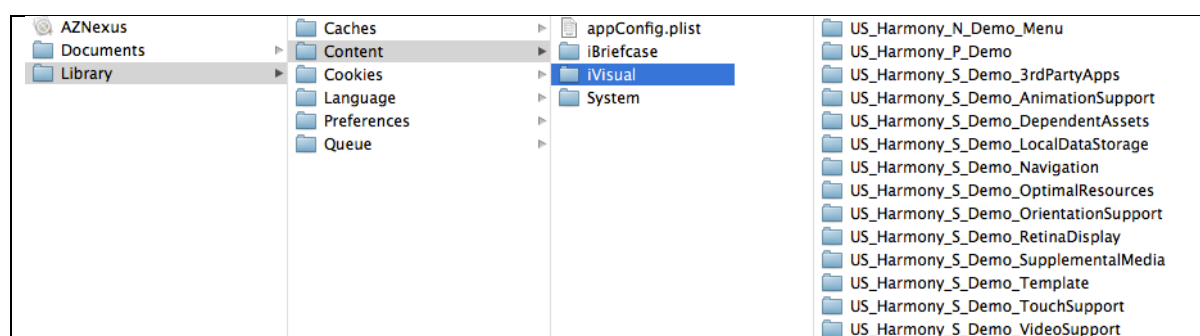


FIGURE 2.0: Sample presentation file storage structure comprised of one Presentation Asset (US_Harmony_P_Demo), one Navigation Layer Asset (US_Harmony_N_Demo_Menu), and multiple Slide Assets (all assets with the US_Harmony_S_ prefix).

Naming Conventions

The base directory of any presentation, slide, navigation layer, or data asset within the Harmony Viewer's Content/iVisual directory corresponds to a unique AssetID through which other system assets may reference or access the asset's subcontents. The same AssetID is used throughout the asset's lifecycle, from its initial upload to the HarVie Publisher server, through its distribution to a specific user's iPad. Since all presentation assets deployed to an individual iPad are stored in the same iVisual directory, each asset must be uniquely named to prevent naming conflicts when multiple presentations are deployed to the same iPad.

Any naming conflicts identified upon initial content upload to the HarVie Publisher will constitute rejection of the asset. Any asset rejected due to a naming conflict must be renamed. All external references to that asset must also be updated to reflect the updated name. To minimize the potential for asset naming conflicts and to improve asset sharing across presentations and markets, it is important that all agencies adhere to a global asset naming convention.

Since all asset types used by the Harmony Viewer framework coexist in the same asset library directory, the unique naming convention can be applied directly to the asset package/asset root directory name according to the following formula:

- Country Code_Language_Brand_Asset Type_Unique Identifier

Key Property	String Format	Example Value
Country Code	[2 characters]	BE
Language Code	[3 characters]	FRA
Brand	[Free text]	BRILINTA
Asset Type	[1 character]	P
Unique Identifier	[Free text]	PharmacistAlpha
Example Result: BE_FRA_BRILINTA_P_PharmacistAlpha		

Notes:

- Language code is only required in regions where multiple languages are supported.
- Unique Identifier property must remain unchanged for the lifetime of the asset. As such, this should not include a version or date because these values could change over time as the asset is routinely updated. Version and update information will be maintained by the system through metadata that is independent from the AssetID formed via this naming convention.

VALID KEY PROPERTY VALUES

Country and language codes used for asset naming should adhere to the recognized ISO standards that can be on the ISO.org website:

- http://www.iso.org/iso/home/standards/country_codes.htm
- http://www.iso.org/iso/home/standards/language_codes.htm

Asset types used for asset naming should adhere to the following values:

Asset Type	Code
Presentation	P
Navigation Layer	N
Slide	S
Data	D

Presentation Slides

The primary building blocks of Harmony Viewer presentations are the individual content screens, called “slides.” Each slide asset within the Harmony Viewer framework exists as a uniquely named, self-contained content directory. This content directory should minimally contain the slide’s primary media file (HTML, PDF, or video file) and a slide configuration file (config.plist).

To simplify modular slide reuse across multiple presentations, the slide’s content directory should also contain any data, source code, or art files upon which the slide relies (CSS, JavaScript, images, etc.). It is common practice to group related file types into appropriately named subdirectories of the slide’s root directory (CSS files in a “css” subdirectory, image files in an “images” subdirectory, etc.), but this is not necessary to achieve proper functionality of the slide.

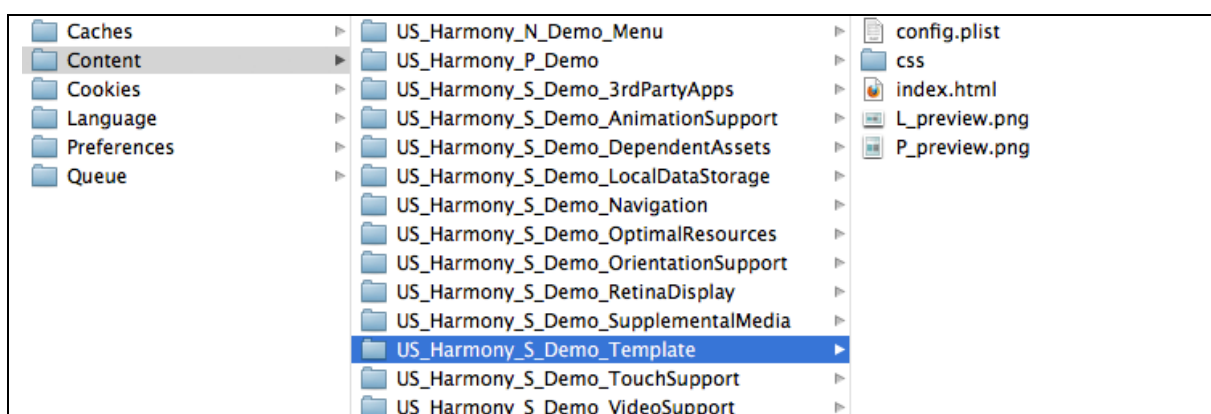



FIGURE 3.0: Sample file structure for a screen asset

Due to the performance and stability problems inherent in relying on potentially unstable Internet connections, all Harmony Viewer presentations are performed in offline mode, using content files available in the app’s local library. Consequently, presentation slides should never require access to online resources for proper display or functionality.

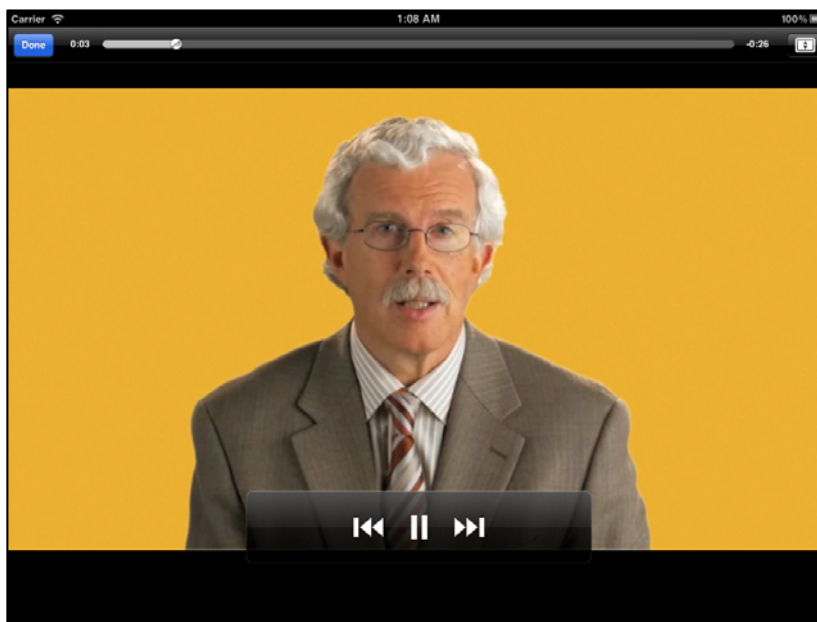
Modal Slides

Most slides used in Harmony Viewer presentations are HTML screens that are displayed at the lowest “layer” of the application’s interface hierarchy (beneath the navigation layer and any supplemental media controllers), leaving the navigation layer available for user interaction and gesture based navigation to adjacent slides.

To simplify content reuse, the Harmony Viewer also supports several modal display options for slide content. These modal views are presented above the application’s navigation layer and consequently prevent user interaction with the underlying presentation content until they are closed. Modal slides cannot be included in the presentation’s Active Call (discussed in section “8.2 Active Call StoryFlow”), but provide a simple mechanism for launching modal content that is referenced from many other slides in a presentation or from a global button in the presentation’s navigation layer (e.g. Prescribing Information). The available modal slide controllers include:

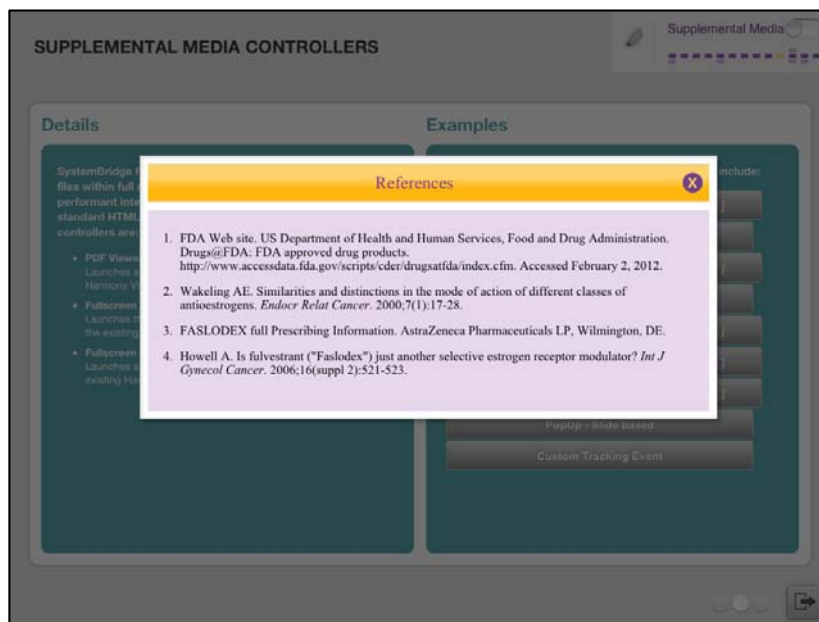
Modal Controller	Description
<p>PDF Viewer</p>	 <ul style="list-style-type: none"> • Launches a full screen PDF Viewer dialog on top of the existing Harmony Viewer content. • The specified title is displayed in the dialog's header. • A single tap toggles the visibility of the dialog's header and footer controls. • Horizontal swipe gestures navigate between pages in the document. • Pinch zoom gestures can be used to zoom in and out of PDF page content. • A page scrubber control is available at the bottom of the dialog where touch events can be used to quickly preview and navigate to other pages of the PDF document. • A "Table of Contents" button at the upper right corner of the screen displays a list of PDF page links if corresponding bookmarks are included in the PDF document. Bookmarks specified in Acrobat prior to the deployment of the PDF will automatically populate the "Table of Contents" pop-up dialog. • Regardless of the orientation support of the Harmony Viewer presentation, the PDF Viewer supports both landscape and portrait orientations.

Fullscreen Video Viewer



- Launches the standard iOS full screen video controller on top of the existing Harmony Viewer content.

HTML PopUp



- Launches a layer of HTML on top of the existing Harmony Viewer content with a transparent background.
- Useful for a multitude of popup effects when it is preferable to store the popup HTML separately from the source slide's HTML (e.g. if popup content is sufficiently dense to negatively impact the loading time of the slide HTML).
- Multiple layers of popups can be launched on top of one another.
- Popup HTML must include SystemBridge JavaScript code (`closeTopPopUp`) to dismiss the popup.

Slide Configuration File

Slide assets must include a “config.plist” file in their root directory. The slide asset config file lists specific properties that control how the slide is presented to the user during the presentation, as well as information about the slide that may be used to populate other features in the Harmony Viewer user interface or backend data systems.

Key	Type	Value
▼ Root	Dictionary	(7 items)
Title	String	Template
URL	String	index.html
Landscape Preview	String	L_preview.png
Portrait Preview	String	P_preview.png
Thumbnail Preview	String	L_preview.png
Delay Display	Boolean	NO
Menu Visibility	String	Active

FIGURE 4.0: Sample slide asset configuration file

The parameters available within in slide configuration file include:

Key Property	Description
Title	<p>(Required) String</p> <ul style="list-style-type: none"> • A unique plain text title for this slide. • The title is often used by the Harmony Viewer to refer to this screen content when doing so via absolute URL would be awkward (e.g. in a presentation active call configuration or a SystemBridge goToSlide execution). • Slide title is also used to populate certain Harmony Viewer user interface elements (e.g. breadcrumb, thumbnail menu, pre-call slide sorter, etc.).

URL	<p>(Required for standard non-modal slides) String</p> <ul style="list-style-type: none"> • Used only for slides of the standard non-modal asset type. Should not be included for modal slides (PDF, Video, or PopUp). • The local path, relative to the slide's root directory, of the HTML file that should be displayed for this content screen. • HTML slides of this type are displayed in the standard manner (e.g. in a fullscreen HTML view, beneath the presentation's navigation layer). • Only HTML slides that use the URL parameter are eligible for inclusion in the presentation's Active Call, which defines the order in which presentation slides can be accessed via swipe gestures. More detailed information about the Harmony Viewer's Active Call functionality can be found in the "8.0 Presentation" section. • Most slides in a presentation use the URL mode of display.
PDF	<p>(Required for PDF Viewer modal slides) String</p> <ul style="list-style-type: none"> • Used only for slides of the PDF Viewer modal type • The local path, relative to the slide's root directory, of the PDF file that should be displayed for this content screen. • PDF slides are displayed in the Harmony Viewer's PDF Viewer, on top of the existing presentation content. • PDF slides cannot be included in the presentation's Active Call. • PDF slides provide a simple mechanism for launching PDF content that is referenced from many other slides in a presentation or from a global button in the presentation's navigation layer (e.g. Prescribing Information).
Video	<p>(Required for Fullscreen Video modal slides) String</p> <ul style="list-style-type: none"> • Used only for slides of the Fullscreen Video modal type • The local path, relative to the slide's root directory, of the video file that should be displayed for this content screen. • Video slides are displayed in the standard iOS fullscreen video controller, on top of the existing presentation content. • Video slides cannot be included in the presentation's Active Call. • Video slides provide a simple mechanism for launching video content that is referenced from many other slides in a presentation or from a global button in the presentation's navigation layer (e.g. Prescribing Information).

PopUp	<p>(Required for HTML PopUp modal slides) String</p> <ul style="list-style-type: none"> • Used only for slides of the HTML PopUp modal type • The local path, relative to the slide's root directory, of the HTML file that should be displayed for this content screen. • HTML slides of this type are displayed in an HTML layer that sits on top of the existing presentation content. • PopUp slides cannot be included in the presentation's Active Call. • PopUp slides provide a simple mechanism for launching HTML popup content that is referenced from many other slides in a presentation or from a global button in the presentation's navigation layer (e.g. Important Safety Information).
Portrait Preview	<p>(Optional) String</p> <ul style="list-style-type: none"> • The local path, relative to the slide's root directory, of the image file (PNG or JPG) that should be displayed while the system is loading the slide's corresponding HTML content. • The portrait preview will be used only when the iPad device is in portrait orientation. As such, presentations that don't support portrait orientation need not include this. • To provide a seamless transition from the slide's preview image to its corresponding HTML, the preview image should be identical to the HTML content's initial fully loaded state (e.g. if the slide contains an animation, the preview image should match the first frame of the animation). • If a portrait preview image is not included in the slide content, the application's users may see short flashes of white screen while the slide's HTML content is loading.

Landscape Preview	<p>(Optional) String</p> <ul style="list-style-type: none"> • The local path, relative to the slide's root directory, of the image file (PNG or JPG) that should be displayed while the system is loading the slide's corresponding HTML content. • The landscape preview will be used only when the iPad device is in landscape orientation. As such, presentations that don't support landscape orientation need not include this. • To provide a seamless transition from the slide's preview image to its corresponding HTML, the preview image should be identical to the HTML content's initial fully loaded state (e.g. if the slide contains an animation, the preview image should match the first frame of the animation). • If a landscape preview image is not included in the slide content, the application's users may see short flashes of white screen while the slide's HTML content is loading. If a separate "Thumbnail Preview" is not included in the slide content, the landscape preview image will also be used to populate some Harmony Viewer user interface elements (e.g. thumbnail menu, pre-call slide sorter, etc.).
Thumbnail Preview	<p>(Optional) String</p> <ul style="list-style-type: none"> • The local path, relative to the slide's root directory, of the image file (PNG or JPG) that should be used to populate the Harmony Viewer's thumbnail representations of this slide content (e.g. thumbnail menu, pre-call slide sorter, etc.). • If a thumbnail preview is not included in the slide content, the landscape preview image will be used to populate any thumbnail based user interface.

