SharePlus™
POWERED BY INFRAGISTICS

Launchpads Developer Guide

Version 1.2

SharePlus™ 4.0 Launchpads Developer Guide

Table of Contents

How to Use this Guide

This guide was formatted with the reader in mind, including not only illustrative images and diagrams but also elements like notes and links, in order to highlight/redirect to relevant information.

Note: Notes include information that needs to be highlighted, and sometimes tips for the reader.

About Tables	Details
Importance	Tables add value for the user by presenting complex data in a user-friendly and more readable format.

Gesture icons provide a close-to-reality representation for applications with touch-based UI.

Introducing the Launchpads SDK

Overview

The Launchpads SDK allows you enhance SharePlus user experience by integrating web technologies like HTML, CSS, AND JavaScript into the native User Interface. With that goal in mind, SharePlus provides rich launchpads that can be used as Homes, either offering a customized view of a given site (Site Homes) or displaying content for a SideBar module (Application Home). SharePlus Application Launchpads are developed leveraging modern standard client-side web technologies like HTML5 and jQuery. Because of this, high quality interactions can be achieved by using CSS and JavaScript frameworks. In addition, Application Launchpads can communicate with SharePlus by two means:

- Invoking native SharePlus actions by using **S+ links.**
- Reading SharePlus data using the **JavaScript API.**

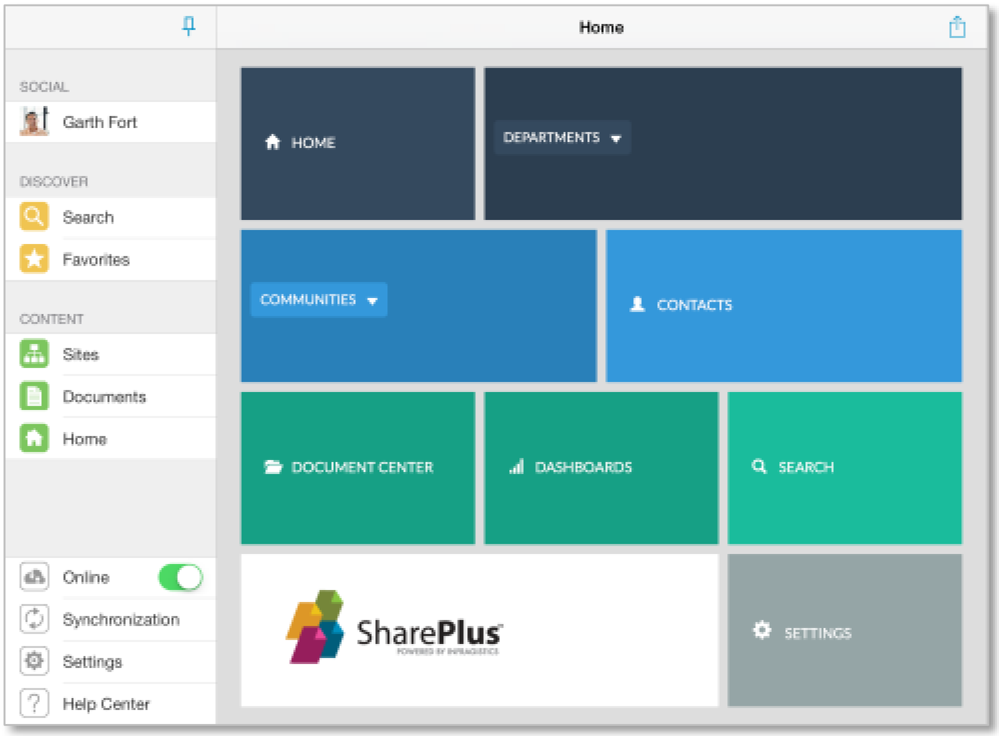
Main Features

The following table summarizes the main features of Application Launchpads.

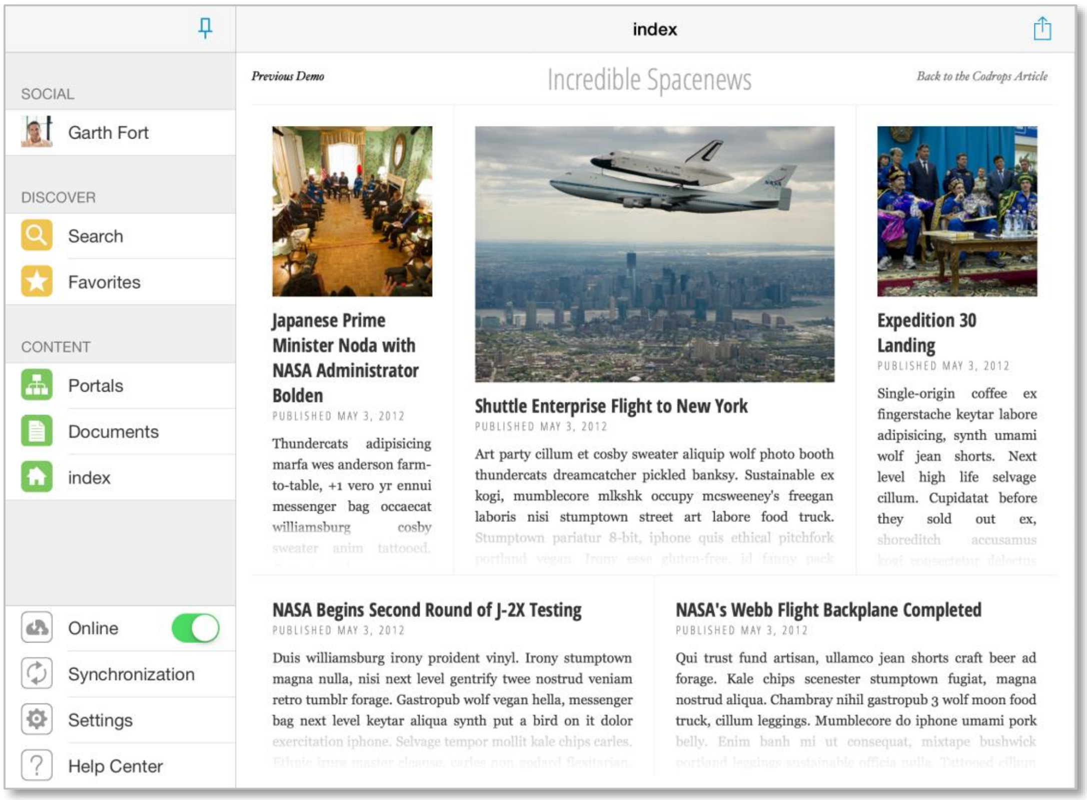
Feature	Description
Custom Home Screen	You can enhance your application's UX, allowing access to personalized Home content.
Launchpads Flexibility	Easily manage the location, configuration settings, and navigation of your launchpad.
Short Learning Curve	The learning curve is really short, you will make rapid progress.
JavaScript Support	Access SharePlus functionalities through JavaScript.

Custom Home Screen

Building an HTML launchpad to be used as Home is especially powerful. When combined with SharePlus URL Schemes, you can create a rich HTML page with markup, images, CSS, and virtually any content you want to display. This launchpad can contain links to different locations in the application to navigate or open SharePoint content. In addition, high quality interactions can be achieved by using CSS and JavaScript frameworks like JQuery.



The sample below, inspired by Flipboard, includes JQuery code that gives a different look to the page layout, letting you navigate pages with a swipe gesture. Data is being retrieved from SharePoint through the SharePlus JavaScript API.



Japanese Prime Minister Noda with NASA Administrator Bolden

PUBLISHED MAY 3, 2012

Thundercats adipisicing marfa wes anderson farm-to-table, +1 vero yr ennui messenger bag occaecat williamsburg cosby sweater anim tattooed.