Delay Display	<p>(Optional) Boolean</p> <ul style="list-style-type: none"> • Default value = NO • NO = HTML content for this slide will be displayed as soon as the standard iOS system events report that it has finished loading • YES = HTML content will not be displayed until the slide's HTML explicitly reports that it is ready for display using the JavaScript function <code>SystemBridge.pageLoadComplete()</code> (see <code>SystemBridge</code> reference for further detail). • This option is useful for screens that have additional media loading or JavaScript initialization tasks that extend beyond the standard system loading events (e.g. delay display of the page until a large video is loaded and ready to play). • If HTML display is delayed using this mechanism, the slide's preview image will remain on screen throughout the delay.
Menu Visibility	<p>(Optional) String</p> <ul style="list-style-type: none"> • Valid values = Active, Inactive, Hidden • Default value = Active • This property controls the display and functionality of this slide's entry in the Harmony Viewer's thumbnail menu. • Active = This slide's thumbnail will be included in the thumbnail menu and will function as an active button by which the user can navigate to this slide. • Inactive = This slide's thumbnail will be included in the thumbnail menu, but it will NOT function as an active button by which the user can navigate to this slide. This option is useful for limiting the access points to certain presentation content. • Hidden = This slide's thumbnail will not be included in the thumbnail menu at all. This option is useful for eliminating redundant entries in the thumbnail menu (e.g. the Prescribing Information slide already has a navigation layer button devoted to it).

HTML Guidelines

Harmony Viewer slide layout and functionality is implemented almost entirely in standard HTML code. The Harmony Viewer supports all web technologies that are supported by the iPad's Mobile Safari web browser. This includes almost all common web based file formats like HTML, mp4 video, PDF, and various image formats; however, no Flash content is supported on iPad devices.

Though standard HTML design practices apply to Harmony Viewer slides, certain design and coding practices are encouraged to achieve the best possible system performance. To

further optimize presentation performance, the Harmony Viewer also provides HTML and/or JavaScript access to supplemental native functionality that is exclusive to the Harmony Viewer framework.

This document is accompanied by sample HTML code for all features described in this section.

Base HTML Template

The base template for slide HTML files has the following characteristics:

- **Properly sized to fit the device dimensions**

The native screen resolution of all iPad devices, as it relates to HTML development, is 1024x768 pixels. iPad models with the higher resolution retina display (2048x1536 pixels) still respond to HTML code as if their screen resolution is 1024x768 pixels. As such, it is recommended that all HTML slides set the dimensions of the document's body element to match the 1024x768 pixel dimensions.

If the HTML page dimensions do not match the device dimensions, the page will still be displayed. Finger gestures may be used to resize or pan the slide content, just as a user would expect from standard iOS web browser behaviors; however, there are clear user experience advantages to maintaining a 1-to-1 relationship between the device's native screen dimensions and the designed dimensions of individual HTML slides.

CSS media queries can be used to properly size the body element automatically based on the device's current user interface orientation (portrait vs. landscape).

```
body {
    margin:0;
    padding:0;
    position:relative;
    font-family:Helvetica Neue, Verdana, sans-serif;
    font-size:14px;
    overflow:hidden;
}

@media only screen and (device-width:768px) and (orientation:landscape)
{
    body { width:1024px; height:768px; }
}

@media only screen and (device-width:768px) and (orientation:portrait)
{
    body { width:768px; height:1024px; }
}
```

FIGURE: Sample CSS code for properly sized HTML body

- **Pinch zoom disabled**

The standard iOS pinch zoom gestures are supported in Harmony Viewer slide HTML, though its use is generally discouraged. Generic pinch zoom and panning gestures were primarily developed to allow mobile devices to compensate for the problems inherent in viewing web content that was designed for desktop systems (e.g. content layout that is too dense to read on a mobile screen). It is generally preferred for Harmony Viewer slide content to be specifically designed for the iPad form factor; so generic pinch zoom gestures should not be necessary. Disabling pinch zoom gestures also prevents gesture confusion on pages that include custom interactive content that relies on touch event processing.

Page wide pinch zoom gestures can be enabled or disabled from within each individual HTML slide via the parameters of the “viewport” meta element in the HTML head element.

```
<html>
<head>
  <meta name="viewport" content="width=device-width; initial-scale=1.0; maximum-scale=1.0;"/>
</head>
<body>
</body>
</html>
```

FIGURE: Disable pinch zoom by restricting the scale parameters of the viewport meta element

```
<html>
<head>
  <meta name="viewport" content="width=device-width">
</head>
<body>
</body>
</html>
```

FIGURE: Enable pinch zoom by eliminating the scale parameters or specifying a maximum-scale > 1.0 in the viewport meta element

- **References local file resources**

As previously discussed, each slide asset within the Harmony Viewer framework exists as a uniquely named, self-contained content directory. To simplify modular slide reuse across multiple presentations, the slide’s content directory should also contain all data, source code, or art files upon which the HTML file relies (CSS, JavaScript, images, etc.). It is common practice to group related file types into appropriately named subdirectories of the slide’s root directory (CSS files in a “css” subdirectory, image files in an “images” subdirectory, etc.), but this is not imposed or enforced by the Harmony Viewer framework. There are a small number of use cases in which it is advantageous for an HTML slide to reference data that is external to its

own asset directory (see section 9.0 “Presentation Independent Data Assets”), but the vast majority of slide specific file resources should be stored locally and referenced in the HTML code via relative file paths.

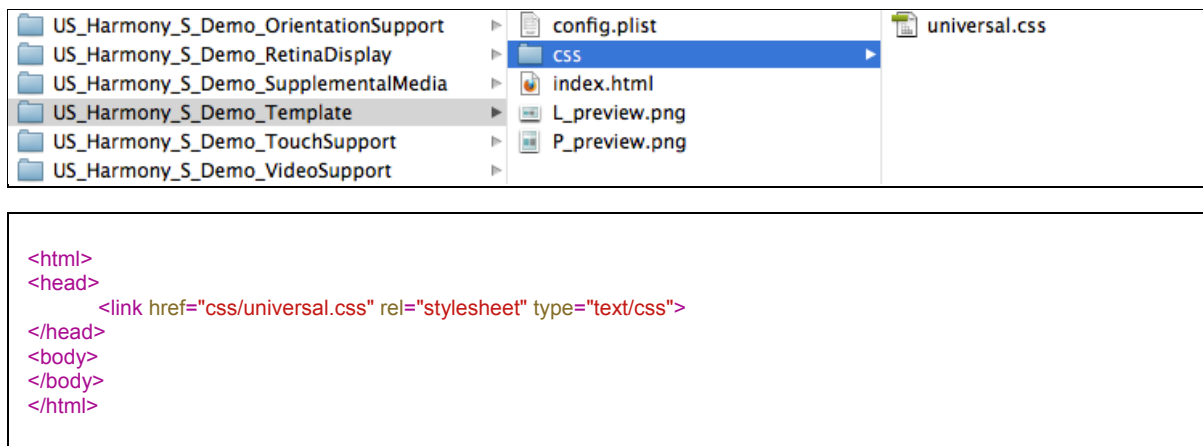


FIGURE: Store slide specific files in local slide directory and reference via relative file paths

- **Responds properly to slide lifecycle events**

The lifecycle for an HTML slide in the Harmony Viewer is slightly different from that of a conventional web browser. To improve presentation performance, the Harmony Viewer framework routinely performs some preloading and caching of HTML slides that are likely to be displayed soon. To allow content creators maximal control over the initialization and display of each screen, particularly those with interactive or animated content, the Harmony Viewer automatically executes a couple of additional JavaScript based lifecycle events, notifying the HTML slide when it appears on screen and later when it disappears from view.

Event	Description
DOMContentLoaded	(Optional) Standard HTML Event <ul style="list-style-type: none"> • Triggers when the page HTML and scripts have completed loading and are ready to execute • Images and other file assets may still be loading • CSS styles may not yet be applied • May occur when slide is not on screen • Do not automatically initiate animation or video playback with this event

onload	<p>(Optional) Standard HTML Event</p> <ul style="list-style-type: none"> • Triggers when the HTML page is fully loaded, including all dependent images and styles • May occur when slide is not on screen • Do not automatically initiate animation or video playback with this event
gotFocus	<p>(Optional) Harmony Viewer Exclusive Event</p> <ul style="list-style-type: none"> • Triggers when the HTML page becomes visible on screen • Executed after DOMContentLoaded event if HTML page is still loading when it appears on screen • Useful for initiating automatic playback of video or animated content
lostFocus	<p>(Optional) Harmony Viewer Exclusive Event</p> <ul style="list-style-type: none"> • Trigger when the HTML page has shifted offscreen • All relevant lostFocus events are executed before any gotFocus events are executed • Useful for automatically pausing video or resetting interactive or animated content

```

<html>
<head>
  <script>

    function initialize() {}
    document.addEventListener('DOMContentLoaded', initialize);

    onload = function() {}

    function gotFocus() {}

    function lostFocus() {}

  </script>
</head>
<body>
</body>
</html>

```

FIGURE: JavaScript function stubs for each stage of the HTML lifecycle (all functions are optional)

Optimal Font Utilization

In the course of standard HTML development, several methods are commonly used for depicting text:

- **Live text using system provided fonts**

HTML text is assigned font characteristics from those that are supported by the target platform (iOS) without the need for any additional supporting font files. This is the most common strategy and, due to optimizations that the native text rendering system provides, the best performing technique. For optimal HTML rendering performance, this method should be used for the vast majority of slide text.

```
body {
  font-family:Helvetica Neue, Verdana, sans-serif;
  font-size:14px;
  color:#555;
}
```

FIGURE: Sample of standard CSS font characteristics

- **Live text using custom fonts**

The standard iOS HTML rendering infrastructure supports the use of custom TTF and SVG fonts (TTF fonts load slightly faster). HTML text is assigned font characteristics similar to those provided for system fonts, but a local font definition file provides the text rendering rules. This provides a great deal of flexibility in the use of fonts, but there are significant performance disadvantages to using custom fonts. The font rendering process required to generate a text block with a custom font is significantly slower than the same process with a standard system font.

If custom fonts are used for large amounts of text on an HTML slide, this often results in a noticeable delay before text blocks appear on screen. Consequently, custom fonts should be used sparingly, if at all, in Harmony Viewer slides (e.g. slide titles or text associated with unique graphic elements). It is strongly recommended that iOS system fonts be used for nearly all HTML text, relegating custom font usage to only branded header lines or titles.


```
@font-face {
  font-family: 'FrutigerBoldCondensed';
  src: url('../fonts/frutigerlstd-boldcn-webfont.ttf')format('truetype');
  font-weight: normal;
  font-style: normal;
}

.customFont {
  font-family:'FrutigerBoldCondensed';
  font-size:22px;
  line-height:26px;
}
```

FIGURE: Sample of CSS custom font characteristics

- **Image based text**

It is not uncommon for text blocks to be pre-rendered into static image files for insertion into HTML image elements in lieu of live styled text. Historically, there were some legitimate rationales for resorting to this technique; however, the current iOS HTML rendering system provides enough alternative font styling techniques to render the practice obsolete. Custom font support, as discussed above, allows nearly any font design to be leveraged with live HTML text. Other CSS properties can be leveraged to control a multitude of other common text effects including shadowing, complex coloring, and 2D/3D transformations.

Image based text also suffers some significant disadvantages over the live text alternatives. Image based text is much more difficult to edit and maintain over time as content evolves. Image based text also requires that a disproportionate number of file resources be used for optimal display of content. If the text block layout differs between portrait and landscape orientation, both versions of the image must be created and maintained. If the presentation content is targeted to support both the retina and conventional displays, multiple resolutions of the image must also be created and maintained.

IMAGE BASED TEXT SHOULD NEVER BE USED IF ANY ALTERNATIVE EXISTS.

Retina Display Support

The native screen resolution of all iPad devices is 1024x768 pixels, but newer iPad models have a higher pixel density retina display (2048x1536 pixels). HTML content can take best advantage of this extra display resolution by intelligently applying resolution independent text and graphics techniques and explicitly providing multi-resolution versions of any raster images.

Resolution Independent Techniques

Text blocks rendered using standard system fonts or custom embedded fonts will automatically utilize the proper pixel density for the current display device. The same is true of CSS generated graphical effects like drop shadows, rounded corners, 2D and 3D transformations, etc. No additional code or file resources are required to fully utilize higher

resolution displays with these techniques, so content developers should take advantage of these options whenever possible.

Multiple Resolution Raster Images

Any text or graphical elements that rely on raster images will NOT automatically scale gracefully for higher resolution displays. If HTML content is developed specifically for the lower density displays, this content will likely appear slightly blurred when viewed on higher density Retina displays. If raster images must be used in content, the content should include multiple resolution versions of each raster image (a normal resolution image for use on non-retina displays and a double resolution image for use on double pixel density retina displays). There are several techniques available for dynamically selecting the proper resolution raster image based on the current display characteristics.

CSS media queries can be used to detect when an iPad with retina display is being used to display HTML content. In response, the CSS properties can specify a higher resolution image (2x) for use as the background image for HTML elements.

```
.retinalImage {
    background-image:url('../images/retina.png');
    background-size:100%;
    width:87px;
    height:87px;
}

@media only screen and (-webkit-min-device-pixel-ratio: 2)
{
    .retinalImage {
        background-image:url('../images/retina-2x.png');
    }
}
```

FIGURE: Sample of CSS based background image resolution detection

If the HTML content uses inline images (standard tags), content creators can rely on the device's built in image scaling. If a double resolution raster image source is specified, but the normal resolution image's dimensions are specified, the image will be properly scaled for display on both retina and non-retina displays. Standard web developers frown on this practice because it forces users of non-retina displays to download larger image files than are strictly necessary for their displays, but since Harmony Viewer content is always presented using content in a local repository, any negative performance impact is negligible.

```
<html>
<body>
    
</body>
</html>
```

FIGURE: Sample of automatically scaled by specifying a double resolution source image (174x174 pixels) with a normal resolution width and height (87x87 pixels)

Orientation Support

The Harmony Viewer supports both landscape and portrait interface orientations. Content creators can choose to enable or disable any of the supported orientations to suit the needs of the individual presentation. If presentation content is intended to support both device orientations, CSS media queries should be used to dynamically manipulate the HTML layout for each orientation.

```
body {
    margin:0;
    padding:0;
    position:relative;
    font-family:Helvetica Neue, Verdana, sans-serif;
    font-size:14px;
    overflow:hidden;
}

@media only screen and (device-width:768px) and (orientation:landscape)
{
    body { width:1024px; height:768px; }
}

@media only screen and (device-width:768px) and (orientation:portrait)
{
    body { width:768px; height:1024px; }
}
```

FIGURE: Sample CSS code for properly sized HTML body based on orientation media query

If CSS based transformations aren't sufficient for orientation based HTML manipulations, the JavaScript event "onorientationchange" can be used to trigger any additional HTML manipulations via JavaScript code. The "onorientationchange" event is triggered whenever the iPad device changes orientation. Within the "onorientationchange" function, the "window.orientation" property can be used to determine the current device orientation and respond appropriately.

```
<html>
<head>
    <script>
        onorientationchange = function() {
            if (window.orientation == 90 || window.orientation == -90) {
                // landscape orientation
            }
            else {
                // portrait orientation
            }
        }
    </script>
</head>
<body>
</body>
</html>
```

FIGURE: Sample JavaScript "onorientationchange" function implementation

Touch Event Support

The Harmony Viewer supports all of the standard iOS HTML touch events (ontouchstart, ontouchmove, ontouchend, ontouchcancel). More information about these events and how to detect, process, and respond to them can be found in Apple's official Safari Web Content Guide: Handling Events

(<http://developer.apple.com/library/ios/#DOCUMENTATION/AppleApplications/Reference/SafariWebContent/HandlingEvents/HandlingEvents.html>). Though much of this topic falls outside the scope of this guide, there are a couple of related techniques that are widely used in Harmony Viewer content.

Removing onClick Delay

Standard HTML development practice is to use “onclick” JavaScript events to respond to user interaction with button like elements. “onclick” events can likewise be used on touch based iOS devices to respond to finger taps. Unfortunately, since iOS devices respond to many other types of touch events (double taps, swipes, pinch zooms, etc.), there is about a 300ms delay in the execution of “onclick” events while the system verifies that the original tap wasn't intended to begin some more complicated gesture. This delay is acceptable in a web browser, but can unnecessarily hamper the perceived interactivity and responsiveness of HTML content in an application like the Harmony Viewer.

Some content developers solve this problem by replacing the offending “onclick” events with “ontouchstart” events. This successfully removes the 300ms delay before execution, but since the event executes as soon as the user's finger touches the screen, it can have unintended side effects. Particularly in the Harmony Viewer, where swipe gestures are used to navigate between pages, awkwardly placed “ontouchstart” events can hamper navigation. **Consequently, “ontouchstart” events should NOT replace “onclick” events in Harmony Viewer content.**

Mobile developer Matteo Spinelli solved the click delay problem in a more elegant and effective way. He developed a third party JavaScript helper library, NoClickDelay, that more effectively processes the available touch events and executes the “onclick” event in response to a more appropriate occurrence of “ontouchend”. The details of his solution and the JavaScript source code can be found on his web site (<http://cubiq.org/remove-onclick-delay-on-webkit-for-iphone>). It is strongly encouraged that all Harmony Viewer content use NoClickDelay or some similar script to expedite “onclick” execution.

Since NoClickDelay is so often used in Harmony Viewer slides, a version of the script has been embedded in the standard Harmony Viewer application distribution. The embedded JavaScript file can be loaded by an HTML slide using the “resource://js/noClickDelay.js” file path.

```
<html>
<head>
  <script type="text/javascript" src="resource://js/noClickDelay.js"></script>

  <script>
    function initialize() {
      new NoClickDelay('expeditedLink');
    }
    document.addEventListener('DOMContentLoaded', initialize, false);
  </script>
</head>
<body>
  <li id="expeditedLink" onclick="alert('Expedited ONCLICK');">Expedited ONCLICK</li>
</body>
</html>
```

FIGURE: Sample NoClickDelay usage for expedited “onclick” event execution

Scrollable Page Content

It’s recommended that Harmony Viewer slide content be designed to fit the iPad screen dimensions, but some content doesn’t fit nicely within the screen bounds. Consequently, it’s beneficial to define one or more areas of the slide with self-contained scrollable content blocks. The default iOS HTML standards provide a mechanism for two fingered scrolling of HTML content blocks, but this is a much less intuitive user experience than a single finger scroll.

Once again, mobile developer Matteo Spinelli solved this content scrolling problem in a very effective way. He developed a third party JavaScript helper library, iScroll, which enables single finger scrolling of an HTML <div> element of fixed dimensions. The script also incorporates a parameter to set and customize a scrollbar, which helps to visually identify the content as scrollable. The details of his solution and the JavaScript source code can be found on his web site (<http://cubiq.org/iscroll-4>). It is strongly encouraged that all Harmony Viewer content use iScroll or some similar script to implement scrollable content areas within slide content.

Since iScroll is so often used in Harmony Viewer slides, a version of the script has been embedded in the standard Harmony Viewer application distribution. The embedded JavaScript file can be loaded by an HTML slide using the “resource://js/iscroll.js” file path.

```

<html>
<head>
  <script type="text/javascript" src="resource://js/iscroll.js"></script>

  <script>
    function initialize() {
      new iScroll('scrollWrapper');
    }
    document.addEventListener('DOMContentLoaded', initialize, false);
  </script>
</head>
<body>
  <div id="scrollWrapper">
    <div id="scroller">
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam eleifend placerat lorem,
quis bibendum tellus mattis sit amet. Ut hendrerit lobortis faucibus. Pellentesque id risus quis purus vulputate
fringilla. Sed ac felis lectus, eget blandit ipsum. Sed nec velit eu nulla pellentesque placerat eu ut odio. Vivamus
vitae nisi eu erat laoreet fermentum. Sed ut sem quis diam iaculis hendrerit. Donec aliquet quam in nulla
tristique sed vulputate metus accumsan. Nam nec turpis in erat vulputate pulvinar sed sit amet neque. Donec
purus ligula, volutpat ac placerat at, sollicitudin nec erat. Vestibulum lobortis, enim non eleifend gravida, augue
est porttitor ante, non eleifend nisl metus a eros. Sed eget quam lacus, in mattis elit. In hac habitasse platea
dictumst.</p>
    </div>
  </div>
</body>
</html>

```

FIGURE: Sample iScroll implementation of a scrollable content area

Video Support

HTML5 Video

The Harmony Viewer supports audio and video media via the standard HTML5 `<audio>` and `<video>` elements, as is supported by the Mobile Safari web browser on the iPad. Detailed information about the usage of these media elements can be found in Apple's Safari HTML5 Audio and Video Guide

(http://developer.apple.com/library/safari/#documentation/AudioVideo/Conceptual/Using_HTML5_Audio_Video/Introduction/Introduction.html).

File Formats and Compression Settings

Though iPad devices with retina displays technically support video with maximum size matching the screen's dimensions (2048x1536 pixels), it is recommended that Harmony Viewer content limit video to a resolution that can be viewed on all supported devices (e.g. maximum dimensions of 1024x768 pixels). Content creators should also consider the intended context of the video content when deciding on the compression standards to apply. If video content will comprise only a small section of a presentation screen, the video dimensions and compression settings can be reduced to shrink file size appropriately. Video that is intended to be viewed full screen must be encoded with less compression, potentially resulting in a much larger file.

The official supported video formats for iPad 2 and iPad (with retina display) are:

- H.264 video up to 1080p, 30 frames per second, High Profile level 4.1 with AAC-LC audio up to 160 Kbps, 48kHz, stereo audio in .m4v, .mp4, and .mov file formats
- MPEG-4 video up to 2.5 Mbps, 640 by 480 pixels, 30 frames per second, Simple Profile with AAC-LC audio up to 160 Kbps per channel, 48kHz, stereo audio in .m4v, .mp4, and .mov file formats
- Motion JPEG (M-JPEG) up to 35 Mbps, 1280 by 720 pixels, 30 frames per second, audio in ulaw, PCM stereo audio in .avi file format

More information about converting audio and video content into iPad supported formats can be found in Apple's iTunes Podcasting Resources (<http://www.apple.com/itunes/podcasts/specs.html#formattingvideo>).

Video Response to gotFocus and lostFocus Events

iPad devices support playback of only one audio or video file at a time. Content creators must ensure that Harmony Viewer slide content is designed accordingly. Since the Harmony Viewer performs some HTML page preloading and caching, it's possible for screens that contain audio or video files to be resident in memory without being visible on screen. As a result, it's imperative that HTML slides containing audio or video files appropriately respond to the system's "gotFocus" and "lostFocus" events. Audio/video playback should never be initiated before the page receives a "gotFocus" event. Likewise, audio/video playback should always be stopped when the page receives a "lostFocus" event.

```
<html>
<head>
  <script>
    function gotFocus() {
      document.getElementById('basicVideo').play();
    }

    function lostFocus() {
      document.getElementById('basicVideo').pause();
    }
  </script>
</head>
<body>
  <video id="basicVideo" src="media/KOL.m4v" preload controls width="433" height="243" />
</body>
</html>
```

FIGURE: Sample video playback control in response to gotFocus and lostFocus events

Animation Support

The Harmony Viewer supports animation in HTML via both content driven and code driven mechanisms. Content driven animation is supported using HTML5 video and animated GIFs. Code driven animation is supported via animated CSS transitions, CSS based keyframe animation, and the HTML5 canvas element. Flash content is in no way supported on iPad devices.

Detailed information about CSS based animation and visual effects can be found in Apple's Safari CSS Visual Effects Guide (<http://developer.apple.com/library/safari/#documentation/InternetWeb/Conceptual/SafariVisualEffectsProgGuide/Introduction.html>).

Detailed information about the HTML5 canvas element can be found in Apple's Safari HTML5 Canvas Guide (<http://developer.apple.com/library/safari/#documentation/AudioVideo/Conceptual/HTML5-canvas-guide/Introduction/Introduction.html>).

Animation Response to `gotFocus` and `lostFocus` Events

As previously discussed in the Video Support section, since the Harmony Viewer performs some HTML page preloading and caching, it's possible for screens that contain animated content to be resident in memory without being visible on screen. As a result, it's imperative that HTML slides containing animation appropriately respond to the system's "`gotFocus`" and "`lostFocus`" events. Animation should never be initiated before the page receives a "`gotFocus`" event. Likewise, animation should always be stopped, or reset to its initial state, when the page receives a "`lostFocus`" event.

Use of animated GIFs is discouraged in Harmony Viewer content because it is not possible to reliably reset playback in response to these events.

Third Party App Integration

The Harmony Viewer supports communication with other iOS applications through Custom URL Schemes.

For various security and performance reasons, the iPhone/iPad operating system imposes very strict file, data, and process execution restrictions on all third party applications. Each application is permitted to operate only on data that is accessible from within its own dedicated storage and processing location on the device, known as its "sandbox." No iOS application can directly access the data housed within another app's sandbox.

In addition to strict file sharing restrictions, there are active processing restrictions that prevent two apps from communicating directly with one another. When a new app is launched, the iPad's operating system immediately halts all active processing for any previously launched apps. Since only one app can be active at a time, it's impossible for two apps to communicate with one another directly. In the absence of direct communication, iOS apps must rely on the operating system to broker all inter-app communication operations.

Unlike conventional desktop or laptop systems, there are very few methods provided by iOS for this purpose. The most flexible approach available is the use of custom URL schemes. A custom URL string can be used to send a complex message to a specific application. In turn, an iOS application can be configured to receive messages that have been formatted according to a custom URL scheme. When a URL that conforms to a prescribed custom URL scheme is launched from any application on an iOS device, the operating system automatically launches the associated app, passing in the entire URL string. The newly

launched app can then process the received string and respond appropriately to any data contained therein.

Launching External Applications

Custom URL schemes are commonly used to launch external applications. For example, tapping on a conventional “<http://www.google.com>” link from within an email message will automatically close the iPad’s email app, launch the Safari web browser app, and navigate to the web page that corresponds to the link. Apple provides built-in support for several common URL schemes. The handlers for these schemes are fixed and cannot be changed.

Application	URL Scheme	Example
Safari	http	http://www.google.com
	https	https://mail.google.com
	ftp	ftp://ftp.mozilla.org
Mail	mailto	mailto:who.ever@wherever.com?cc=someone.else@wherever.com&subject=Hi&body=How%20are%20you?
Phone	tel	tel:1-800-555-5555
Messages	sms	sms:1-800-555-5555
Maps	http://maps.google.com	http://maps.google.com/maps?q=cupertino
YouTube	http://www.youtube.com	http://www.youtube.com/watch?v=VIDEO_IDENTIFIER
iTunes	http://phobos.apple.com	http://phobos.apple.com/WebObjects/MZStore.woa/wa/viewAlbum?i=156093464&id=156093462&s=143441
AppStore	http://itunes.apple.com	http://itunes.apple.com/us/app/glossario-paginemediche.it/id377692932?mt=8
Facetime	facetime	facetime://1-800-555-5555

Beyond the URL schemes supported directly by Apple, third party applications can extend support for additional custom URL schemes. Whenever a custom URL is executed on the device, if any app is configured to receive that scheme, the app is launched. An error message is displayed if no receiver app of the correct type is currently installed.

Application	URL Scheme	Example
Skype	skype	skype:echo123?call
Facebook	fb	fb://profile
		fb://friends
		fb://feed

HTML Based Execution of Custom URL Schemes

The Harmony Viewer supports the launching of external iOS applications via the execution of HTML or JavaScript code embedded in that HTML content. A custom URL scheme can be launched through a conventional web style link on an HTML page or through a JavaScript redirect of the current URL location.

```
<html>
<body>
  <a href="skype:echo123?call">Click Here to Launch Skype!</a>
</body>
</html>
```

FIGURE: Sample external app launch via HTML link

```
<html>
<head>
  <script>
    function launchSkype() {
      window.location = "skype:echo123?call";
    }
  </script>
</head>
<body>
  
</body>
</html>
```

FIGURE: Sample external app launch via JavaScript URL redirect

When an external application is launched from within the Harmony Viewer, the Harmony Viewer application is sent to the background, effectively closed, while the requested external application is opened for use. Since the two applications aren't active concurrently, the Harmony Viewer cannot communicate with, monitor, or track any interactions that occur in the external app. Consequently, it is recommended that any external app launch be paired with a custom tracking event in the Harmony Viewer's HTML content so that its use can be noted.

When the Harmony Viewer is "closed," the state of the active presentation is stored for later use. While the user interacts with the external application, the Harmony Viewer presentation remains effectively paused. When the user closes the external app and relaunched the Harmony Viewer, s/he has the option of resuming the previous presentation session where s/he left off, or returning to the Harmony Viewer landing screen to start a new presentation. If the Harmony Viewer presentation is resumed, the session's tracking log will include time stamped events that indicate that the presentation was paused and subsequently resumed.

SystemBridge JavaScript Library

Most of the visual and interactive content elements displayed by the Harmony Viewer framework are HTML, CSS, and JavaScript based. In some cases, these HTML based interactive elements need additional support from native Obj-C code to accomplish their desired behaviors due to the functional and performance limitations of HTML based views. SystemBridge.js is a JavaScript library, developed exclusively for the Harmony Viewer, to simplify these interactions between HTML based content and the underlying Obj-C code base. The SystemBridge JavaScript functions allow HTML content creators to execute a wide range of native functionality without resorting to direct modification of Obj-C code.

Each of the SystemBridge functions corresponds to a system command string that is ultimately parsed for execution by the objective-C framework. The key categories of SystemBridge functionality include:

- Content Navigation (e.g. SystemBridge.goToSection)
- Modal Media Controllers (e.g. SystemBridge.launchPDFViewer)
- Data Storage/Access (e.g. SystemBridge.setSessionValue)
- Tracking (e.g. SystemBridge.trackEvent)

A full functional specification of the available SystemBridge JavaScript functions can be found in Appendix A of this document.

Instantiation of the SystemBridge Library

The SystemBridge JavaScript library source code is not distributed for direct inclusion in the local file assets of Harmony Viewer content. This is in part to eliminate redundant inclusion of the JavaScript file within the app's local content repository (many HTML slides will use the SystemBridge library). More importantly, the code implemented in the SystemBridge library should be maintained in precise synchronization with the underlying source code of the Harmony Viewer itself to prevent discrepancies of implementation that could break functionality.

Instead of including the SystemBridge library from a local file resource, as would be typical in HTML content, it should instead be referenced from a version that has been embedded directly into the Harmony Viewer application. The embedded JavaScript file can be loaded by an HTML slide using the "resource://js/SystemBridge.js" file path.

```
<html>
<head>
  <script type="text/javascript" src="resource://js/SystemBridge.js"></script>
</head>
<body>
  <ul>
    <li>SystemBridge to launch external controllers</li>
    <li onclick="SystemBridge.launchFullscreenVideo('media/KOL.m4v');"> FullScreenVideo</li>
    <li onclick="SystemBridge.launchPDFViewer('media/PI.pdf','PDF Title');">PDFViewer</li>
    <li onclick="SystemBridge.launchFullscreenWebViewer('index.html','Web
Title');">FullScreenWebViewer</li>
    <li onclick="SystemBridge.launchPopUp('popUp.html', 1024, 768, 0, 0);">PopUp - multiple levels</li>
  </ul>
</body>
</html>
```

FIGURE: Sample instantiation of SystemBridge.js from an embedded resource


Content Navigation

The following SystemBridge functions can be used to initiate navigation through Harmony Viewer presentation content:

Function	Description
goToSlide	<ul style="list-style-type: none"> • Navigates to a specific presentation slide • Target slide is specified using the slide's Title field • Target slide will be launched in a modal controller if appropriate <ul style="list-style-type: none"> ◦ If slide configuration specifies a PDF parameter, slide will be launched in the PDF Viewer. ◦ If slide configuration specifies a PopUp parameter, slide will be launched in an HTML PopUp layer. • Example: SystemBridge.goToSlide("Efficacy");
goToAlternateCall	<ul style="list-style-type: none"> • Navigates to an alternate Active Call Storyflow (see section 8.3 Active Call Branching). • Optionally navigates to a specific presentation slide within the new Active Call Storyflow. • Target slide is specified using the slide's Title field • Example: SystemBridge.goToAlternateCall('alternateCallTitle', 'pageTitle');
slideToLeftPage	<ul style="list-style-type: none"> • Navigates to the presentation slide that would normally be arrived at via a left swipe gesture. • Example: SystemBridge.slideToLeftPage();
slideToRightPage	<ul style="list-style-type: none"> • Navigates to the presentation slide that would normally be arrived at via a right swipe gesture. • Example: SystemBridge.slideToRightPage();
slideToAbovePage	<ul style="list-style-type: none"> • Navigates to the presentation slide that would normally be arrived at via an up swipe gesture. • Example: SystemBridge.slideToAbovePage();
slideToBelowPage	<ul style="list-style-type: none"> • Navigates to the presentation slide that would normally be arrived at via a down swipe gesture. • Example: SystemBridge.slideToBelowPage();