Shuttle Enterprise Flight to New York

PUBLISHED MAY 3, 2012

Art party cillum et cosby sweater aliquip wolf photo booth thundercats dreamcatcher pickled banksy. Sustainable ex kogi, mumblecore mlkshk occupy mcsweeney's freegan laboris nisi stumptown street art labore food truck. Stumptown pariatur 8-bit, iphone quis ethical pitchfork portland vegan. Irony esse gluten-free, id fanny pack

Expedition 30 Landing

PUBLISHED MAY 3, 2012

Single-origin coffee ex fingerstache keytar labore adipisicing, synth umami wolf jean shorts. Next level high life selvage cillum. Cupidatat before they sold out ex, shoreditch accusamus

NASA Begins Second Round of J-2X Testing

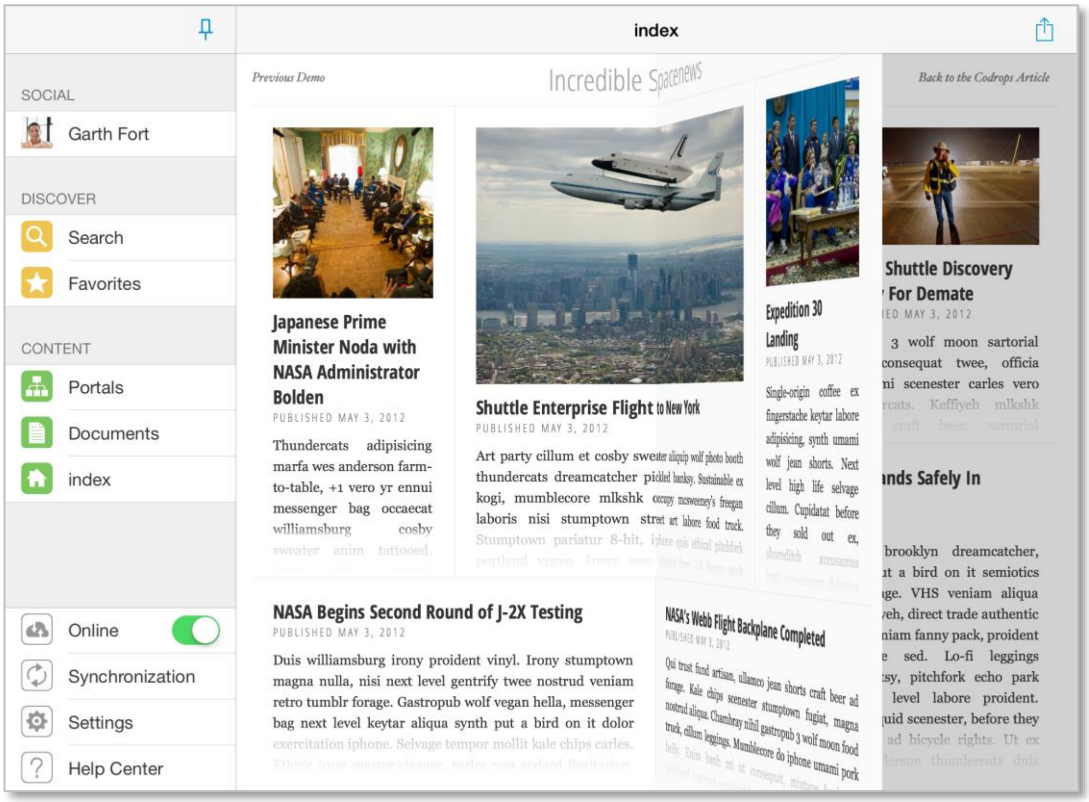
PUBLISHED MAY 3, 2012

Duis williamsburg irony proident vinyl. Irony stumptown magna nulla, nisi next level gentrify twee nostrud veniam retro tumblr forage. Gastropub wolf vegan hella, messenger bag next level keytar aliqua synth put a bird on it dolor exercitation iphone. Selvage tempor mollit kale chips carles. Ethnic irure master cleanse, carles non godard flexitarian.

NASA's Webb Flight Backplane Completed

PUBLISHED MAY 3, 2012

Qui trust fund artisan, ullamco jean shorts craft beer ad forage. Kale chips scenester stumptown fugiat, magna nostrud aliqua. Chambray nihil gastropub 3 wolf moon food truck, cillum leggings. Mumblecore do iphone umami pork belly. Enim banh mi ut consequat, mixtape bushwick portland leggings sustainable officia nulla. Tattooed cillum



Shuttle Discovery For Demate

ED MAY 3, 2012

3 wolf moon sartorial consequat twee, officia ni scenester carles vero rcats. Keffiyeh mlkshk craft beer sartorial

nds Safely In

brooklyn dreamcatcher, it a bird on it semiotics ge. VHS veniam aliqua eh, direct trade authentic niam fanny pack, proident e sed. Lo-fi leggings sy, pitchfork echo park level labore proident. uid scenester, before they ad bicycle rights. Ut ex lerson thundercats duis

Launchpads Flexibility

Working with URL Schemes, JavaScript APIs, and the configuration settings of Application Launchpads you can achieve many tasks:

Remote deployment

Launchpads can be deployed using a centralized location, allowing a quick and easy distribution to all company devices.

Access SharePlus Native Interface

Use **URL Schemes** to call different actions within SharePlus and present the results on SharePlus native interface.

Extend your launchpad with SharePlus data

Use the **set of SharePlus JavaScript APIs** to access the SharePlus application, retrieve data from SharePoint, and display the information in your launchpad.

Short Learning Curve

The Application Launchpads feature is not difficult to learn and you don't need further knowledge besides HTML5 and JavaScript. Also, you can use Application Launchpads in combination with URL Schemes and in that scenario you don't even need JavaScript.

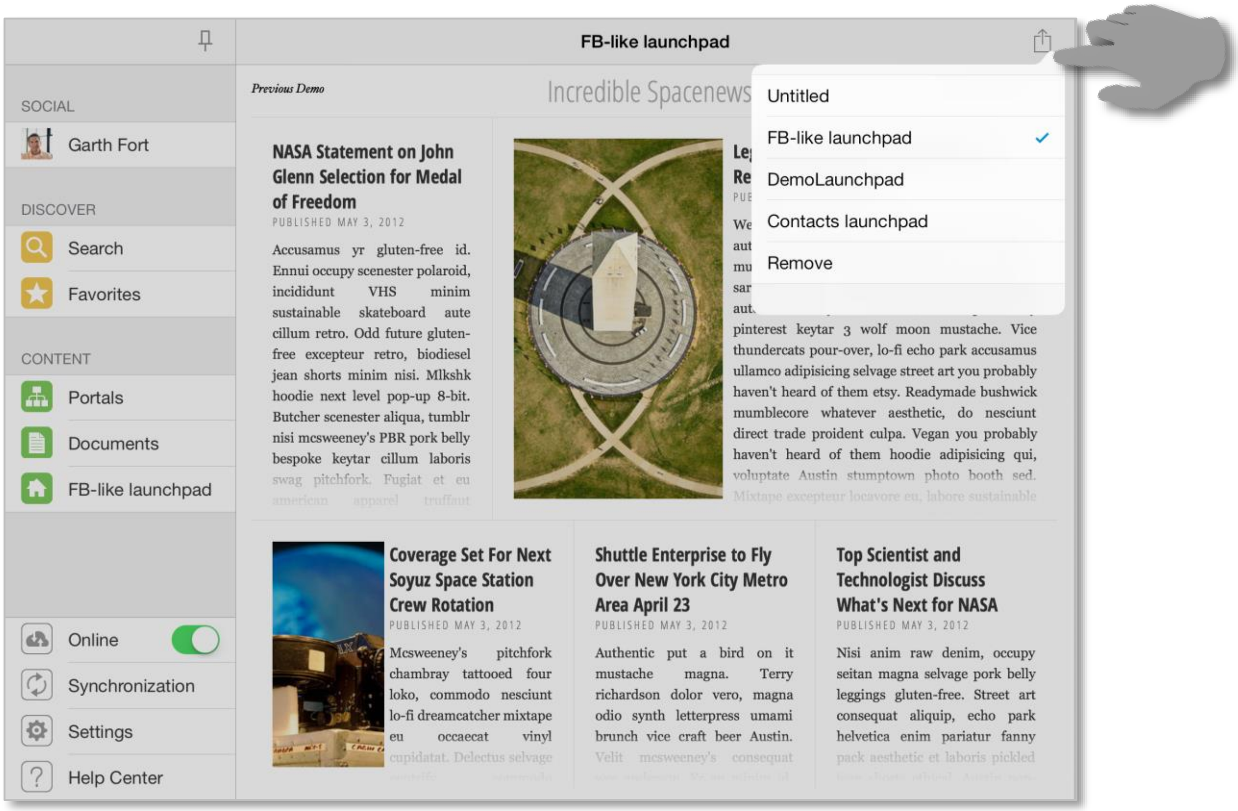
JavaScript Support

A set of pre-defined API functions can be accessed through a bridge file (SharePlusBridge) which makes the connection between your code and SharePlus. The **SharePlusBridge** will help you develop meaningful behavior with your implementation. Manage the configuration settings, check the connection status, change your launchpad at run-time, or download application resources from an URL.
For further information see the **Introduction** section from Application Launchpads API.