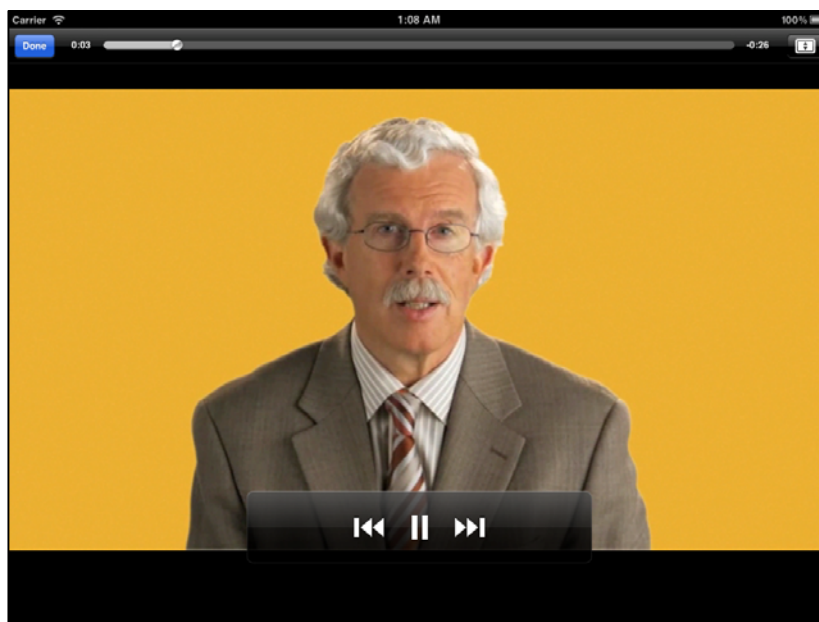
Modal Media Controllers

SystemBridge functions can be used to launch certain media files within full screen modal dialogs that support more performant interactions than are available from within a standard HTML based environment. The available modal media controllers are:

Media Controller	Description
PDF Viewer	<p>launchPDFViewer launchPDFViewerAtPage</p>  <ul style="list-style-type: none"> • Launches a full screen PDF Viewer dialog on top of the existing Harmony Viewer content. • The specified title is displayed in the dialog's header. • LaunchPDFViewerAtPage can be used to launch a PDF to a specific page in the document. • Optionally, a tracking message can also be specified as an additional parameter. • Example: SystemBridge.launchPDFViewer('media/PI.pdf', 'PDF Title');

Fullscreen Video Viewer

launchFullscreenVideo

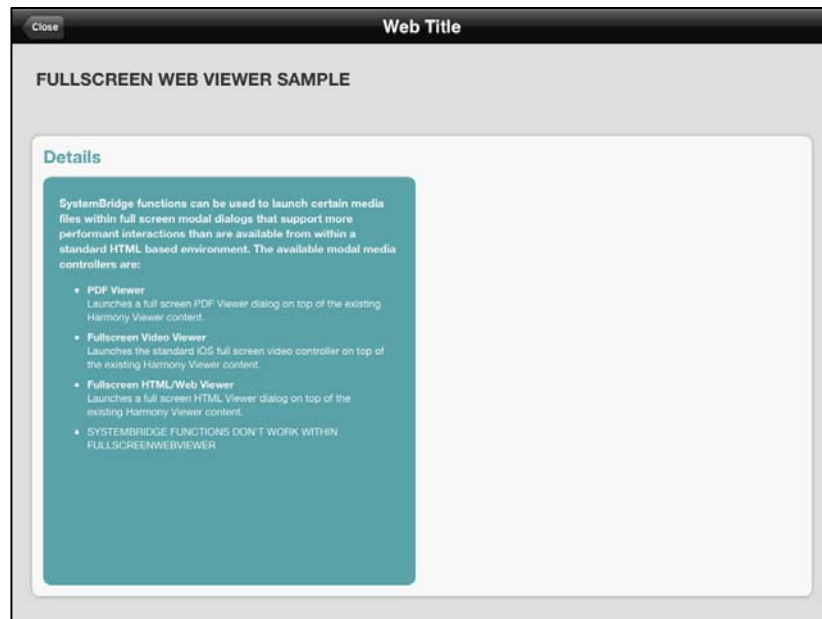


- Launches the standard iOS full screen video controller on top of the existing Harmony Viewer content.
- Optionally, a tracking message can also be specified as an additional parameter.
- Example:

```
SystemBridge.launchFullscreenVideo('media/KOL.m4v');
```

Fullscreen HTML/Web Viewer

launchFullscreenWebView

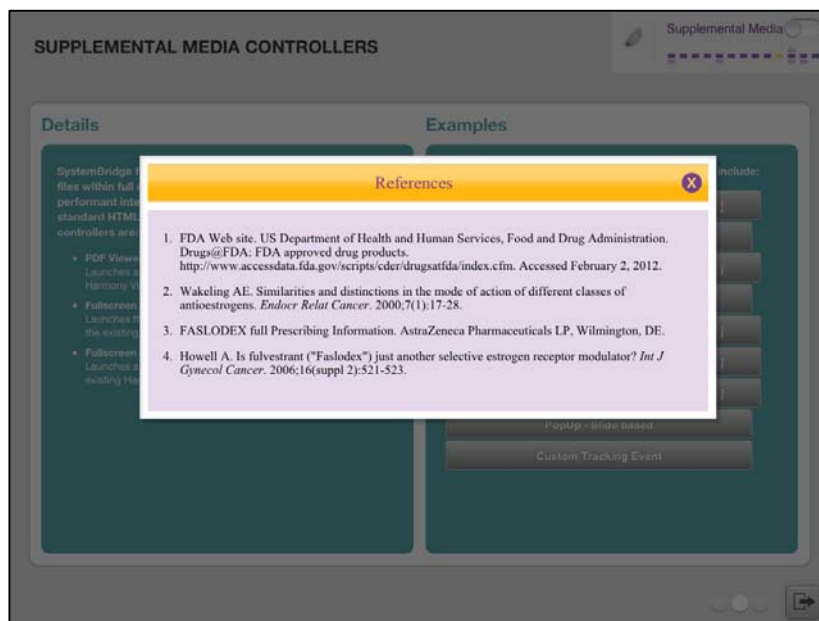


- Launches a full screen HTML Viewer dialog on top of the existing Harmony Viewer content.
- The specified title is displayed in the dialog's header.
- Optionally, a tracking message can also be specified as an additional parameter.
- This can be used both to view local HTML content and web based HTML content; however, since stable internet connections are NOT guaranteed during Harmony Viewer presentation sessions, significant use of web based content is discouraged.
- Example:

```
SystemBridge.launchFullscreenWebView('index.html','Web Title');
```

HTML PopUp

launchPopUp closeTopPopUp



- Launches a layer of HTML on top of the existing Harmony Viewer content with a transparent background.
- Popup HTML must include SystemBridge JavaScript code (closeTopPopUp) to dismiss the popup.
- Optionally, a tracking message can also be specified as an additional parameter.
- Useful for a multitude of popup effects when it is preferable to store the popup HTML separately from the source slide's HTML (e.g. if popup content is sufficiently dense to negatively impact the loading time of the slide HTML).
- Multiple layers of popups can be launched on top of one another.
- Example:

```
SystemBridge.launchPopUp('popUp.html', 1024, 768, 0, 0);
```

Data Storage/Access

iOS, and consequently the Harmony Viewer, supports HTML based data storage and access via the HTML5 localStorage API. The localStorage API provides a way for HTML pages to store named key/value data pairs locally. This data is designed to persist even after the HTML page is closed. Unfortunately, in the course of developing the Harmony Viewer, a significant bug was found in the iOS localStorage implementation. Certain iOS software updates have a history of compromising previously stored data, effectively invalidating

access to the data until the app is entirely deleted and reinstalled. Use of localStorage in Harmony Viewer slide content is therefore NOT recommended.

In order to compensate for some localStorage functional irregularities and provide slightly expanded functionality, SystemBridge JavaScript functions have been added that operate similarly to the localStorage APIs.

Function	Description
setSessionValue	<ul style="list-style-type: none"> Stores a specified key/value data pair into a presentation session specific data store. At the end of the presentation session, the session data store is cleared, so this should only be used to store temporary data. The session data store is a useful place to store information that needs to be communicated between different slides in a single presentation (e.g. on SlideX, option B was selected, so SlideY's display should be modified accordingly). Example: SystemBridge.setSessionValue('whateverKey', 'whateverData');
requestSessionValue	<ul style="list-style-type: none"> Retrieves a specified key/value data pair from a presentation specific data store. Adheres to the standard request and callback model that is common for many web technologies. requestSessionValue specifies the data key and a JavaScript callback function that will be executed when the corresponding data value is available. Example: SystemBridge.requestSessionValue('whateverKey', onWhateverDataIsReady);
setPersistentValue	<ul style="list-style-type: none"> Stores a specified key/value data pair into a persistent data store. The persistent data store is NOT cleared at the end of each presentation session, so it can be used to store long-lived data. Example: SystemBridge.setPersistentValue('whateverKey', 'whateverData');

requestPersistentValue	<ul style="list-style-type: none"> Retrieves a specified key/value data pair from the persistent data store. Adheres to the standard request and callback model that is common for many web technologies. <code>requestPersistentValue</code> specifies the data key and a JavaScript callback function that will be executed when the corresponding data value is available. Example: <code>SystemBridge.requestPersistentValue('whateverKey', onWhateverDataIsReady);</code>
-------------------------------	---

Tracking

The Harmony Viewer performs most of its presentation tracking tasks without requiring intervention by content creators; however, the SystemBridge library does provide a “trackEvent” function that can be used within HTML content to explicitly track custom events that fall outside the scope of the standard tracking operations. More comprehensive information can be found in the Tracking section of this guide.

Miscellaneous Utility Functions

The SystemBridge library also includes several useful utility functions that are fully documented in Appendix A of this guide.

Navigation Layer

The Navigation Layer is intended to house all of the persistent screen elements for the presentation. Typically this includes functional elements like navigation or menu buttons, as well as purely aesthetic branding elements.

Due to technical limitations on the iPad, the Navigation Layer functionality could not be achieved entirely with HTML. Instead, a custom Navigation Layer class was created using native Obj-C code. This class contains a user interface template for xCode's Interface Builder (Apple's Obj-C software development environment), which allows the Navigation Layer's graphics to be fully customized.

Default Navigation layer

A single default Navigation Layer skin is embedded in the Harmony Viewer app. The default navigation layer provides a presentation agnostic design that can be used by many different presentations to access all of the Harmony Viewer functionality without requiring additional xCode development work.

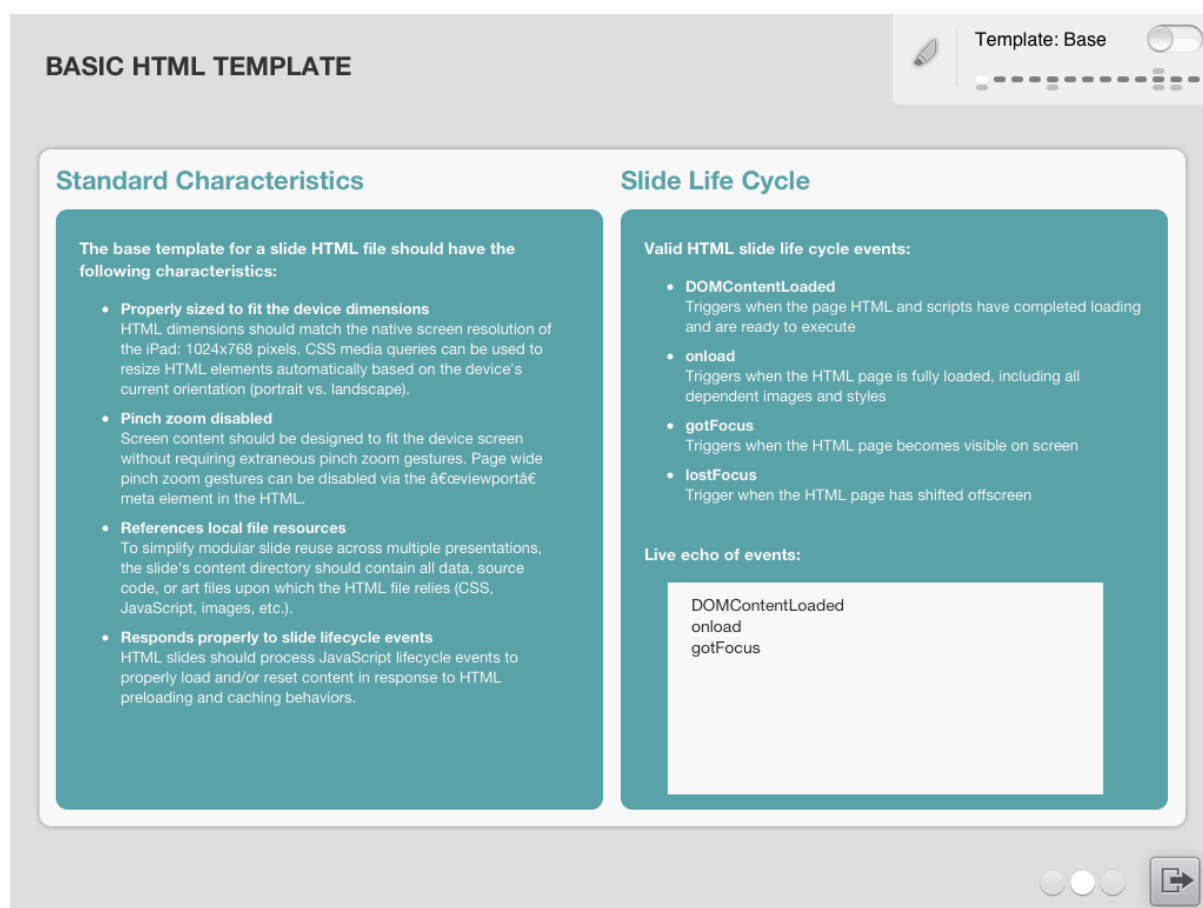


FIGURE: Default navigation layer when in presentation mode. Navigation layer user interface elements are confined to top right and bottom right corners of the screen.

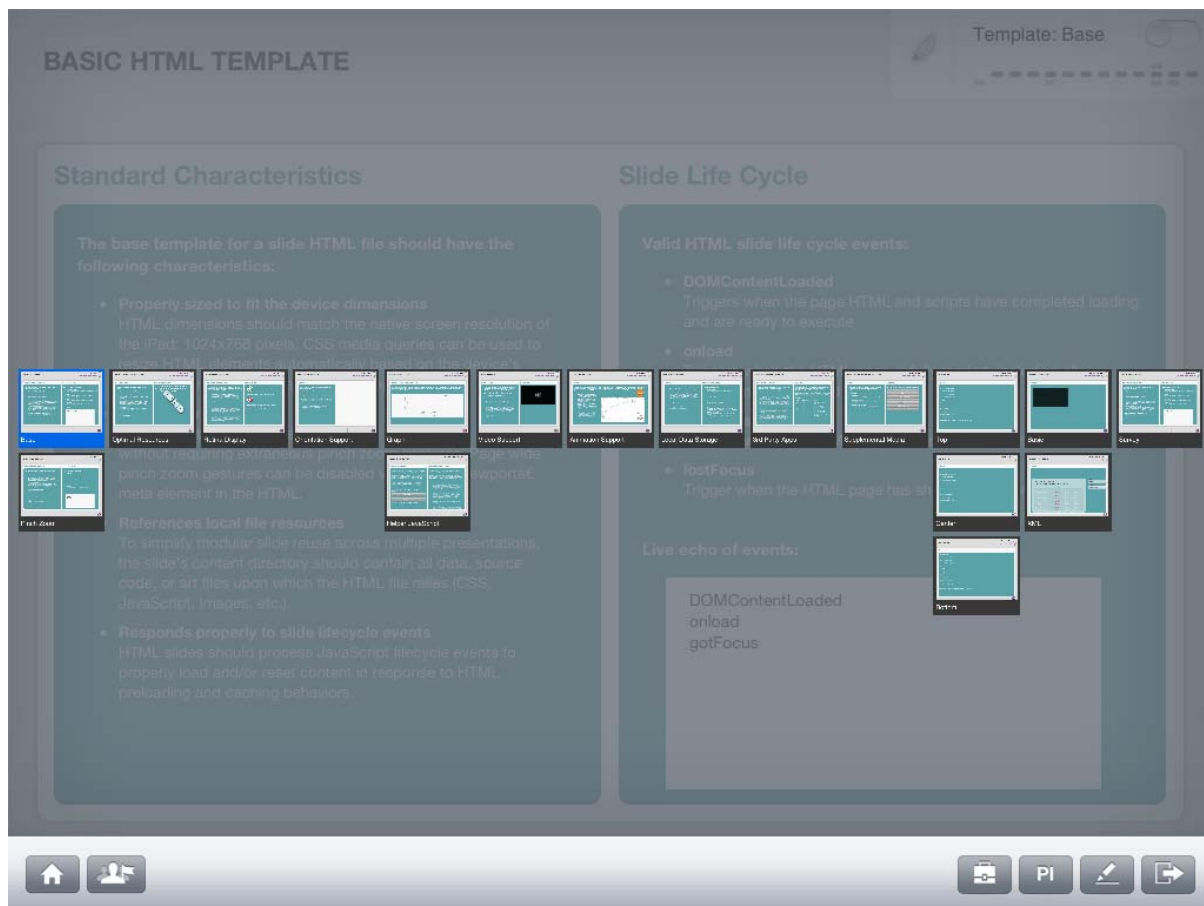












FIGURE: Default navigation layer when full menu layer has been accessed (via swipe up from bottom of screen). Navigation layer user interface elements consume the full screen.