Homes and Application Launchpads

Home content

"Home" refers to the content displayed when accessing a Site Home or the Application Home within SharePlus application. You can configure different types of content to be displayed as Home, i.e., Application Launchpads, ReportPlus dashboards, and PDF files. In addition, a Home can have more than one content assigned, in those cases you are able to switch between different content by using a selector.



NASA Statement on John Glenn Selection for Medal of Freedom
PUBLISHED MAY 3, 2012
Accusamus yr gluten-free id. Ennui occupy scenester polaroid, incididunt VHS minim sustainable skateboard aute cillum retro. Odd future gluten-free excepteur retro, biodiesel jean shorts minim nisi. Mlkshk hoodie next level pop-up 8-bit. Butcher scenester aliqua, tumblr nisi mcsweeney's PBR pork belly bespoke keytar cillum laboris swag pitchfork. Fugiat et eu american apparel truffaut

pinterest keytar 3 wolf moon mustache. Vice thundercats pour-over, lo-fi echo park accusamus ullamco adipisicing selvage street art you probably haven't heard of them etsy. Readymade bushwick mumblecore whatever aesthetic, do nesciunt direct trade proident culpa. Vegan you probably haven't heard of them hoodie adipisicing qui, voluptate Austin stumptown photo booth sed. Mixtape excepteur locavore eu, labore sustainable

Coverage Set For Next Soyuz Space Station Crew Rotation
PUBLISHED MAY 3, 2012
Mcsweeney's pitchfork chambray tattooed four loko, commodo nesciunt lo-fi dreamcatcher mixtape eu occaecat vinyl cupidatat. Delectus selvage

Shuttle Enterprise to Fly Over New York City Metro Area April 23
PUBLISHED MAY 3, 2012
Authentic put a bird on it mustache magna. Terry richardson dolor vero, magna odio synth letterpress umami brunch vice craft beer Austin. Velit mcsweeney's consequat

Top Scientist and Technologist Discuss What's Next for NASA
PUBLISHED MAY 3, 2012
Nisi anim raw denim, occupy seitan magna selvage pork belly leggings gluten-free. Street art consequat aliquip, echo park helvetica enim pariatur fanny pack aesthetic et laboris pickled

Application Launchpads as Home content

SharePlus provides rich launchpads that can be displayed as home content, either customizing Site Homes or the Application Home.

Application Home – Home content can be displayed in SharePlus when accessing the Application Home module in the SideBar. This module can be loaded by default when opening the application from scratch.

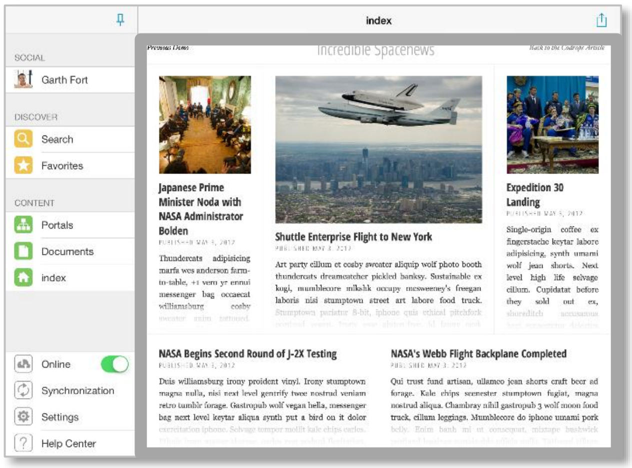
Site Homes – Portals and sites can both display home content in SharePlus, presenting the user with a customized view for a given site. Site Homes can be pre-configured and shared across sites, displaying different content depending on the SharePoint site's context.

Launchpad canvas size

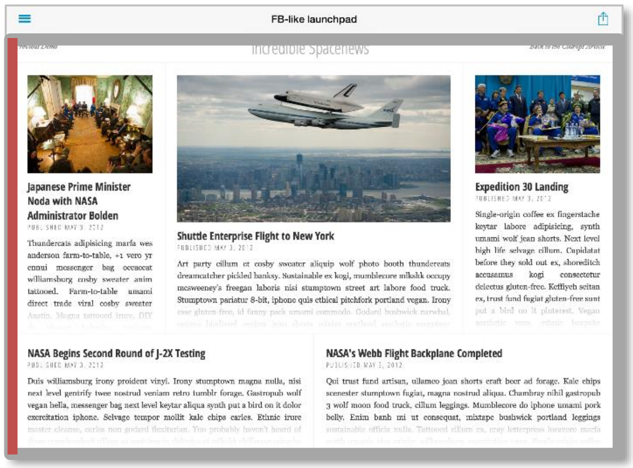
The canvas size where the launchpad is displayed depends on:

- The device orientation
- Whether the SideBar is hidden or not

Landscape Orientation

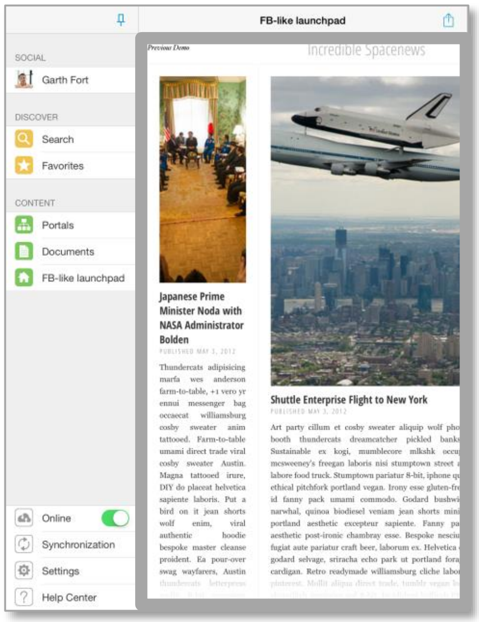

index
SOCIAL
Garth Fort
DISCOVER
Search
Favorites
CONTENT
Portals
Documents
index
Online
Synchronization
Settings
Help Center
Incredible Spacenews
Japanese Prime Minister Noda with NASA Administrator Bolden
Shuttle Enterprise Flight to New York
Expedition 30 Landing
NASA Begins Second Round of J-2X Testing
NASA's Webb Flight Backplane Completed

1608 x 1408 px

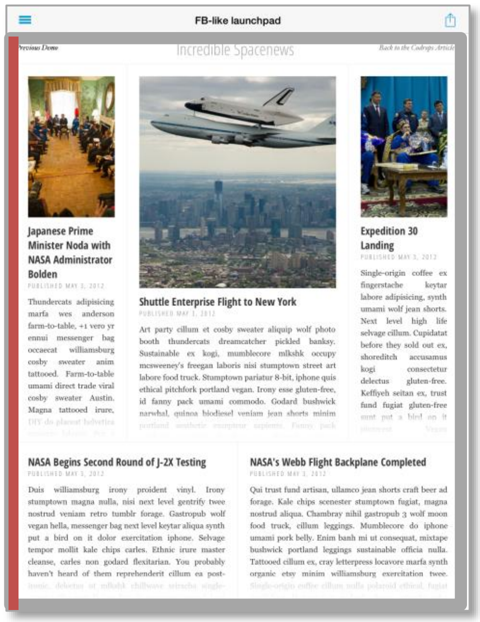

FB-like launchpad
Incredible Spacenews
Japanese Prime Minister Noda with NASA Administrator Bolden
Shuttle Enterprise Flight to New York
Expedition 30 Landing
NASA Begins Second Round of J-2X Testing
NASA's Webb Flight Backplane Completed

2048 x 1408 px

Portrait Orientation



1095 x 1919 px


FB-like launchpad
Incredible Spacenews
Japanese Prime Minister Noda with NASA Administrator Bolden
Shuttle Enterprise Flight to New York
Expedition 30 Landing
NASA Begins Second Round of J-2X Testing
NASA's Webb Flight Backplane Completed

1536 x 1919 px

Launchpad canvas limitations

In SharePlus you can switch between a visible and hidden SideBar. Because of that, a section of the canvas is always reserved for the swipe gesture that displays a hidden SideBar. That section consists of a 20 px width column located on the left side of the canvas and it is highlighted in red in the images above.
Take into account that launchpads cannot process gestures within that section.

Introduction

Application Launchpads can be displayed as home content, either customizing Site Homes or the Application Home. A specific file (HTML, webarchive, etc.) is loaded as the launchpad source. The file can be downloaded from a specified static URL or stored inside the SharePlus application. Application Launchpads interact with SharePlus by invoking the application using S+ links or reading SharePlus data using the JavaScript API.

Custom HTML page

Using an HTML page combined with SharePlus **URL Schemes** allows you to include markup, images and CSS (following certain restrictions dictated by the device). This page can contain links to different parts in the application using **URL Schemes** to navigate.

Note: When creating a launchpad with more than just HTML code (images, sounds, videos), you need to use the webarchive file format.

Remote Deployment

Your launchpad can be downloaded from a **static URL** specified in the Configuration File. The URL location must be **public or** otherwise located **under the same Domain as the Configuration File** to share its credentials. The launchpad can be available offline, and will later synchronize with the URL every time you start SharePlus. Cache is used to avoid downloading the same resource that is referenced by the URL every time. The launchpad is downloaded again when there are changes, though.

Creating your First Application Launchpad

In this procedure you will create a new Application Launchpad, a **custom HTML page** that allows you to navigate to different parts of the SharePlus application, **using URL Schemes**.

Overview

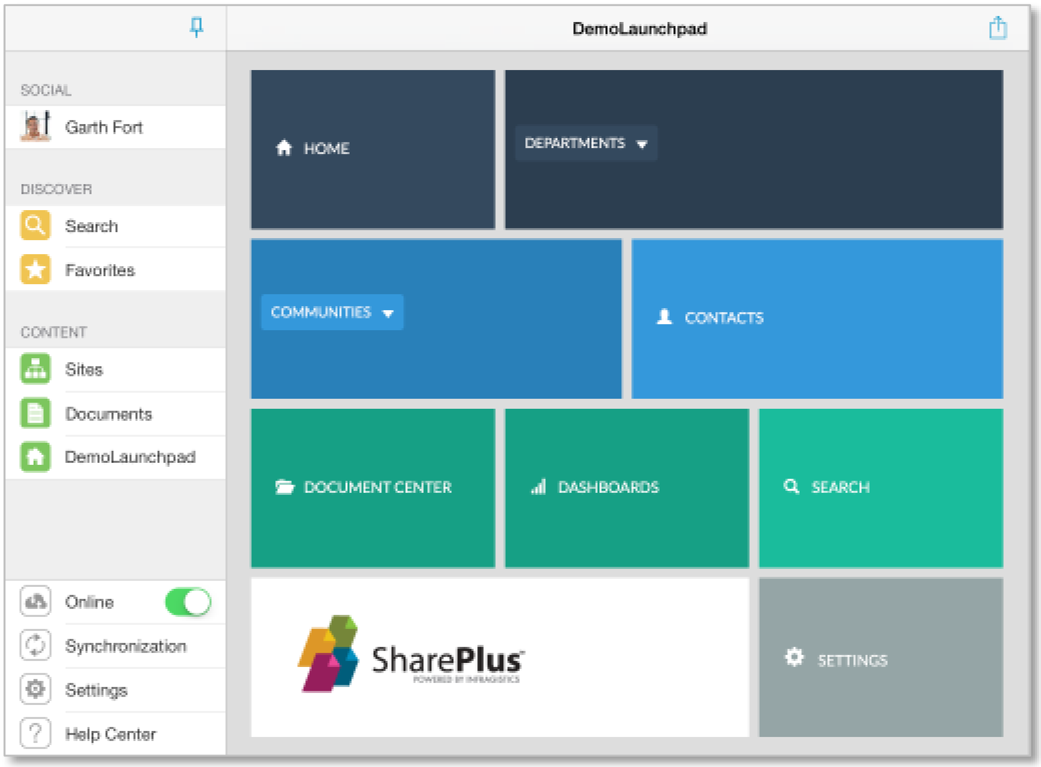
The process has the following steps:

1. Creating a custom launchpad
2. Preparing your launchpad for SharePlus
3. Displaying your launchpad within SharePlus

Steps

1. **Create a custom launchpad**

   a. **Start with a custom HTML home page**
   Create a custom HTML home page that contains a few buttons. These buttons will then navigate to different parts of the application, using URL schemes.
   After adding markup, images, and CSS, your custom page may look similar to the following one:



   b. **Add behavior to your HTML page**
   Using URL Schemes you add behavior to your page by navigating or invoking actions within SharePlus. Below you will find a few buttons with some possible customizations for the launchpad displayed above:

   - Opening SharePlus Settings.

```
<a class="button" href="splus://?action=settings">SETTINGS</a>
```

   - Navigating to a static URL

```
<a class="button" href="splus://MySharePointPortal">HOME</a>
```

   The navigation to static URLs is recommended. When navigating to a dynamic URL (one that includes JavaScript or PHP scripts) you won't be able to return back to your launchpad.

   - Navigating to a Document Library

```
<a class="button" href="splus://MySharePointPortal/Site/Documents">DOCUMENT CENTER</a>
```

- Opening an Document

```
<a class="button" href="splus://
MySharePointPortal/Site/Documents/sampleDoc.docx?action=viewdocument">DASHBOARDS</a>
```

For further details about URL Schemes refer to Application Integration in *SharePlus Administrator Guide.*

2. Prepare your launchpad for SharePlus

a. **Create a ZIP file**

Launchpads packaging and deployment can be done using ZIP files.

i. Add files to the compressed file.
You need all the required files for your launchpad to work, including HTML, JavaScript, CSS, images, and other assets.

ii. Get the ZIP file ready.
Name the file using the **.web.zip** extension. E.g.: DemoLaunchpad.web.zip. Later in this procedure, you will configure your launchpad using the *Add as Home* action through the UI.

For further details refer to the Launchpads Deployment section.

b. **Upload your launchpad to SharePoint**

Add your launchpad to a document library, making it accessible from the SharePlus application.

Note: Launchpad packaging using ZIP files is only available for SharePlus 4.0.4 or later versions. The Webarchive file format can be used in previous versions.

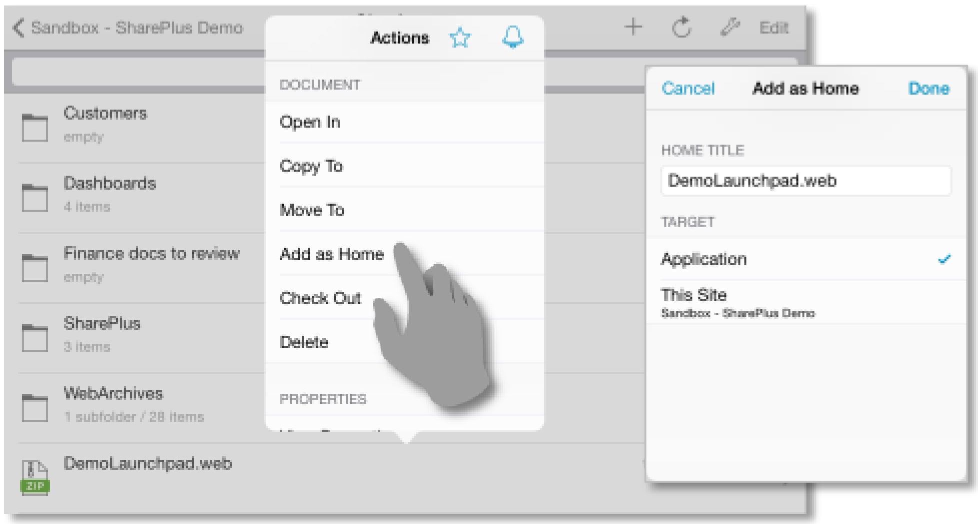
3. Display your launchpad within SharePlus.

a. **Configure your launchpad as the Application Home**

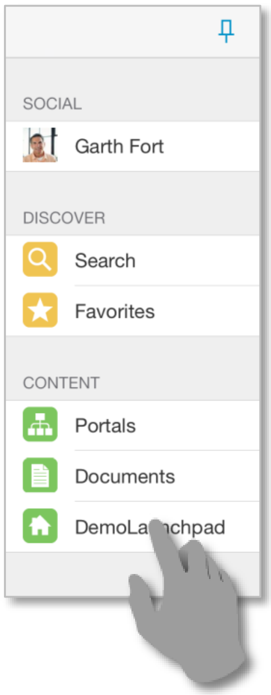
You can manually configure a launchpad as the Application Home through the User Interface.