The following standard user interface elements are provided in the default navigation layer skin:

Feature	Icon	Description
Breadcrumb		<ul style="list-style-type: none"> The breadcrumb control provides a visual indication of the current screen in the presentation. A text block displays the title of the current slide above a simple map indicating the position of the current slide in the presentation's Active Call map.

Screen Lock Slider		<ul style="list-style-type: none"> • When activated, the screen lock slider disables all navigation layer and swipe gesture functionality. • All functionality within the current slide asset remains enabled. • This feature is useful for disabling potentially distracting functionality when the user wants to hand the iPad to another person to peruse or interact with the current slide. • This also prevents a secondary user from inadvertently executing unwanted system functionality (e.g. navigating away from the current slide in the presentation).
Highlighter Button		<ul style="list-style-type: none"> • When activated, the user can draw on the screen with finger gestures or a stylus to draw attention to specific slide content. • While drawing, most other functionality is disabled. • When deactivated, the highlighted content is cleared from the screen. • On screen drawing is not retained between drawing sessions.
Attitudinal Indicator Button		<ul style="list-style-type: none"> • The attitudinal indicator button is a three state toggle that, when tapped, toggles between Negative, Neutral, Positive. • Default position = Null • If an attitudinal value is indicated via this control during a presentation session, the screen in question will retain that value for tracking purposes. • Attitudinal data values can be reviewed and/or altered in the Harmony Viewer's Post Call Review interface prior to the transmission of presentation tracking data to any external data warehouse or reporting systems.

Exit Presentation Button		<ul style="list-style-type: none"> • A single tap on the exit presentation button will launch a popup dialog containing a list of all presentations that are currently available to the application, as well as an “Exit Presentation” option. • The user can select any presentation from the list presented to navigate to another presentation. • The user can select the “Exit Presentation” option to close the current presentation. When closing the presentation, the user can opt to return to the Harmony Viewer landing screen, proceed to the Post-Call Review screen, save the presentation tracking record for later review, or purge the presentation tracking record from the system entirely. • A double tap on the exit button will automatically navigate to the next presentation available to the application, if a targeted presentation order was previously established.
Home Button		<ul style="list-style-type: none"> • The home button returns the user to the first slide in the presentation.
Audience Change Button		<ul style="list-style-type: none"> • A single tap on the audience change button flags the current position in the presentation’s tracking log with an audience change event. • Audience change events can be matched with appropriate HCP additions or deletions on the Post-Call Review screen.
Briefcase Button		<ul style="list-style-type: none"> • A single tap on the briefcase button launches the briefcase user interface on top of the current presentation interface.
Prescribing Information Button		<ul style="list-style-type: none"> • A single tap on the prescribing information button navigates to the defined “PI” slide of the current presentation. The “PI” slide can correspond to any of the slide types (standard HTML, PDF, etc.).

Sample Signature Button		<ul style="list-style-type: none"> • A single tap on the sample signature button launches the external Oracle iSales application to the appropriate sample signature screen. • When returning to the Harmony Viewer app from the Oracle iSales app, the user will be prompted to resume the existing presentation session.
Thumbnail Menu		<ul style="list-style-type: none"> • The thumbnail menu fills the majority of the screen when launched. • The thumbnail imagery is automatically generated based on the slide and presentation structure information found in the slide and presentation configuration files. • The thumbnail map can be panned via finger drags and scaled via standard pinch zoom gestures. • A single tap on an individual slide thumbnail will navigate to that presentation slide and dismiss the thumbnail menu. • A single tap in the background area of the thumbnail menu will dismiss the thumbnail menu.

NOTE: Not all of the above navigation layer functionality is appropriate for all markets. Market specific configurations will be created and distributed as appropriate.

Optional Customized Navigation Layer

A fully customizable navigation layer template will be provided in cases where the default navigation layer does not meet the needs of a particular presentation. In the xCode based navigation layer template, the imagery associated with the default navigation layer user interface elements can be changed at will. The template also provides mechanisms for adding new buttons to the navigation layer that target specific content navigation or system commands.

The xCode template and further description of the associated editing and compiling procedures will be added to this document when the specification has been finalized for subsequent releases.

Presentations

In order to maximize creative flexibility, while maintaining a minimal memory footprint, presentation content is created in a highly modular manner. Individual content modules are organized into distinct presentations via small presentation configuration files that are distributed with the base presentation asset, and which can be updated independently of any referenced slide content.

Each presentation asset within the Harmony Viewer framework exists as a uniquely named, self-contained content directory. This content directory should minimally contain the presentation's default configuration file (`_default.plist`) and a cover image (JPG or PNG) used for the presentation's thumbnail graphic on the Harmony Viewer presentation selection screen.

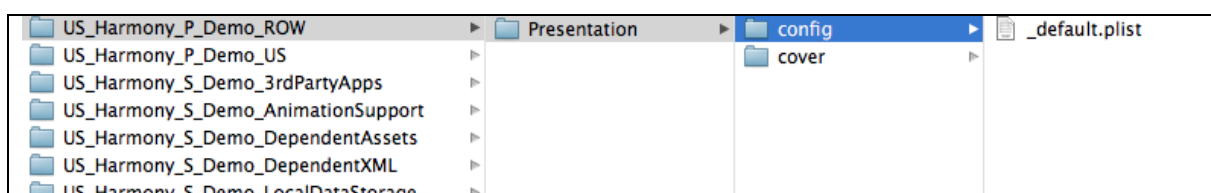


FIGURE 3.0: Sample file structure for a presentation asset

Alternate presentation configuration files can also be included in the presentation asset's "config" directory for selection from the Harmony Viewer presentation selection screen. The "`_default.plist`" configuration will be loaded by default when the user taps the presentation thumbnail on the presentation selection screen, but alternate presentation configurations are available from the presentation's alternate configurations popover menu.

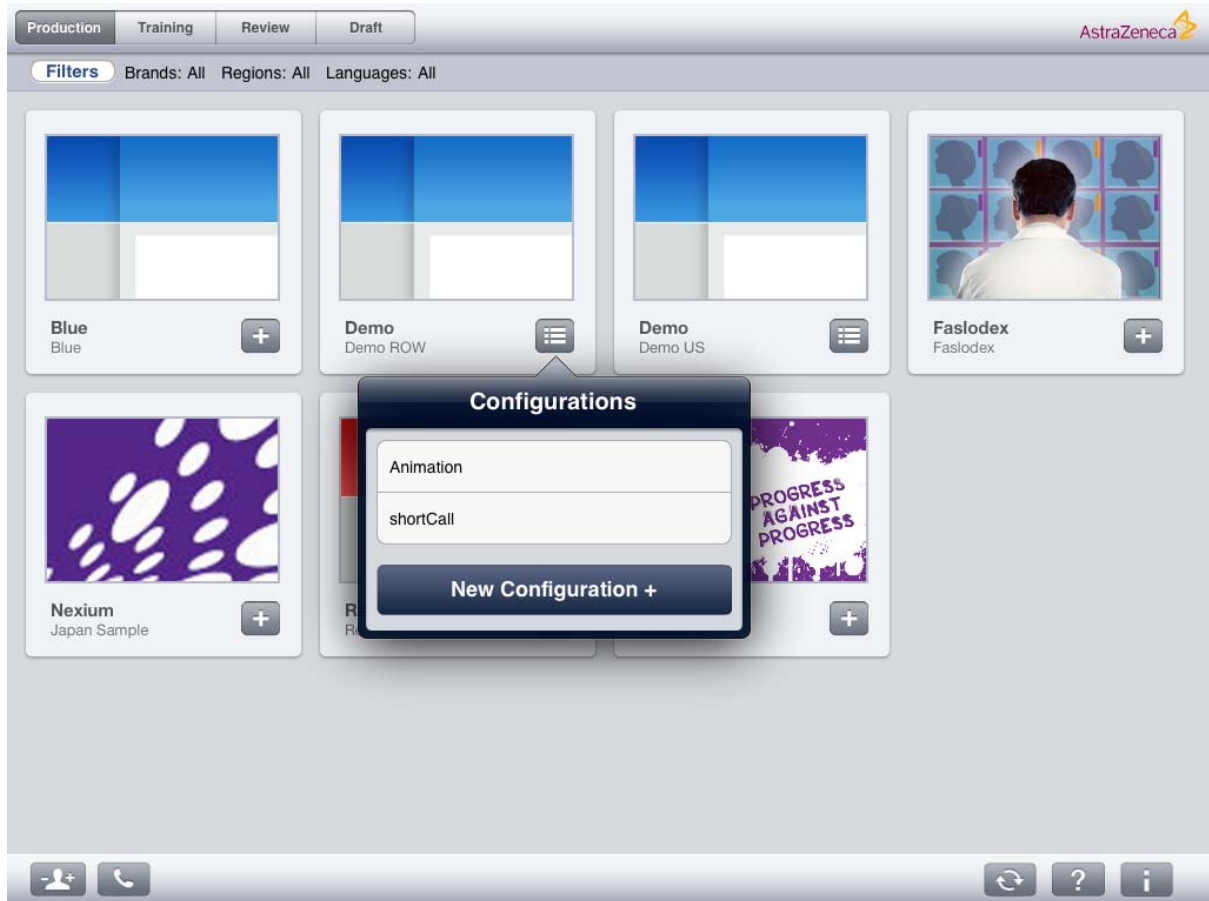


FIGURE: Harmony Viewer presentation selection screen with alternate configuration popover displayed.

Presentation Configuration File

The primary purpose of a presentation configuration file is to specify the overall content structure to the Harmony Viewer; however, some additional Harmony Viewer properties, that are configurable at the individual presentation level, can also be specified in the presentation configuration file.

Key	Type	Value
▼ Root	Dictionary	(7 items)
▶ Slides	Array	(16 items)
▶ ActiveCall	Array	(13 items)
▶ AlternateCalls	Dictionary	(2 items)
▶ Menu	Dictionary	(1 item)
▶ Supported Orientations	Dictionary	(4 items)
ScreenLockEnabled	Boolean	YES
DrawingEnabled	Boolean	YES

FIGURE: Sample presentation configuration file

Slides Array

The most important information in a presentation configuration file is the Slides Array. The Slides Array specifies the hierarchical organization of screen content in the presentation (content sections, subsections, etc.), as well as the core parameters for each individual content screen that are required for proper processing and display in the Harmony Viewer. This set of slide information is also used to dynamically generate a thumbnail-based popup menu that is accessible throughout the presentation.

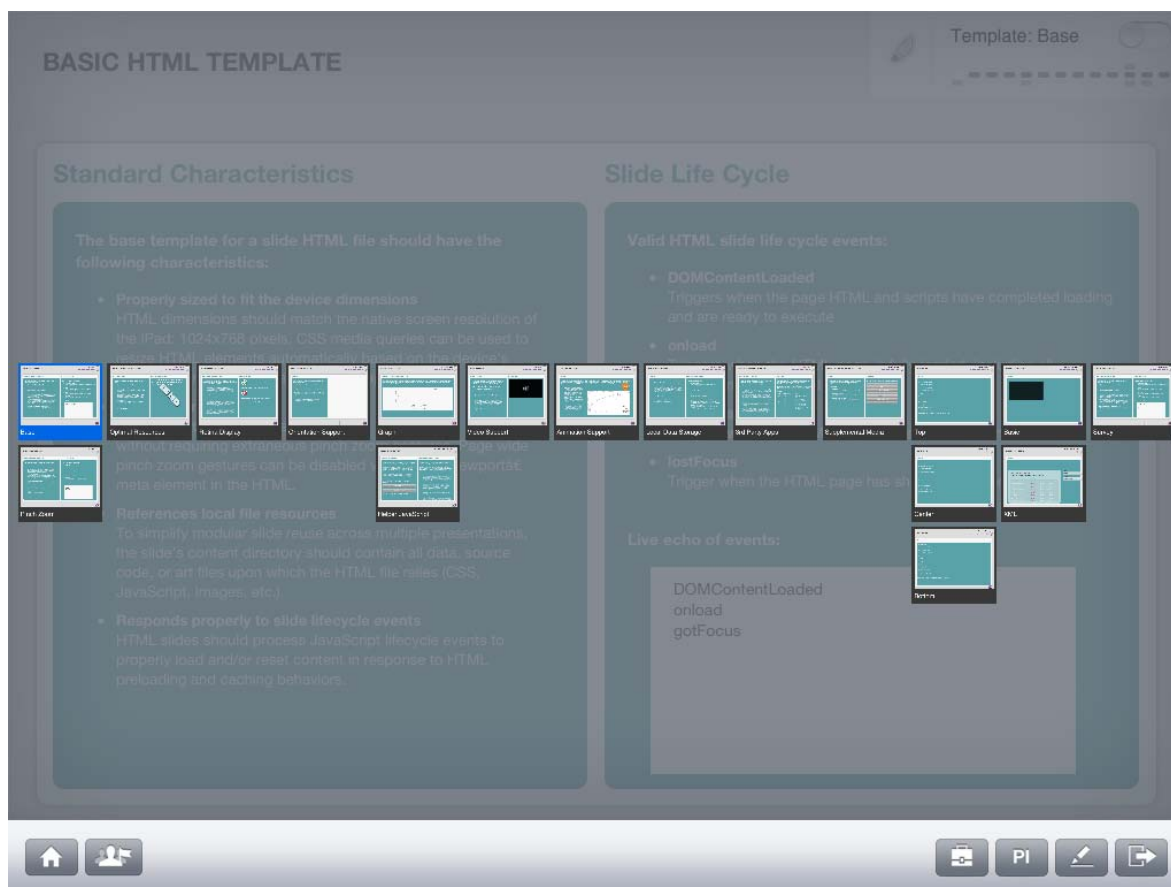


FIGURE: Popup menu detail from sample presentation

In the interests of content modularity and reuse, presentation configuration files should refer almost exclusively to external slide assets. As such, most slide configuration data should be confined to the individual slide configuration files. The presentation configuration file should refer to these external slide assets by AssetID. The presentation configuration parameters required to reference external slide assets are:

Key Property	Description
Title	<p>(Required) String</p> <ul style="list-style-type: none"> • A unique plain text title for this slide. • The title is often used by the Harmony Viewer to refer to this screen content when doing so via absolute URL would be awkward (e.g. in a presentation active call configuration or a SystemBridge goToSlide execution). • Slide title is also used to populate certain Harmony Viewer user interface elements (e.g. breadcrumb, thumbnail menu, pre-call slide sorter, etc.).
AssetID	<p>(Required) String</p> <ul style="list-style-type: none"> • The AssetID value corresponds to the external asset's base directory within the Harmony Viewer's local content repository, as detailed in the "Naming Conventions" section of this document. • Other configuration parameters for the externally referenced slide asset are automatically inherited from the slide asset's internal configuration file (e.g. Title, URL, etc. are extracted from the slide configuration file and need not be explicitly included in the presentation configuration file). • Slide parameters explicitly included in the presentation configuration file take precedence over parameters listed in the external slide's configuration file (e.g. if a slide Title is included in the presentation configuration file, the Harmony Viewer will ignore the Title parameter specified in the external slide's configuration file). • In the interests of content reuse and modular efficiency, it is strongly recommended that all presentation slides be stored externally and referenced in the presentation configuration file using the AssetID parameter.
Subcontent	<p>(Optional) Array</p> <ul style="list-style-type: none"> • The existence of a subcontent array indicates that an additional array of slide content exists hierarchically below this screen. • Subcontent arrays can be used to group related content together into sections. • There is no limit to the number of hierarchical content levels, though it is very rare for content to delve deeper than two levels. • Adjacent slides within a subcontent group can be accessed via a vertical finger swipe within the presentation.

Key	Type	Value
▼ Root	Dictionary	(7 items)
▼ Slides	Array	(16 items)
▶ Item 0	Dictionary	(3 items)
▶ Item 1	Dictionary	(3 items)
▼ Item 2	Dictionary	(2 items)
Title	String	Retina Display
AssetID	String	US_Harmony_S_Demo_RetinaDisplay
▶ Item 3	Dictionary	(3 items)
▶ Item 4	Dictionary	(3 items)
▶ Item 5	Dictionary	(3 items)
▶ Item 6	Dictionary	(3 items)
▶ Item 7	Dictionary	(3 items)
▶ Item 8	Dictionary	(2 items)
▶ Item 9	Dictionary	(3 items)
▶ Item 10	Dictionary	(2 items)
▼ Item 11	Dictionary	(2 items)
Title	String	Dependent Assets
▼ Subcontent	Array	(2 items)
▼ Item 0	Dictionary	(2 items)
Title	String	Basic
AssetID	String	US_Harmony_S_Demo_DependentAssets
▼ Item 1	Dictionary	(2 items)
Title	String	XML
AssetID	String	US_Harmony_S_Demo_DependentXML
▶ Item 12	Dictionary	(3 items)
▶ Item 13	Dictionary	(3 items)
▶ Item 14	Dictionary	(2 items)
▶ Item 15	Dictionary	(2 items)
▶ ActiveCall	Array	(13 items)
▶ AlternateCalls	Dictionary	(2 items)
▶ Menu	Dictionary	(1 item)
▶ Supported Orientations	Dictionary	(4 items)
ScreenLockEnabled	Boolean	YES
DrawingEnabled	Boolean	YES

FIGURE: Slide array detail from sample presentation configuration file

Active Call Storyflow

Another important part of the presentation configuration file is the Active Call Array. The Active Call Array contains a simple list of slide titles, corresponding to those specified in the Slides Array.