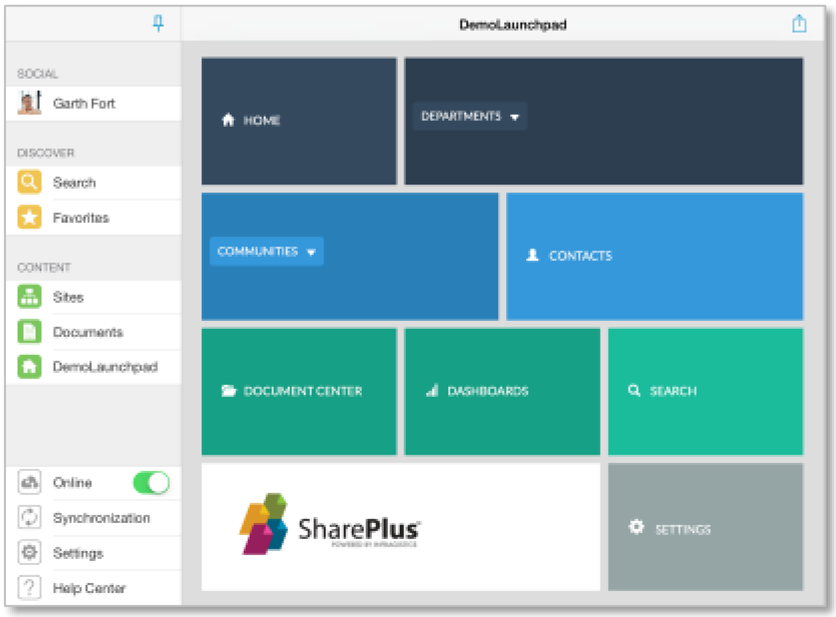
i. Access the actions menu for the launchpad item.
Tap & hold over the launchpad to open the *Actions* menu and select the *Add as Home* action.

ii. Choose where to display your launchpad.
You can choose between two targets, Application (Application Home) or This Site (the site where the item is located).



b. **Access your launchpad**
Tap over the Application Home module in the SideBar to display your launchpad.





Launchpads Deployment

Your launchpads can be displayed as home content in different areas of the SharePlus application. The Launchpads deployment process consists on preparing the deployment package and later configuring the package as home content.

Preparing Launchpads for Deployment

Launchpads packaging can be done using:

- ZIP files - Only available in SharePlus 4.0.4 or later versions
- Webarchive files – To create webarchives you need to use Apple's Safari app.

Note: When available, ZIP files are strongly recommended over Webarchive files, as the latter format may present issues with JavaScript code.

The ZIP Deployment Package

The compressed file should contain all the required files for your launchpad to work, including HTML, JavaScript, CSS, images, and other assets. SharePlus searches the ZIP file for the main HTML page to be displayed, both *.html* and *.htm* extensions are supported. The following steps describe how the ZIP file is processed by SharePlus:

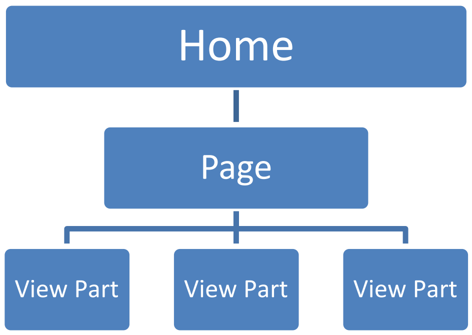
1. SharePlus searches the main folder for a file named *index.html* , *index.htm* , *default.html* or *default.htm.*
2. If no file was found, SharePlus searches the main folder for any file with *.html* or *.htm* extensions.
3. If both searches were unsuccessful, SharePlus searches the files in the first subfolder available.

Note: To manually configure ZIP files in-app (using the *Add as Home* action), you need to name this files using the **.web.zip** extension. E.g.: myLaunchpad.web.zip

Configuring Launchpads as Home Content

About Home Content

Home content is specified in the Configuration File by creating and configuring Pages, which include one or more elements called View Parts. View Parts are in charge of showing content on the screen when the Home is loaded.



Understanding Pages and View Parts

In SharePlus, pages are not HTML web pages from a web site. A SharePlus Page is a different matter, it has a name for identification and it is used as a container of elements (View Parts).
Pages are created in the Configuration File and they display Home content by including one or more View Parts. Each View Part can display an Application Launchpad, ReportPlus Dashboard, or a PDF file. When having more than one View Part, the user can swap between them through the User Interface.

View Part
• Application Launchpads
• ReportPlus Dashboards
• PDF files

View Parts

They are in charge of displaying content inside a Page, which is displayed on screen when a Home is loaded. There are two types of View Parts OOTB, Application Launchpads and ReportPlus Dashboards. New View Part controllers can be added and configured for a custom application using SharePlus Native SDK.

Configuration Summary

This section will help you set up your launchpads as home content for different areas of the SharePlus application.
You can configure a launchpad to be displayed when accessing the Application Home module in the SideBar or any site (Site Home)
The following table lists the available scenarios, additional scenario information is available after the following table.

Scenario	Details	Configuration Method
In-App Configuration	Configure a Site Home or the Application Home through the UI.	Actions menu – Add as Home
Application Home	Create a Page holding the launchpad and reference the PageName in the HomeManagement feature.	Configuration File – Pages, HomeManagement
Site Home	Add an entry to the MobileNavigation list and include the URL to the launchpad.	Site Configuration a.k.a. "MobileNavigation"
Default Home - Portal and sites	Configure the default page settings to be used across all the sites of a portal.	Configuration File – HomeManagement

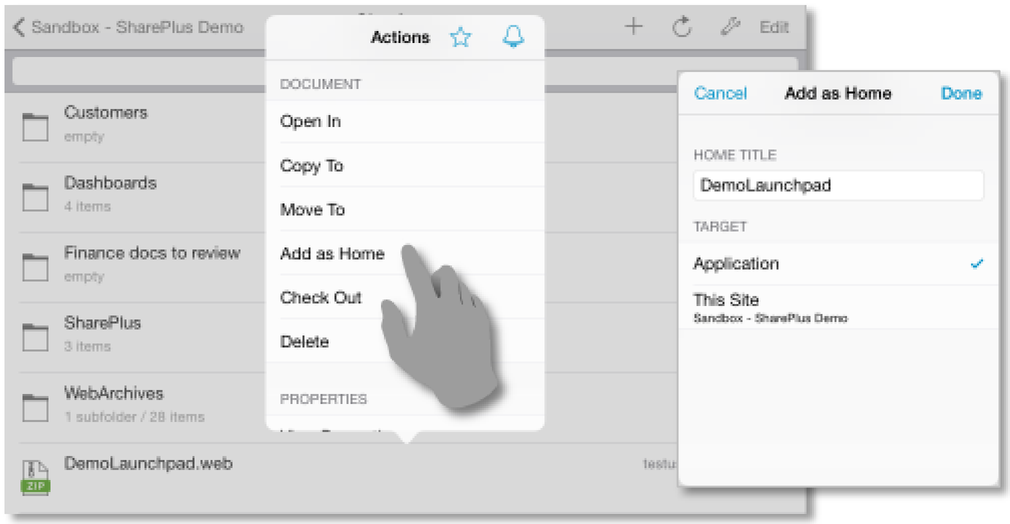
In-App Configuration

You are able to manually configure a launchpad for a Site Home or the Application Home module through the User Interface.

Steps

1. **Access the Actions menu**
   Tap & hold over a launchpad to open the *Actions* menu and select the *Add as Home* action.

2. **Then choose where to display your launchpad**
   You can choose between two targets, Application (Application Home module) or This Site (the site where the item is located).



Application Home

The Application Home will be displayed when tapping the module in the SideBar. You can pre-configure this Home in the Configuration File using an already created Page.