NOTE: A slide that exists in a Subcontent hierarchy within the Slides Array can be included in the Active Call Array by specifying the full hierarchical path. This path can be constructed by concatenating the relevant slide title values with “::” delimiters (e.g. if the Slides Array

contains an entry with Title = “Safety”, which in turn has a Subcontent item with Title = “Recovery”, the “Recovery” slide’s path would be “Safety::Recovery”).

The order of slides in the Active Call Array defines the order of screens that are accessible via the Harmony Viewer’s horizontal gesture navigation. As previously discussed, subcontent groupings in the presentation configuration file’s Slides array define the order of screens that are accessible via the vertical gesture navigation. In this way, the Slides and Active Call arrays together define the map of gesture accessible presentation content, as is ultimately depicted in the navigation layer’s breadcrumb control.

Key	Type	Value
▼ Root	Dictionary	(7 items)
▶ Slides	Array	(16 items)
▼ ActiveCall	Array	(9 items)
Item 0	String	Optimal Resources
Item 1	String	Template::Base
Item 2	String	Retina Display
Item 3	String	Orientation Support
Item 4	String	Video Support
Item 5	String	Animation Support
Item 6	String	Local Data Storage
Item 7	String	3rd Party Apps
Item 8	String	Supplemental Media
▶ AlternateCalls	Dictionary	(2 items)
▶ Menu	Dictionary	(1 item)
▶ Supported Orientations	Dictionary	(4 items)
ScreenLockEnabled	Boolean	YES
DrawingEnabled	Boolean	YES

FIGURE: ActiveCall array detail from sample presentation configuration file

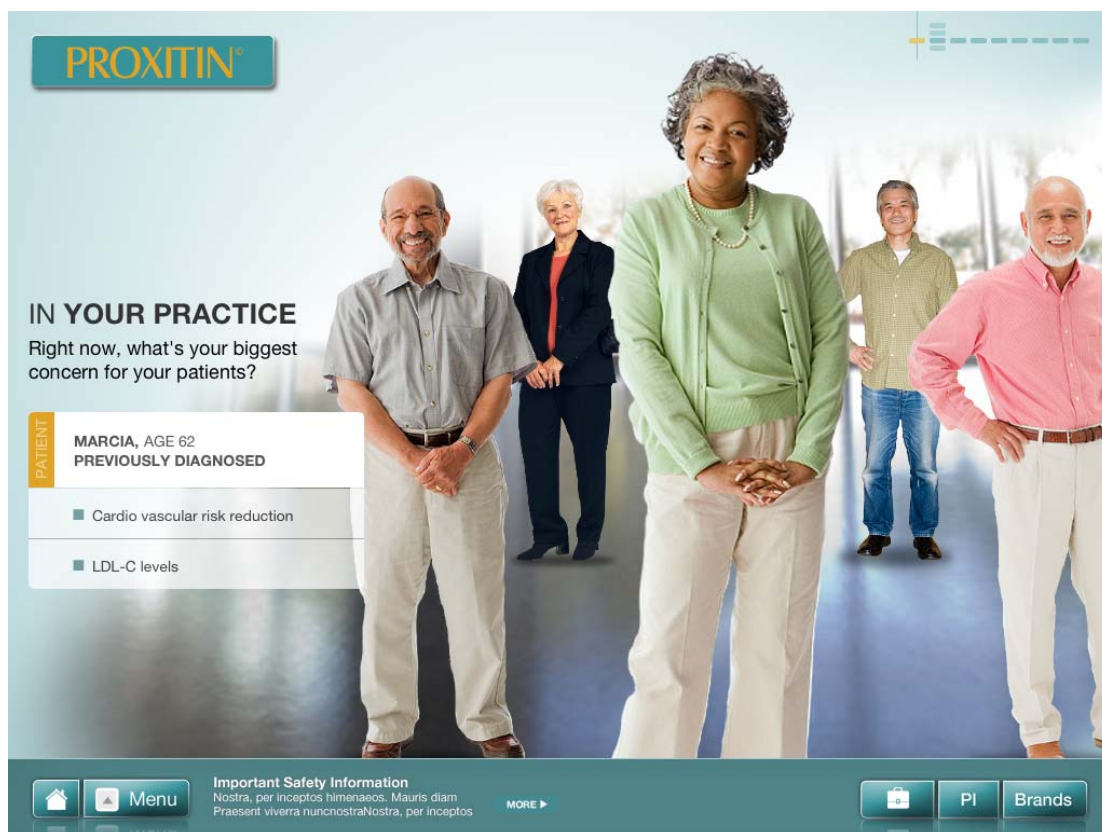


FIGURE: Sample presentation screen with breadcrumb graphic in upper right corner denoting the screens that can be accessed via gesture navigation. Horizontal screen adjacency is defined by the ActiveCall array. Vertical screen adjacency is derived from adjacent subcontent in the Slides array.

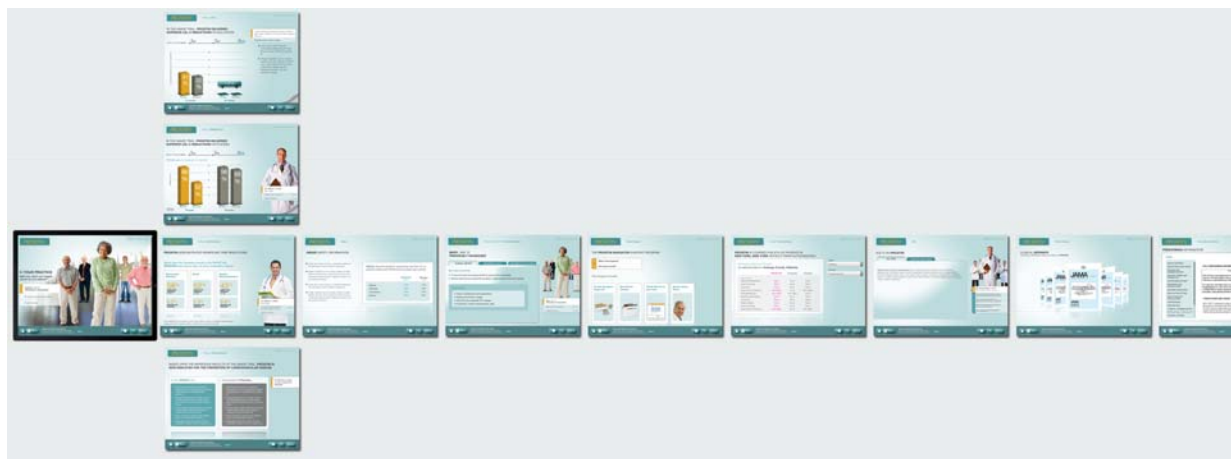


FIGURE: Map of gesture navigable content structure as defined by the ActiveCall and Slides arrays in a sample presentation configuration file

The Active Call concept allows for presentations to be dynamically reconfigured based on user interaction or the application of dynamic presentation configuration rules with minimal effort. By simply altering the Active Call configuration, a multitude of custom presentations can be generated from identical screen content files. No additional file downloads or redundant file storage is required.

Key	Type	Value
▼ Root	Dictionary	(7 items)
▶ Slides	Array	(16 items)
▼ ActiveCall	Array	(4 items)
Item 0	String	Template::Base
Item 1	String	Orientation Support
Item 2	String	Touch Support::Graph
Item 3	String	Video Support
▶ AlternateCalls	Dictionary	(2 items)
▶ Menu	Dictionary	(1 item)
▶ Supported Orientations	Dictionary	(4 items)
ScreenLockEnabled	Boolean	YES
DrawingEnabled	Boolean	YES

FIGURE: Altered ActiveCall array detail from sample presentation configuration file. All other configuration data, including the Slides array, remains unchanged from the previous example.

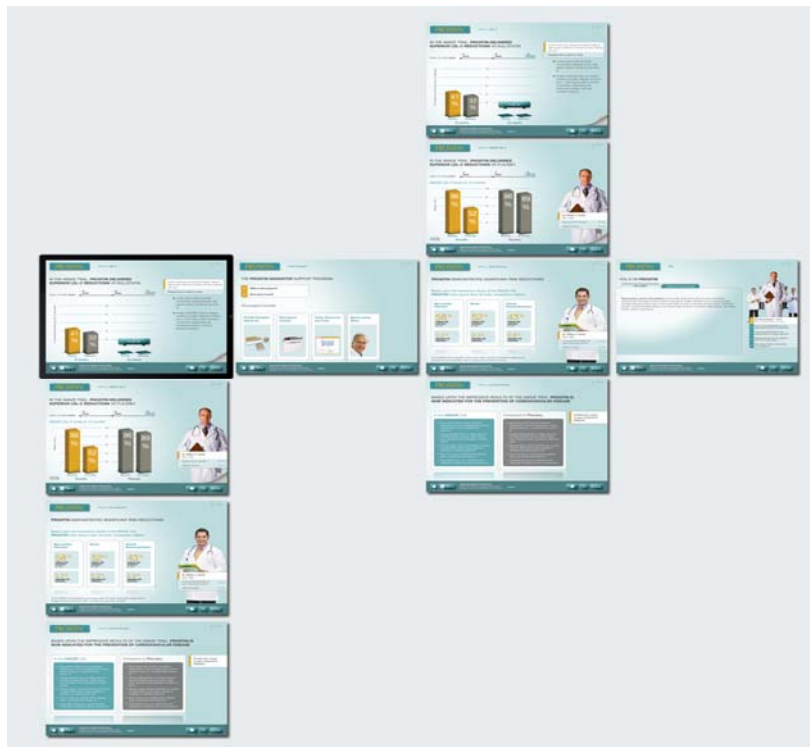


FIGURE: Map of gesture navigable content structure as defined by the revised ActiveCall array in a sample presentation configuration file

Active Call Branching

Although the Active Call Storyflow concept allows for intuitive, efficient, and fluid navigation through presentation slide content, it is sometimes advantageous to rapidly jump to an alternate Storyflow from within a presentation session. Many presentations can benefit from the inclusion of one or more logical branching points that can dynamically direct the user to content that is targeted more towards the audience's needs (e.g. doctor vs. nurse branch or MOA vs. efficacy branch). The Harmony Viewer supports this functionality via a combination of presentation configuration file properties and a SystemBridge JavaScript function.

Alternate call storyflows can be included in the presentation configuration file within an "AlternateCalls" dictionary. Each alternate call should be specified within the "AlternateCalls" dictionary as a uniquely named Array of slide titles. When an alternate call storyflow is loaded, the presentation will function identically to the previously discussed ActiveCall functionality (e.g. breadcrumb graphics and gesture navigation will change to reflect the modified storyflow).

Key	Type	Value
▼ Root	Dictionary	(7 items)
▶ Slides	Array	(16 items)
▶ ActiveCall	Array	(13 items)
▼ AlternateCalls	Dictionary	(2 items)
▼ Animation	Array	(4 items)
Item 0	String	Video Support
Item 1	String	Animation Support
Item 2	String	Touch Support::Graph
Item 3	String	Supplemental Media
▶ shortCall	Array	(4 items)
▶ Menu	Dictionary	(1 item)
▶ Supported Orientations	Dictionary	(4 items)
ScreenLockEnabled	Boolean	YES
DrawingEnabled	Boolean	YES

FIGURE: AlternateCalls dictionary detail from sample presentation configuration file. All other configuration data, including the Slides array, remains unchanged from the previous example.

The alternate call storyflow can be loaded from within a presentation slide's HTML content via the provided `SystemBridge.goToAlternateCall` function. A specific slide can optionally be included in the function call to navigate to said slide within the specified alternate call storyflow.

```

<html>
<head>
  <script type="text/javascript" src="resource://js/SystemBridge.js"></script>
</head>
<body>
  <a onclick="SystemBridge.goToAlternateCall('Animation');">
    Switch to the Animation storyflow </a>
</body>
</html>

```

FIGURE: Sample HTML code demonstrating Active Call Branching

Per Presentation Properties

Several other Harmony Viewer properties that can be configured on a per presentation basis are also included in the presentation configuration file. Valid per presentation properties include:

Key Property	Description						
Menu	<div>(Optional) Dictionary</div> <div><table><tr><td>▼ Menu</td><td>Dictionary</td><td>(1 item)</td></tr><tr><td>AssetID</td><td>String</td><td>US_Harmony_N_Demo_Menu_ROW</td></tr></table></div> <div><ul style="list-style-type: none">Specifies configuration parameters for the presentation’s navigation layer (as detailed in the “Navigation Layer” section of this document).If no Navigation Layer key is included, the presentation will use the default Navigation Layer skin that is compiled into the core Harmony Viewer application.An AssetID key can be used within the Navigation Layer dictionary to reference an external Navigation Layer asset for use with this presentation.</div>	▼ Menu	Dictionary	(1 item)	AssetID	String	US_Harmony_N_Demo_Menu_ROW
▼ Menu	Dictionary	(1 item)					
AssetID	String	US_Harmony_N_Demo_Menu_ROW					
Supported Orientations	<div>(Optional) Dictionary</div> <div><ul style="list-style-type: none">Specifies which of the four available user interface orientations are supported by this presentation.Available orientation keys:<ul style="list-style-type: none">Landscape Left (Boolean) – default value = YESLandscape Right (Boolean) – default value = YESPortrait (Boolean) – default value = YESPortrait Upside Down (Boolean) – default value = YES</div>						
ScreenLockEnabled	<div>(Optional) Boolean</div> <div><ul style="list-style-type: none">Specifies whether or not the Screen Lock control should be displayed in the presentation’s navigation layer.Default Value = YES</div>						
DrawingEnabled	<div>(Optional) Boolean</div> <div><ul style="list-style-type: none">Specifies whether or not the Highlighter control should be displayed in the presentation’s navigation layer.Default Value = YES</div>						

ExitOnDoubleTap	<p>(Optional) Boolean</p> <ul style="list-style-type: none"> • Specifies whether or not a double tap on the presentation navigation layer's Exit button should navigate to the next presentation in the presentation list. • Default Value = YES
------------------------	--

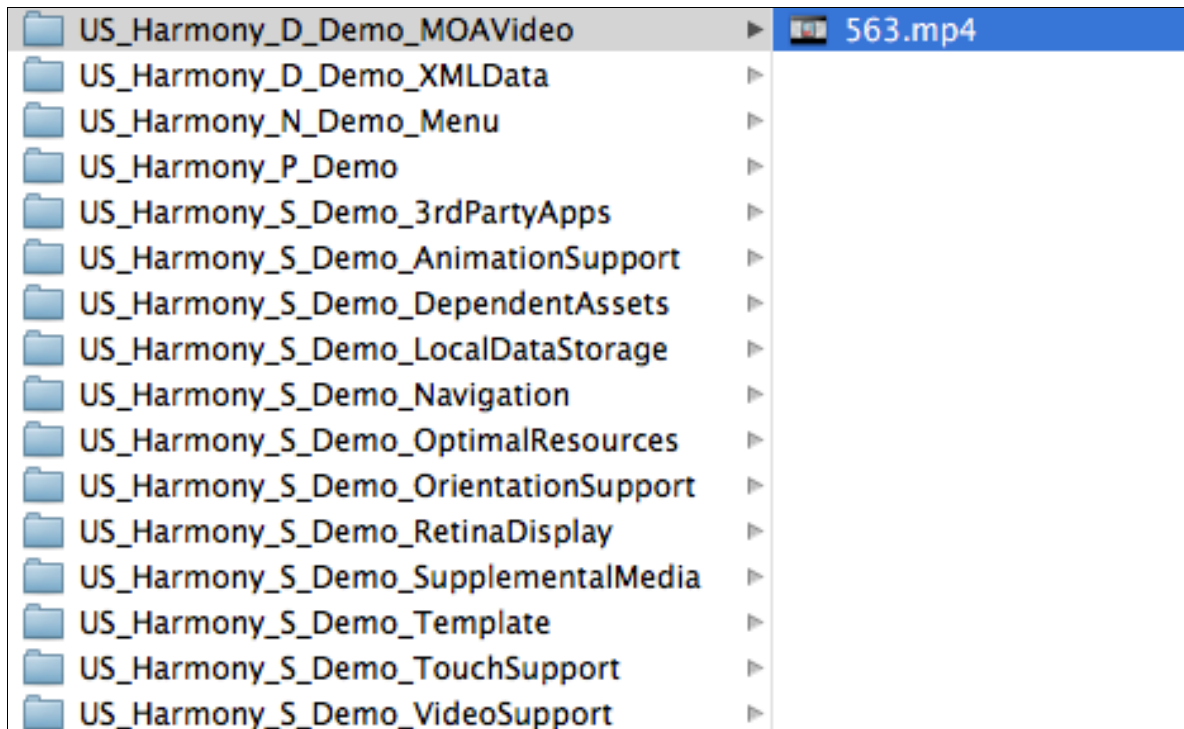
Presentation independent Data Assets

The Harmony Viewer supports an additional modular asset type called a “Data Asset.” Data asset directories can contain arbitrary files of any type and require no configuration file because data assets are not accessed and displayed directly by the Harmony Viewer framework. For data assets to be useful, they must be referenced and used by a separate HTML slide file.

Data assets are useful for sharing files across multiple slides/presentations without downloading or storing redundant data on the iPad. They can also be used to subdivide data heavy slides into multiple content packages for more granular control over future update download sizes (e.g. isolate a large video file in a separate data asset so that the corresponding slide’s HTML data can be updated independently). Typical examples of data assets include:

- **Pure data files** (e.g. XML files for use by a formulary finder slide). It is sometimes advantageous to separate the underlying data files from purely graphical display files so that they can be updated independently at differing intervals.
- **Repetitive file assets** (e.g. resource files that are used by many different slides or presentations). It is sometimes advantageous to isolate common resources in a data asset so that there is only one asset to maintain and store on the system instead of replicating and maintaining potentially dozens of copies of the same files within many different independent slide asset directories.
- **Large media files** (e.g. videos or PDFs). It is sometimes advantageous to isolate large media files from other slide content to prevent repetitive downloading of the media file during content updates. For example, if a slide asset contains both a video file and an HTML file, a small update to the HTML file will force the user to update the entire slide asset, including the large, unchanged, media file. Isolating the media file to its own asset will prevent unnecessary “update” downloads.

Data assets, as well as any other content asset, can be referenced directly from slide assets via the provided “package://” URL scheme. HTML code can access the root directory of the external asset by specifying the desired AssetID as the URL’s initial path component. Additional path components can then be appended to the URL to target any files contained within the asset’s root directory.



```
<html>
<body>
  <video src="package://US_Harmony_D_Demo_MOAVideo/563.mp4" width="433" height="243" />
</body>
</html>
```

FIGURE: Sample data asset usage: 1) separate data asset is created to house a video file, 2) slide asset's HTML file references the data asset via the "package://" URL scheme.

Tracking

Harmony Viewer presentation sessions generate tracking data via automatic logging of common user behaviors and application events. This includes typical activities like loading a content slide, viewing a video, or navigating a PDF. The Harmony Viewer also extends additional custom tracking functionality to HTML content creators through the SystemBridge JavaScript library.

A typical call session spans one or more presentations and a variety of user interactions and content display. Each call session includes an application generated SessionID, some user specific data (PRID and Country), and some SFA specified HCP data (HCP_ID and ActivityID).

Each tracking event available within the system adheres to the following data structure, though not all events utilize all available fields:

Key Property	Description
Event Type	(Integer) <ul style="list-style-type: none"> Specifies the event type Valid values = Slide, PDF, PopUp, Video, HTML, Map, Briefcase, Pause Presentation, Survey, Custom, Draw, Screen Lock, Orientation Change, Sample Signature
Brand	(String) <ul style="list-style-type: none"> Specifies the brand name associated with the content that spawned the event. If the media item in question doesn't have its own brand identifier, a value may be inferred from the slide or presentation that spawned the event.
AssetID	(String) <ul style="list-style-type: none"> Specifies the AssetID associated with the content that spawned the event. If the media item in question doesn't have its own AssetID, a value may be inferred from the slide or presentation that spawned the event.
ParentID	(String) <ul style="list-style-type: none"> Specifies the AssetID of the parent object of the content that spawned the event.

Title	(String) <ul style="list-style-type: none"> Specifies the Title of the content that spawned the event.
Message	(String) <ul style="list-style-type: none"> Specifies a “Message Delivered” for event. The existence of a “Message” differentiates typical tracking events from events that bear inclusion in external SFA or business reporting systems. Several SystemBridge media launching functions include an optional parameter for inclusion of a Message.
Data	(String) <ul style="list-style-type: none"> A generic string field that is used to house event specific data. Specific details as to data field content depend on the particular event type.
Start Time	(Date) <ul style="list-style-type: none"> The timestamp when the event was spawned.
End Time	(Date) <ul style="list-style-type: none"> The timestamp when the event completed. End Time is valid only for events that imply duration (e.g. Slide views).
HCP Rating	(Integer) <ul style="list-style-type: none"> An optional HCP rating of the slide. Only valid for Slide events. Valid values = null, negative, neutral, positive

Automated Tracking Events

The Harmony Viewer tracks the most common presentation events automatically. These events are all logged to an internal SQLite database and are periodically transmitted to an external data repository on the HarVie Publisher server system. If desired, these exhaustive tracking records can be further transmitted to an external analytics or reporting system.

Automated content/media display events and their relevant properties include:

Event	Key Property	Value
Slide	Brand	Slide.Brand (if defined in slide configuration file) Presentation.Brand (otherwise)
	AssetID	Slide.AssetID
	ParentID	Presentation.AssetID
	Title	Slide.Title
	Message	Slide.Message (if defined in slide configuration file) Slide.AssetID (otherwise)
	Data	N/A
	Start Time	Timestamp when slide first displayed on screen
	End Time	Timestamp when slide is no longer displayed (slide changed or otherwise obscured by another user interface)
	HCP Rating	Rating specified during presentation or in post-call review screen via attitudinal controller
PDF (if launched from a PDF style slide)	Brand	Slide.Brand (if defined in slide configuration file) Presentation.Brand (otherwise)
	AssetID	Slide.AssetID
	ParentID	Presentation.AssetID
	Title	Slide.Title
	Message	Slide.Message (if defined in slide configuration file) Slide.AssetID (otherwise)
	Data	PDF file name
	Start Time	Timestamp when PDF Viewer first displayed on screen
	End Time	Timestamp when PDF Viewer is closed
	HCP Rating	Rating specified on post-call review screen via attitudinal controller
PDF (if launched from an HTML or PopUp slide via SystemBridge command)	Brand	Slide.Brand (if defined in slide configuration file) Presentation.Brand (otherwise)
	AssetID	N/A
	ParentID	Slide.AssetID
	Title	PDF.Title (as specified in SystemBridge function)
	Message	PDF.TrackingMessage (if specified in SystemBridge function)
	Data	PDF file name
	Start Time	Timestamp when PDF Viewer first displayed on screen
	End Time	Timestamp when PDF Viewer is closed
	HCP Rating	Rating specified on post-call review screen via attitudinal controller (only available if Message field contains a valid value)
PDF (if launched from Briefcase interface)	Brand	Asset.Brand (if defined in briefcase asset configuration file) Presentation.Brand (otherwise)
	AssetID	Asset.AssetID
	ParentID	Presentation.AssetID
	Title	Asset.Title
	Message	N/A
	Data	PDF file name

	Start Time	Timestamp when PDF Viewer first displayed on screen
	End Time	Timestamp when PDF Viewer is closed
	HCP Rating	N/A
PDF Page	Brand	Inherited from parent PDF event
	AssetID	Inherited from parent PDF event
	ParentID	Inherited from parent PDF event
	Title	N/A
	Message	N/A
	Data	PDF page number
	Start Time	Timestamp when PDF page changed
	End Time	N/A
	HCP Rating	N/A
PDF Bookmark	Brand	Inherited from parent PDF event
	AssetID	Inherited from parent PDF event
	ParentID	Inherited from parent PDF event
	Title	Bookmark text
	Message	N/A
	Data	PDF page number
	Start Time	Timestamp when PDF bookmark selected
	End Time	N/A
	HCP Rating	N/A
PopUp (if launched from a PopUp style slide)	Brand	Slide.Brand (if defined in slide configuration file) Presentation.Brand (otherwise)
	AssetID	Slide.AssetID
	ParentID	Presentation.AssetID
	Title	Slide.Title
	Message	Slide.Message (if defined in slide configuration file) Slide.AssetID (otherwise)
	Data	PopUp HTML file name
	Start Time	Timestamp when PopUp HTML layer first displayed on screen
	End Time	Timestamp when PopUp HTML layer is closed
	HCP Rating	Rating specified on post-call review screen via attitudinal controller

PopUp (if launched from an HTML or PopUp slide via SystemBridge command)	Brand	Slide.Brand (if defined in slide configuration file) Presentation.Brand (otherwise)
	AssetID	N/A
	ParentID	Slide.AssetID
	Title	N/A
	Message	PopUp.TrackingMessage (if specified in SystemBridge function)
	Data	PopUp HTML file name
	Start Time	Timestamp when PopUp HTML layer first displayed on screen
	End Time	Timestamp when PopUp HTML layer is closed
	HCP Rating	Rating specified on post-call review screen via attitudinal controller (only available if Message field contains a valid value)
Video (if launched from a Video style slide)	Brand	Slide.Brand (if defined in slide configuration file) Presentation.Brand (otherwise)
	AssetID	Slide.AssetID
	ParentID	Presentation.AssetID
	Title	Slide.Title
	Message	Slide.Message (if defined in slide configuration file) Slide.AssetID (otherwise)
	Data	Video file name
	Start Time	Timestamp when Fullscreen Video Viewer first displayed on screen
	End Time	Timestamp when Fullscreen Video Viewer is closed
	HCP Rating	Rating specified on post-call review screen via attitudinal controller
Video (if launched from an HTML or PopUp slide via SystemBridge command)	Brand	Slide.Brand (if defined in slide configuration file) Presentation.Brand (otherwise)
	AssetID	N/A
	ParentID	Slide.AssetID
	Title	N/A
	Message	Video.TrackingMessage (if specified in SystemBridge function)
	Data	Video file name
	Start Time	Timestamp when Fullscreen Video Viewer first displayed on screen
	End Time	Timestamp when Fullscreen Video Viewer is closed
	HCP Rating	Rating specified on post-call review screen via attitudinal controller (only available if Message field contains a valid value)

Video (if launched from Briefcase interface)	Brand	Asset.Brand (if defined in briefcase asset configuration file) Presentation.Brand (otherwise)
	AssetID	Asset.AssetID
	ParentID	Presentation.AssetID
	Title	Asset.Title
	Message	N/A
	Data	Video file name
	Start Time	Timestamp when Fullscreen Video Viewer first displayed on screen
	End Time	Timestamp when Fullscreen Video Viewer is closed
	HCP Rating	N/A
Video Pause	Brand	Inherited from parent PDF event
	AssetID	Inherited from parent PDF event
	ParentID	Inherited from parent PDF event
	Title	N/A
	Message	N/A
	Data	Timecode of current video playback position
	Start Time	Timestamp when pause button was tapped
	End Time	N/A
	HCP Rating	N/A
Video Play	Brand	Inherited from parent PDF event
	AssetID	Inherited from parent PDF event
	ParentID	Inherited from parent PDF event
	Title	N/A
	Message	N/A
	Data	Timecode of current video playback position
	Start Time	Timestamp when play button was tapped
	End Time	N/A
	HCP Rating	N/A
Video Seek	Brand	Inherited from parent PDF event
	AssetID	Inherited from parent PDF event
	ParentID	Inherited from parent PDF event
	Title	N/A
	Message	N/A
	Data	Timecode of current video playback position
	Start Time	Timestamp when seek event was triggered
	End Time	N/A
	HCP Rating	N/A

Video Stop	Brand	Inherited from parent PDF event
	AssetID	Inherited from parent PDF event
	ParentID	Inherited from parent PDF event
	Title	N/A
	Message	N/A
	Data	Timecode of current video playback position
	Start Time	Timestamp when stop event was triggered
	End Time	N/A
	HCP Rating	N/A
HTML (if launched from an HTML or PopUp slide via SystemBridge command)	Brand	Slide.Brand (if defined in slide configuration file) Presentation.Brand (otherwise)
	AssetID	N/A
	ParentID	Slide.AssetID
	Title	HTML.Title
	Message	HTML.TrackingMessage (if specified in SystemBridge function)
	Data	HTML file name
	Start Time	Timestamp when Fullscreen Web Viewer first displayed on screen
	End Time	Timestamp when Fullscreen Web Viewer is closed
	HCP Rating	Rating specified on post-call review screen via attitudinal controller (only available if Message field contains a valid value)
HTML (if launched from Briefcase interface)	Brand	Asset.Brand (if defined in briefcase asset configuration file) Presentation.Brand (otherwise)
	AssetID	Asset.AssetID
	ParentID	Presentation.AssetID
	Title	Asset.Title
	Message	N/A
	Data	HTML file name
	Start Time	Timestamp when Fullscreen Web Viewer first displayed on screen
	End Time	Timestamp when Fullscreen Web Viewer is closed
	HCP Rating	N/A

Additional generic user interface events and their relevant properties include:

Event	Key Property	Value
Map	Brand	Presentation.Brand
	AssetID	N/A
	ParentID	Presentation.AssetID
	Title	N/A
	Message	N/A
	Data	N/A
	Start Time	Timestamp when thumbnail map first displayed on screen
	End Time	Timestamp when thumbnail map is no longer displayed (dismissed or otherwise obscured by another user interface)
	HCP Rating	N/A
Briefcase	Brand	Presentation.Brand
	AssetID	N/A
	ParentID	Presentation.AssetID
	Title	N/A
	Message	N/A
	Data	N/A
	Start Time	Timestamp when Briefcase interface first displayed on screen
	End Time	Timestamp when Briefcase interface is dismissed
	HCP Rating	N/A
Orientation Change	Brand	Presentation.Brand
	AssetID	N/A
	ParentID	Presentation.AssetID
	Title	N/A
	Message	N/A
	Data	New device orientation. Valid values = Portrait, PortraitUpsideDown, LandscapeLeft, LandscapeRight
	Start Time	Timestamp when orientation change occurs
	End Time	N/A
	HCP Rating	N/A
Sample Signature	Brand	Presentation.Brand
	AssetID	N/A
	ParentID	Presentation.AssetID
	Title	N/A
	Message	N/A
	Data	N/A
	Start Time	Timestamp when Sample Signature button tapped
	End Time	N/A
	HCP Rating	N/A
Pause Presentation	Brand	Presentation.Brand
	AssetID	N/A

	ParentID	Presentation.AssetID
	Title	N/A
	Message	N/A
	Data	N/A
	Start Time	Timestamp when presentation is paused (application closed or sent to background)
	End Time	Timestamp when presentation is resumed
	HCP Rating	N/A
Draw	Brand	Slide.Brand
	AssetID	Slide.AssetID
	ParentID	Presentation.AssetID
	Title	Slide.Title
	Message	N/A
	Data	ON (if highlighter is turned on by this interaction) OFF (if highlighter is turned off by this interaction)
	Start Time	Timestamp when highlighter button tapped
	End Time	N/A
	HCP Rating	N/A
Screen Lock	Brand	Slide.Brand
	AssetID	Slide.AssetID
	ParentID	Presentation.AssetID
	Title	Slide.Title
	Message	N/A
	Data	ON (if Screen Lock is turned on by this interaction) OFF (if Screen Lock is turned off by this interaction)
	Start Time	Timestamp when Screen Lock slider changes state
	End Time	N/A
	HCP Rating	N/A

Custom Tracking Messages

The full Harmony Viewer tracking log, containing all of the events discussed in the previous section, is useful for analytics purposes, but it is also common to record a subset of these tracking events to external Sales Force Automation systems to aid in customer tracking activities. The Harmony Viewer tracking system uses the “Message” data field to designate which of the available tracking events should be designated for use in SFA integrations. Any tracking event that contains a valid, non-null, “Message” value will appear both in the exhaustive analytics record (sometimes referred to as click stream data), as well as in a separate message data record.

Within the Harmony Viewer, the only event that is guaranteed to possess a “Message” is the Slide event. A custom message value can be specified for a content slide by adding a “Message” field and corresponding string value to the slide configuration file. If no “Message” field is included for a slide, the slide’s AssetID will be used automatically.

Key	Type	Value
▼ Root	Dictionary	(5 items)
URL	String	index.html
Title	String	3rd Party Apps
Landscape Preview	String	L_preview.png
Portrait Preview	String	P_preview.png
Message	String	Custom Tracking Message

FIGURE 4.0: Sample slide configuration file with “Message” specified

Similarly, all SystemBridge JavaScript functions that launch any of the Harmony Viewer supplemental media controllers (launchFullscreenVideo, launchPDFViewer, launchPDFViewerAtPage, launchFullscreenWebViewer, and launchPopUp) have an optional “trackingMessage” parameter that can be used to specify a “Message” to include in the corresponding tracking events.