Steps Overview

1. Creating a Page that includes an Application Launchpad
2. Configuring your launchpad for the Application Home

Steps

1. **Create a Page that includes an Application Launchpad**
   Define a new page in the Configuration File or use an existing one. You need an Application Launchpad View Part to be specified for the page.
   a. Open the Configuration File.
      Navigate to the Pages section to create a new Page.

b. Define a Page.

```
<key>Pages</key>
<array>
      <dict>
            <key>PageName</key>
            <string>customPageName</string>
            <key>ShowSelector</key>
            <false/>
            <key>ViewParts</key>
            <array>
            ...
            </array>
      </dict>
</array>
```

The three elements are:

- **PageName** – The name used to reference the page when configuring a Home for the application or sites.
- **ShowSelector** – Enables/Disables the possibility to select between the different View Parts from a page/view container. Does nothing when there is only one View Part specified.
- **ViewParts** – Array of the View Part elements defined for the page.

c. Specify a View Part containing an Application Launchpad.
Each page has an array of view Part elements that you must specify.

```
<key>ViewParts</key>
<array>
      <dict>
            <key>ViewControllerID</key>
            <integer>0</integer>
            <key>Title</key>
            <string>customApplicationLaunchpad</string>
            <key>Settings</key>
            <dict>
                  <key>Source</key>
                  <string>Documents/LaunchpadSample.html</string>
                  <key>SourceType</key>
                  <integer>1</integer>
            </dict>
      </dict>
</array>
```

The elements are:

- **ViewControllerID** – Specifies the ID defined in the app.plist for the View Controller to be used. For Application Launchpads and PDF files the value specified must be 0.
- **Title** – Specifies the title to be displayed on the top toolbar for this View Part.

- **Source** – Specifies the source location of the launchpad. The source can be an URL or a local path inside the application.
- **SourceType** – Sets the source type of the Application Launchpad. Possible values are:
  - **0 –** to reference an URL to download the file
  - **1 –** to reference a path inside the application, in the application's resource bundle or Local Files if the path starts with "Documents/".

Note on Sources:

- When loading the launchpad from local files, the path must start with "Documents/". This is a useful approach for debugging and testing new launchpads.
- When downloading the launchpad from a static URL, the URL location must be public or otherwise located under the same Domain than the Configuration File to share its credentials.

2. Configure your launchpad for the Application Home

a. Open the Configuration File.
Navigate to the *HomeManagement* feature in the Features section, to set up your launchpad.

b. Reference the Page
You need to reference the page you created by name in the *AppPageName* key value.
The Configuration File should be similar to:

```
<key>HomeManagement</key>
<dict>
       <key>Enabled</key>
       <true/>
       <key>Settings</key>
       <dict>
              ...
              <key>AppPageName</key>
              <string>customPageName</string>
              <key>ShowMainAreaButton</key>
              <false/>
              <key>ShowNavigatorButton</key>
              <true/>
              ...
       </dict>
</dict>
```

The relevant elements are:

- **AppPageName** – Defines the page name to be used as Application Home. The page name must match the name of an existing page in the Pages section.
- **ShowMainAreaButton** – When true, shows the Application Home button permanently at the top of the Main Screen or "Working Surface".
- **ShowNavigatorButton** – When false, hides the Application Home button in the SideBar.

Site Home

When working with Site Configuration (a.k.a. "MobileNavigation"), the *Home* property allows you to set launchpads as home content for a site. In addition, you are able to configure one or more launchpads for the same site.
Your Site Home can contain an Application Launchpad, a PDF document, a ReportPlus dashboard or any content that can be rendered in a web browser.

Column	Type	Description
Home	Single line of text	A list of values separated by comma. Possible values are: • An URL - to reference a resource to be loaded. • A page name – referencing a *PageName* key value defined in the *Pages* section of the Configuration File. • "default" - to reference the default SharePlus site view, which is the native view that displays sub-sites, libraries, and lists.
HomeTitle	Single line of text	A list of titles separated by comma, for each of the Home values specified in the *Home* field respectively.

Note: The "default" value in the Home column will reference the *DefaultSitePageName* key value if this value was manually changed in the Configuration File.

For further details about the MobileNavigation list refer to Site Configuration aka "MobileNavigation" section in *SharePlus Administrator Guide*.

Default Home – Portal and sites

In SharePlus, Portals can present a pre-configured launchpad that can be shared with its sites. In this scenario, different content is displayed by the launchpad depending on the site's context.
You can configure a Site Home including a launchpad to be used by default for a portal and all its sites. To achieve this, Portals' Site Homes need to be pre-configured using Pages and Sites in the Configuration File instead of using the MobileNavigation list.

Steps Overview

1. Creating a Page that includes an Application Launchpad
2. (*Optional*) Enabling the default list view for the Portal
3. Configuring your launchpad for the Portal's Home
4. Defining the Site Home by default (*DefaultSitePageName*)
5. Adjusting inheritance settings (*InheritParentPage*)

Steps

1. **Create a Page that includes an Application Launchpad**
   Define a new page in the Configuration File or use an existing one. You need an Application Launchpad View Part to be specified for the page. For details on how to specify a Page and a View Part including a launchpad see the Application Home scenario in this section.

2. **(*Optional*) Enable the default list view for the Portal**
   When pre-configuring a launchpad as Portal Home, the default content is replaced by the launchpad.
   A View Part with SharePlus default list view needs to be added, so you can switch between content (default list view and the launchpad) by using the selector.

> **Note:** SharePlus default list view shows all the sub-sites, libraries, and lists for portals or sub-sites. The bottom bar also includes the site´s Favorites.

In the Configuration File, the Page with the launchpad should include the following View Part:

```
<key>ViewParts</key>
<array>
      <dict>
            ...
            <key>Settings</key>
            <dict/>
            <key>ViewControllerID</key>
            <integer>2</integer>
            <key>Title</key>
            <string> </string>
            ...
      </dict>
</array>
```

The elements are:

- **ViewControllerID** – Specifies the ID defined in the app.plist for the View Controller to be used. For the default list view the integer value specified must be 2.
- **Title** – Specifies the title to be displayed on the top toolbar for this View Part.

3. **Configure your launchpad for the Portal's Home**
   a. Open the Configuration File.
      Navigate to your Portal configuration in the Sites section, to set up the launchpad to be used.

   b. Reference the Page in the pre-configured Portal
      You need to reference the page you created by name (*PageName* key value).

The Configuration File should be similar to:

```
<key>Sites</key>
<array>
	<dict>
		...
		<key>PageName</key>
		<string>yourCustomPage</string>
		...
	</dict>
</array>
```

4. **Define the Site Home by default (DefaultSitePageName)**
    a. Navigate to the *HomeManagement* feature in the Features.
    b. Reference an existing page by name (*PageName* key value).

   A sample Configuration File snippet illustrating this step is included on the next step.

5. **Adjust inheritance settings (InheritParentPage).**
   Go to *HomeManagement* to enable inheritance between a pre-configured parent site and its sub-sites.
   The *HomeManagement* feature in the Configuration File should be similar to:

```
<key>HomeManagement</key>
<dict>
	<key>Enabled</key>
	<true/>
	<key>Settings</key>
	<dict>
		...
		<key>DefaultSitePageName</key>
		<string>yourCustomPage</string>
		<key>InheritParentPage</key>
		<true/>
		...
	</dict>
</dict>
```

The existing elements are:

- **DefaultSitePageName** – References the page to be used as Site Home by default. The page name must match the name of an existing page in the Pages section.
- **InheritParentPage** – When true, sub-sites inherit pre-configured pages from their parent site.

Getting Started with the API

Introduction

A bridge file called SharePlusBridge allows **communication between SharePlus and your Application Launchpads.** SharePlusBridge.js is a JavaScript file that includes a number of pre-defined APIs that you will use to achieve specific and controlled behavior.

SharePlusOnLoad

When an Application Launchpad has finished loading, a function is called to notify that the launchpad is ready and JavaScript functions are enabled. That function is *SharePlusOnLoad* and it allows you, for example, to load any custom settings that your launchpad may need.

Integrating the SharePlusBridge to your Application Launchpad

Following this procedure you will add functionality to a launchpad displaying contact items from a list in SharePoint. You will use one of the pre-defined set APIs (*List.getItems*) and learn how to make the connection from your launchpad to the bridge.

Overview

The process has 4 required steps:

1. Enabling the bridge in the Configuration File
2. Add a custom HTML page to be used as launchpad
3. (*Optional*) Including the *SharePlusOnLoad* function
4. Adding JavaScript to your launchpad
5. (*Optional*) Checking connection and URL availability
6. Displaying your launchpad within SharePlus

Steps

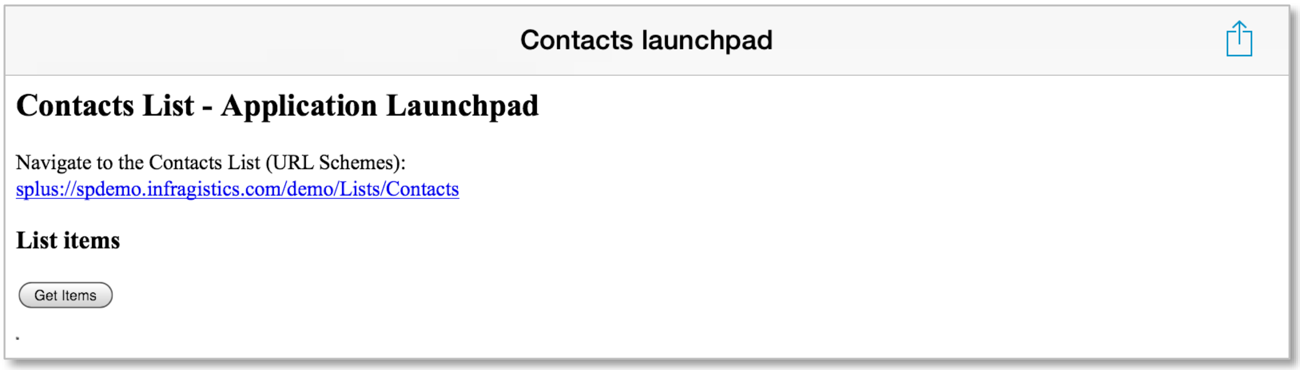
1. **Enable the bridge in the Configuration File**
    a. Open the Configuration File.
    b. Navigate to the WebDashboard feature
    c. Enable the *BridgeEnabled* property
    This is necessary as you will use the SharePlusBridge in this example.

2. **Add a custom HTML page to be used as launchpad**
    In this example, you will create a launchpad that communicates with **List.getItems** API function to retrieve information from the Contacts list in SharePoint's Demo site. These contact items will be displayed in a table in your custom launchpad. The URL Scheme to navigate to the list within SharePlus is also included.

The JavaScript code can be invoked from a button's event, as shown in the following code snippet:

```
<html>
	<head>
		<script type=text/javascript>
		//JavaScript code to be included next…
		</script>
	</head>
	<body>
		<h2>Contacts List – Application Launchpad</h2>
		Navigate to the Contacts List (URL Schemes):<br>
		splus://spdemo.infragistics.com/demo/Lists/Contacts<br>

		<h3>List items</h3>
		<button onclick="javascript:getListItems()">Get Items</button><br>
		<br>
		<table id="listItemsTable" border="1">
		</table>
	</body>
</html>
```

The HTML page used as Application Launchpad should look similar to this:



Contacts List - Application Launchpad

Navigate to the Contacts List (URL Schemes):
splus://spdemo.infragistics.com/demo/Lists/Contacts

List items

3. (*Optional*) Include the *SharePlusOnLoad* function
This function is called to notify that the launchpad is loaded and JavaScript functions are enabled. You could include JavaScript code here to add logic that needs to be executed before the launchpad is displayed. For example, you can load custom settings or check if there is a working connection as shown below.

```
function SharePlusOnLoad(){
	SPlus.Connection.isConnected (function (connected){
		If not connected { SPlus.Utility.showMessage ('Connection Status','Not
		Connected')};
	});
};
```

4. Adding JavaScript to your launchpad

You will call the *List.getItems* method from the API, in order to get items from a SharePoint list.

a. Get the required information ready.

The method receives the following parameters:

- URL to the SharePoint site
- Name of the SharePoint list view
- Array of field values to be retrieved from each item
- A success callback function that receives an Array with the items.
- An error callback function that receives a JavaScript Object with the error description
- A cancel callback function that receives a JavaScript Object with relevant information

Syntax of the API method:

```
SPlus.List.getItems(listUrl, viewName, fieldsValuesArray, onSuccess, onError, onCancel)
```

The code for specifying the *URL*, *viewName*, and *fields* parameters should be similar to the following code snippet:

```
var listUrl = 'http://spdemo.infragistics.com/demo/Lists/Contacts';
var viewName = 'All Contacts';
var fieldsArray = ['ows_Title','ows_FirstName','ows_Email','ows_Company'];
```

b. Call the *List.getItems* API method.

From your JavaScript code, you will call the bridge's List.getItems method, passing all required parameters including the implementation of the callback functions. Should be similar to the following code snippet:

```
SPlus.List.getItems(listUrl, viewName, fieldsValuesArray, function (items) {

       //Loop through all the existing items
       for(var i = 0; i < items.length; i++) {
              //Get a reference to the item and all its fields
              var item = items[i];
              var title = item['ows_Title#displayValue'];
              ...

              //TODO: Add the item fields to the table in the launchpad
       }
}, function (errorResponse) {
       SPlus.Utility.showMessage('ERROR: SPlus.List.getItems',
errorResponse['error#displayValue']);
}, function (cancelResponse) {
       SPlus.Utility.showMessage('CANCEL: SPlus.List.getItems', cancelResponse);
});
```

Complete JavaScript code to be invoked from the button should be similar to the following code snippet:

```
function getListItems() {
     var viewName = 'All Contacts';
     var listUrl = 'http://spdemo.infragistics.com/demo/Lists/Contacts';
     var fieldsValuesArray = ['ows_Title','ows_FirstName','ows_Email','ows_Company'];

     SPlus.List.getItems(listUrl, viewName, fieldsValuesArray, function (items) {
          //Get a reference to the HTML table and create the Header row
          var tableId = "listItemsTable";
          var itemsTable = document.getElementById(tableId);
          var headerRow = document.createElement("tr");

          //Loop through the fieldValuesArray
          for(var f = 0; f < fieldsValuesArray.length; f++) {
               //Add the headers (ows_Title, ows_FirstName, etc.) to the Header row
               var field = fieldsValuesArray[f];
               var headerCol = document.createElement("th");
               headerCol.innerHTML = field;
               headerRow.appendChild(headerCol);
          }
          //Add the Header row to the table
          itemsTable.appendChild(headerRow);

          //Loop through all the existing items
          for(var i = 0; i < items.length; i++) {
               //Get a reference to the item and create the Item row
               var item = items[i];
               var itemRow = document.createElement("tr");

               //Loop through all the fields for an item
               for(var f = 0; f < fieldsValuesArray.length; f++) {
                    //Add the field values to the Item row
                    var field = fieldsValuesArray[f];
                    var fieldName = field;
                    var fieldValue = item[fieldName];
                    var colField = document.createElement("td");
                    colField.innerHTML = fieldValue;
                    itemRow.appendChild(colField);
               }
               //Add every Item row to the table
               itemsTable.appendChild(itemRow);
          }
     }, function (errorResponse) {
       SPlus.Utility.showMessage('ERROR: SPlus.List.getItems',
errorResponse['error#displayValue']);
     }, function (cancelResponse) {
       SPlus.Utility.showMessage('CANCEL: SPlus.List.getItems', cancelResponse);
     });
}
```

In the code above, the *Utility.showMessage* API method is called from the error callback function. For further details, see the **Utility.showMessage** API method.

For details about *List.getItems* in the API Reference, go to **List.getItems**.

5. **(*Optional*) Check connection and URL availability**
   The use of Connection.isConnected and Utility.URL.isAvailable API methods is strongly recommended in some scenarios. The code should be similar to the code snippets included below:

```
SPlus.Connection.isConnected ( function (connected) {
    If not connected { SPlus.Utility.showMessage ('Connection Status','Not Connected')};
});
```

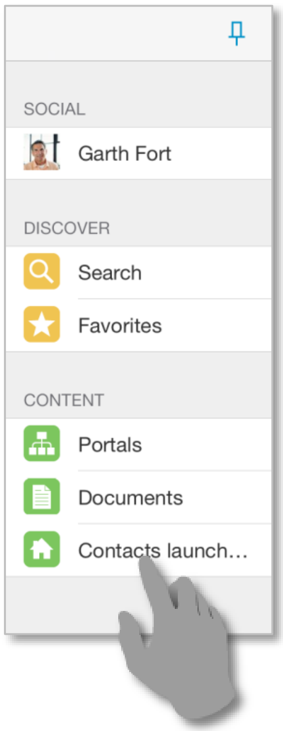
   Please notice that you can work without the need of an internet connection. When working in offline mode and having all required resources already downloaded, you may work offline and synchronize your changes later.