Custom Tracking Events

Although the Harmony Viewer can track the most common presentation events in a more or less automated way, there are cases when additional custom events are useful. In particular, it is not possible for the Harmony Viewer to track user interactions that occur within the context of an HTML slide.

To allow for complete flexibility, custom tracking events can be generated from within HTML slide content via the SystemBridge.trackEvent function. This function can be used to generate arbitrary tracking events that correspond to any user interaction that occurs within an HTML slide (e.g. inline video played, slider dragged to a particular position, etc.). As with the automated tracking events, a “Message” field can be specified to trigger inclusion of the event in the message data record, in addition to the standard exhaustive tracking log.

```
<html>
<head>
  <script type="text/javascript" src="resource://js/SystemBridge.js"></script>
</head>
<body>
  <a onclick="SystemBridge.trackEvent('event title', 'event message', 'event data (can be JSON)');">
    Track custom event </a>
</body>
</html>
```

FIGURE: Sample HTML code demonstrating custom tracking event

The custom tracking event and its relevant properties include:

Event	Key Property	Value
Custom	Brand	Slide.Brand (if defined in slide configuration file) Presentation.Brand (otherwise)
	AssetID	N/A
	ParentID	Slide.AssetID
	Title	Custom.Title (as specified in the SystemBridge function)
	Message	Custom.Message (as specified in the SystemBridge function)
	Data	Custom.Data (as specified in the SystemBridge function)
	Start Time	Timestamp when the event was triggered
	End Time	N/A
	HCP Rating	Rating specified on post-call review screen via attitudinal controller (only available if Message field contains a valid value)

Surveys

The Harmony Viewer includes another custom tracking event that is specifically designed for survey data. `SystemBridge.submitSurveyResponse` can be used within HTML content to record survey data a single response at a time by specifying a survey identifier, a question identifier, and a response data value. Complex survey response data can be submitted via a JSON encoded data string. This distinct tracking event type allows survey data to be better isolated from other tracking data for reporting purposes.


```

<html>
<head>
  <script type="text/javascript" src="resource://js/SystemBridge.js"></script>
  <script type="text/javascript">
    function submitValues(form) {
      // process name (name - textField)
      SystemBridge.submitSurveyResponse(form.name, "firstname",
        form.elements["firstname"].value);

      // process question 1 (purpose - checkboxes)
      var chkboxes = document.getElementsByName("purpose")
      var vals = [];
      for (var i = 0; i < chkboxes.length; i++) {
        if (chkboxes[i].checked)
          vals.push(chkboxes[i].value);
      }
      SystemBridge.submitSurveyResponse(form.name, "purpose",
        vals.join(','));

      // notify user that data was submitted
      SystemBridge.alert("Data Submitted", "Thank you.", "OK");

      // clear the form
      form.reset();
    }
  </script>
</head>
<body>
  <h1>Sample Survey</h1>
  <form name="sampleForm">
    <!-- name -->
    <div class="question">Your First Name</div>
    <div class="answer">
      <input name="firstname" type="text">
    </div>

    <!-- ques 1 -->
    <div class="question">What was your purpose for visiting?</div>
    <div class="answer">
      <input type="checkbox" name="purpose" value="specific">
        I was looking for specific information.</input><br>
      <input type="checkbox" name="purpose" value="general">
        I was looking for general information.</input><br>
      <input type="checkbox" name="purpose" value="browsing">
        I was just browsing.</input><br>
      <input type="checkbox" name="purpose" value="purchase">
        I wanted to make a purchase.</input>
    </div>

    <input type="Button" value="Submit" class="submit"
      onClick="submitValues(this.form);" />
  </form>
</body>
</html>

```

FIGURE: Sample code for an HTML form based survey

The survey response event and its relevant properties include:

Event	Key Property	Value
Survey	Brand	Slide.Brand (if defined in slide configuration file) Presentation.Brand (otherwise)
	AssetID	Survey.ID (as specified in the SystemBridge function)
	ParentID	Slide.AssetID
	Title	Survey.Question (as specified in the SystemBridge function)
	Message	N/A
	Data	Survey.Answer (as specified in the SystemBridge function)
	Start Time	Timestamp when the event was triggered
	End Time	N/A
	HCP Rating	N/A

Dynamic Presentation Customization

This is pending finalization in a subsequent release. Requires the SFA client integration scheduled to be available from April 2013.

Briefcase Assets

This is pending finalization in a subsequent release. Requires the SFA client integration scheduled to be available from April 2013.

Testing and Deployment

This is pending finalization. The business processes for testing and deployment are to be continually refined based on experience from the first and subsequent releases of the Harmony Viewer.

Content Templates

This is pending finalization. The content templates are to be continually refined based on experience from the first and subsequent releases of the Harmony Viewer for different brands.

An example content package demonstrating the features of the Harmony Viewer is available with this release of the Technical Content Development Guide.

Content Checklist

This is pending finalization. The checklist will highlight key features of the content that must be checked to validate the content has been created consistently with these guidelines.

SystemBridge Function Reference

SystemBridge.js is a JavaScript library, developed exclusively for the Harmony Viewer, to simplify interactions between HTML based content and supplemental obj-C functionality. The SystemBridge library allows HTML content creators to execute a wide range of native functionality without resorting to direct modification of Obj-C code.

The following SystemBridge JavaScript functions are provided within the Harmony Viewer application for use within HTML based presentation content:

- Content Navigation
 - SystemBridge.goToSlide(slideTitle)
 - SystemBridge.goToAlternateCall(callTitle, slideTitle)
 - SystemBridge.slideToLeftPage()
 - SystemBridge.slideToRightPage()
 - SystemBridge.slideToAbovePage()
 - SystemBridge.slideToBelowPage()
- Modal Media Views
 - SystemBridge.launchFullscreenVideo(videoPath, trackingMessage)
 - SystemBridge.launchPDFViewer(localPath, title, trackingMessage)
 - SystemBridge.launchPDFViewerAtPage(localPath, title, pageNumber, trackingMessage)
 - SystemBridge.launchFullscreenWebViewer(localPath, title, trackingMessage)
 - SystemBridge.launchPopUp(localPath, trackingMessage)
 - SystemBridge.closeTopPopUp()
- Data Storage/Access
 - SystemBridge.setSessionValue(key, value)
 - SystemBridge.requestSessionValue(key, callback)
 - SystemBridge.setPersistentValue(key, value)
 - SystemBridge.requestPersistentValue(key, callback)
- Tracking
 - SystemBridge.trackEvent(title, message, data)
 - SystemBridge.submitSurveyResponse(survey, question, answer)
- Miscellaneous Utility Functions
 - SystemBridge.setViewFrame(viewName, xPos, yPos, width, height, animated)
 - SystemBridge.setViewProperties(properties)
 - SystemBridge.enableSwipeToPage()
 - SystemBridge.disableSwipeToPage()
 - SystemBridge.pageLoadComplete()
 - SystemBridge.executeJavaScriptInMainView(jsCode)
 - SystemBridge.requestCurrentSlideInfo(callback)
 - SystemBridge.alert(title, message, button)

Content Navigation

SystemBridge.goToSlide(slideTitle)

Loads a new content slide. The slide's HTML properties are specified in the slide configuration pList file or the presentation configuration pList file.

Parameters:

- **slideTitle** – (string - required) Title of a content slide, as specified in the presentation configuration pList file. Subsection content can be specified by concatenating the titles with "::" (e.g. Efficacy -> Section1 = "Efficacy::Section1").

SystemBridge.goToAlternateCall(callTitle, slideTitle)

Loads one of the alternate call storyflows that have been predefined in the presentation configuration file. The change of storyflow will update the presentation breadcrumb display and corresponding gesture navigation. If a slide title is specified, the system will navigate to that slide after the active call change.

Parameters:

- **callTitle** – (string - required) Title of a predefined alternate call storyflow, as specified in the presentation configuration pList file. Specifying a call title of "ActiveCall" will return to the predefined ActiveCall.
- **slideTitle** – (string - optional) Title of a content slide, as specified in the presentation configuration pList file. Subsection content can be specified by concatenating the titles with "::" (e.g. Efficacy -> Section1 = "Efficacy::Section1"). If the slide title is not specified, the alternate call will load to the first slide specified for the storyflow.

SystemBridge.slideToLeftPage()

This forces the slide currently to the left of the active screen to slide into the active onscreen location. This programmatically mimics the result of the user performing a left swipe touch gesture.

SystemBridge.slideToRightPage()

This forces the slide currently to the right of the active screen to slide into the active onscreen location. This programmatically mimics the result of the user performing a right swipe touch gesture.

SystemBridge.slideToAbovePage()

This forces the slide currently to the top of the active screen to slide into the active onscreen location. This programmatically mimics the result of the user performing an up swipe touch gesture.

SystemBridge.slideToBelowPage()

This forces the slide currently to the bottom of the active screen to slide into the active onscreen location. This programmatically mimics the result of the user performing a down swipe touch gesture.

Modal Media Views

SystemBridge.launchFullscreenVideo(videoPath, trackingMessage)

Launches the specified video in the standard iOS full-screen movie controller.

Parameters:

- **videoPath** – (string - required) Path to a video file, specified relative to the slide's root content directory.
- **trackingMessage** – (string - optional) Custom tracking message to be included in the tracking log.

SystemBridge.launchPDFViewer(localPath, title, trackingMessage)

Loads a PDF file into a full-screen PDF Viewer controller.

Parameters:

- **localPath** – (string - required) Path to a PDF file, specified relative to the slide's root content directory.
- **title** – (string - required) Title that appears in the header bar of the PDF Viewer.
- **trackingMessage** – (string - optional) Custom tracking message to be included in the tracking log.

SystemBridge.launchPDFViewerAtPage(localPath, title, pageNumber, trackingMessage)

Loads a PDF file into a full-screen PDF Viewer controller to a specified page.

Parameters:

- **localPath** – (string - required) Path to a PDF file, specified relative to the slide's root content directory.
- **title** – (string - required) Title that appears in the header bar of the PDF Viewer.
- **pageNumber** – (integer - required) page number of PDF to display on load
- **trackingMessage** – (string - optional) Custom tracking message to be included in the tracking log.

SystemBridge.launchFullscreenWebView(localPath, title, trackingMessage)

Loads a URL into a full-screen Web Viewer controller. This control will display any file that is supported by the Mobile Safari browser, including HTML, DOC, XLS, PPT, and PDF.

Parameters:

- **localPath** – (string - required) Path to local content to display, specified relative to the slide's root content directory.
- **title** – (string - required) Title that appears in the header bar of the Web Viewer.
- **trackingMessage** – (string - optional) Custom tracking message to be included in the tracking log.

SystemBridge.launchPopUp(localPath, trackingMessage)

Launches a full-screen Pop-Up view containing a Web Viewer with transparent background for displaying HTML content. The Web Viewer's bounds are always set to fill the entire screen regardless of user interface orientation. Since all HTML pop-ups are loaded on top of the existing presentation slide and navigation layer, they must include a way for the user to execute SystemBridge.closeTopPopUp() to dismiss the pop-up.

Parameters:

- **localPath** – (string - required) Path to local content to display, specified relative to the slide's root content directory.
- **trackingMessage** – (string - optional) Custom tracking message to be included in the tracking log.

SystemBridge.closeTopPopUp()

Closes the most recently launched full-screen PopUp view.

Data Storage/Access

SystemBridge.setSessionValue(key, value)

Stores a key value pair for later reference during the same presentation session. Once the application is closed, all session key value pairs are cleared from memory.

Parameters:

- **key** – (string - required) Unique key that can be used to query this session value later.
- **value** – (string - required) Value to associate with the specified key.

SystemBridge.requestSessionValue(key, callback)

Queries the system for a previously stored session value. When the system is ready to respond to the query, the specified callback function will be executed with a single argument containing the result (e.g. "callback(value);").

Parameters:

- **key** – (string - required) Unique key that can be used to query the system for a specific session value.
- **callback** – (string - required) JavaScript callback function to be executed with the system response.

SystemBridge.setPersistentValue(key, value)

Stores a key value pair for later reference. Persistent key value pairs are NOT cleared from memory when the application is closed.

Parameters:

- **key** – (string - required) Unique key that can be used to query this session value later.
- **value** – (string - required) Value to associate with the specified key.

SystemBridge.requestPersistentValue(key, callback)

Queries the system for a previously stored persistent value. When the system is ready to respond to the query, the specified callback function will be executed with a single argument containing the result (e.g. "callback(value);").

Parameters:

- **key** – (string - required) Unique key that can be used to query the system for a specific session value.
- **callback** – (string - required) JavaScript callback function to be executed with the system response.

Tracking

SystemBridge.trackEvent(title, message, data)

Adds a time-stamped custom tracking event with the specified parameters to the active presentation's tracking log.

Parameters:

- **title** – (string - required) String to be included in the title field of the custom tracking event.
- **message** – (string - required) String to be included in the message field of the custom tracking event.
- **data** – (string - required) String to be included in the data field of the custom tracking event. To pass complex data with this event, a JSON encoded string can be used.

SystemBridge.submitSurveyResponse(survey, question, answer)

Adds a time-stamped survey response event with the specified parameters to the active presentation's tracking log.

Parameters:

- **survey** – (string - required) Survey identification string. This string should be unique enough to differentiate the survey for reporting and analytics purposes.
- **question** – (string - required) Survey question identifier. Since the survey identifier should be unique, the question string can be a unique identifier or the actual question text.
- **answer** – (string - required) Survey response. To pass complex data with this event, a JSON encoded string can be used.

Miscellaneous Utility Functions

SystemBridge.setViewFrame(viewName, xPos, yPos, width, height, animated)

Changes the position and/or size of a specified navigation layer view with optional animated transition. This function only applies to custom navigation layers that have been properly configured with view names specified in its NavigationOutlets.h file.

Parameters:

- **viewName** – (string - required) Name identifier for the navigation layer view.
- **xPos** – (integer - required) Horizontal position of the upper left corner of the view.
- **yPos** – (integer - required) Vertical position of the upper left corner of the view.
- **width** – (integer - required) Width of the view.
- **height** – (integer - required) Height of the view.
- **animated** – (boolean - optional) If "true", the view frame is animated into its new position over 0.25 seconds. If "false", the view frame is moved instantly, with no animated transition.

SystemBridge.setViewProperties(properties)

Applies an arbitrary set of view property changes to the specified navigation layer views. Any property that is available to the native view specified can be set via this method (e.g. control text, color, visibility, etc.). This function only applies to custom navigation layers that have been properly configured with view names specified in its NavigationOutlets.h file.

Parameters:

- **properties** – (array - required) Array of JavaScript objects specifying the view name and property set to be modified. Each object should contain a “name” parameter, specifying the view name, and a “properties” object containing sets of key:value pairs for the property and values to be applied to said view.

Example:

```
var views = new Array();
views[0] = { "name": "breadcrumb",
            "properties": {
                "breadcrumbMainColor": "#8F64B9",
                "breadcrumbSubColor": "#CEAEDC",
                "breadcrumbSelectedColor": "#F8D66C",
                "breadcrumbLineColor": "#68367F"
            }
        };
views[1] = { "name": "slideTitle",
            "properties": {
                "textColor": "#F8D66C"
            }
        };
SystemBridge.setViewProperties(views);
```

SystemBridge.disableSwipeToPage()

Disables swipe navigation in the presentation. Can be useful to force a user to interact with certain slide content before re-enabling swiping, permitting access to subsequent content.

SystemBridge.enableSwipeToPage()

Re-enables swipe navigation, if previously disabled via a previous call to SystemBridge.disableSwipeToPage().

SystemBridge.pageLoadComplete()

If the current slide content has been configured to delay its initial display via the “Delay Display” configuration field, this function call notifies the system that the slide content is now ready for display.

SystemBridge.executeJavaScriptInMainView(jsCode)

Executes arbitrary JavaScript code within the HTML slide that is currently active.

Parameters:

- **jsCode** – (string - required) JavaScript code to execute.

SystemBridge.requestCurrentSlideInfo(callback)

Queries the system for information about the currently active slide. When the system is ready to respond to the query, the specified callback function will be executed with a single argument containing the result (e.g. "callback(pageInfo;"). This function is useful for HTML Pop-Ups that are referenced from multiple slides but contain information that might change slightly based on the context of display (e.g. Important Safety Information).

Parameters:

- **callback** – (string - required) JavaScript callback function to be executed with the system response. The callback function will be executed with a single parameter containing a JavaScript object with the following data keys:
 - Title
 - Brand
 - ContentLocation
 - AssetID
 - SectionName
 - SubSection
 - Status
 - Message

SystemBridge.alert(title, message, button)

Displays a standard iOS alert dialog with the specified title text, message text, and a single button with the specified button text. This function is preferable to the standard JavaScript alert function because it allows more control over the dialog text.

Parameters:

- **title** – (string - required) Text to be used in the alert dialog's top title field.
- **message** – (string - required) Text to be used in the alert dialog's main body.
- **button** – (string - required) Text to be used in the alert dialog's dismissal button.