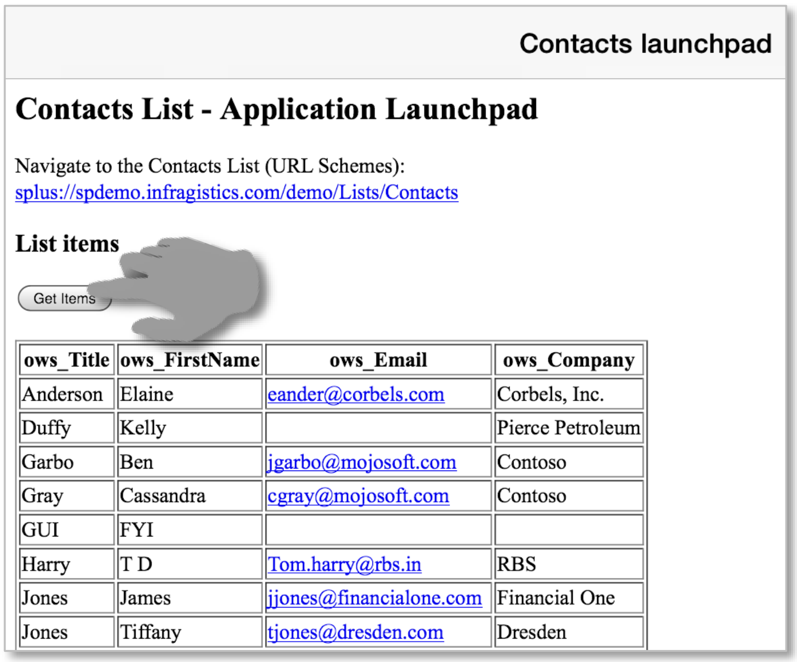
```
var url = 'http://spdemo.infragistics.com/demo';

SPlus.Utility.URL.isAvailable(url, function (available) {
    If not available { SPlus.Utility.showMessage ('URL Status, 'URL is not available')
}, function (errorResponse){
SPlus.Utility.showMessage ('Is Url Available Error',errorResponse['error#displayValue']);
}, function (cancelResponse) {

});
```

   Take into account that with no available connection the URL will never be available as well.

6. **Display your launchpad within SharePlus**
   a. Upload the launchpad to SharePoint and configure it as the Application Home.
      For details about this refer to the last step of Creating your First Application Launchpad.

   b. Access the Application Home module in the SideBar.





Contacts List - Application Launchpad

Navigate to the Contacts List (URL Schemes):
splus://spdemo.infragistics.com/demo/Lists/Contacts

List items

ows_Title	ows_FirstName	ows_Email	ows_Company
Anderson	Elaine	eander@corbels.com	Corbels, Inc.
Duffy	Kelly		Pierce Petroleum
Garbo	Ben	jgarbo@mojosoft.com	Contoso
Gray	Cassandra	cgray@mojosoft.com	Contoso
GUI	FYI		
Harry	T D	Tom.harry@rbs.in	RBS
Jones	James	jjones@financialone.com	Financial One
Jones	Tiffany	tjones@dresden.com	Dresden

API Scenarios

This section is your gateway to important conceptual and task-based information that will help you use the various JavaScript functions and functionalities provided by SharePlusBridge.

Summary

The following table lists some of the available scenarios using JavaScript functions included in the SharePlusBridge file. Additional scenario information is available after the following summary table.

Scenario	Details	API functions
Persisting Application Launchpad Settings	The launchpad settings can be modified, stored in SharePlus, and later loaded when needed to.	*WebDashboard.Settings.save, WebDashboard.Settings.load*
Changing an Application Launchpad	The launchpad can be changed permanently or temporary, according to your needs.	*WebDashboard.setSource, WebDashboard.navigate*
Getting Items from a list, using a query	You can get the items from a SharePoint list by specifying a detailed query.	*List.getListWithOptions*

Persisting Application Launchpad Settings

Overview

The *WebDashboard.Settings.save* and *WebDashboard.Settings.load* JavaScript API methods allow you to manage dynamic settings within the launchpad. Settings can be stored in SharePlus under a specific key, making them persistent, and later they can be loaded again when the launchpad is reloaded.

Code

In the code snippet below, you use the custom JavaScript function **saveSettings** to:

- Create the settings *JSON Object*
- Specify the launchpad's key
- Call the *WebDashboard.Settings.save* API from the bridge to store the settings under a unique *key*.

```
var stringValue = ‘string content‘;
var numericValue = 15;

function saveSettings() {
    var settings = {};
    settings['stringValue’] = stringValue;
    settings['numericValue’] = numericValue;

    var key = 'LaunchpadOne'; // unique key, identifies launchpad settings to be stored
    SPlus.WebDashboard.Settings.save(settings, key, function () {

    }, function (responseError) {
        SPlus.Utility.showMessage('Save Settings Error', responseError['error#displayValue']);
    });
}
```

The callback functions shown above for *WebDashboard.Settings.save* are:

- A success callback function, which receives no parameters and may not be implemented as shown above.
- An error callback function, which receives a *JavaScript object* with the error description.

Custom launchpad settings stored in SharePlus can be loaded later. In the code snippet below, you use the custom JavaScript function **loadSettings** to call the *WebDashboard.Settings.load* API from the bridge. The API function loads previously stored settings identifying them through a unique *key*.

```
function loadSettings() {
    var key = 'LaunchpadOne'; // Key used to identify the launchpad settings to be loaded

    SPlus.WebDashboard.Settings.load(key, function (settings) {
        if (settings) {
            stringValue = settings['stringValue'];
            numericValue = settings['numericValue'];
        }
    }, function (responseError) {
        SPlus.Utility.showMessage('Load Settings Error', responseError['error#displayValue']);
    });
}
```

The callback functions shown above for *WebDashboard.Settings.load* are:

- A success callback function, which receives a *JSON object* with the configuration settings loaded for the launchpad.
- An error callback function, which receives a *JavaScript object* with the error description.

Changing the source of an Application Launchpad

Overview

The WebDashboard.setSource and WebDashboard.navigate JavaScript API methods allow you to change your launchpad. Both methods require the same parameters and they are used in the same way, but they allow you to achieve a slightly different result.

With *WebDashboard.setSource*:
Set a new source for your Application Home launchpad that will be persistent even if SharePlus is shut down.

With *WebDashboard.navigate*:
You can jump to another launchpad temporarily. The source change will remain until SharePlus is shut down or you jump to another launchpad.

Code

In the code snippet below, you:

- Specify the static URL to download the launchpad.
- Set the *sourceType* property to specify that the source will be an URL.
- Call the *WebDashboard.setSource* API from the bridge to set the new permanent source of the launchpad.

```
var source = ‘http://spdemo/demo/SiteAssets/WebArchives/DemoLaunchpad.webarchive‘;
var sourceType = 0;

function setDashboardSource() {
    SPlus.WebDashboard.setSource (source, sourceType, function () {

    }, function (responseError) {
        SPlus.Utility.showMessage('Save Settings Error', responseError['error#displayValue']);
    });
}
```

The callback functions shown above are:

- A success callback function, which receives no parameters and may not be implemented.
- An error callback function, which receives a *JavaScript object* with the error description.

Note: The *WebDashboard.setSource* API function only works for launchpads included in the Application Home.

Getting items from a list, using a query

Overview

The List.getItemsWithOptions JavaScript API method allows you to retrieve the items from a SharePoint list, also specifying a custom Object to query the list. The Array of items retrieved is returned to the success callback function. These items are Objects whose properties correspond to the requested fields.

Code

Before calling the *List.getItemsWithOptions* API method, you need to **get ready the following parameters**:

- URL to the SharePoint site
- Custom JavaScript Object with information to be used to query to the SharePoint list.

You need to use the Query schema of CAML to define queries against the contents of a SharePoint list. The **structure of the *options* Object** is the following:

Property	Type	Description
viewName	String	Name of the SharePoint List view.
fields	Array	Array of field values to be retrieved for each item
where	String	Where condition used in the query.
orderBy	String	OrderBy condition used in the query.
queryOptions	XML node	XML node with several properties used in the query.

The code for **specifying the url and options Object** should be similar to the following code snippet:

```
var listUrl = 'http://spdemo.infragistics.com/demo/Lists/Team%20Discussion';

var options = new Object();
options.viewName = 'All Documents';
options.fields = ['ows_Title', 'ows_Created'];
options.where = '<Or>' +
                   '<Contains>' +
                         '<FieldRef Name="FileLeafRef"/>' +
                         '<Value Type="File">SharePlus</Value>'+
                   '</Contains>' +
                   '<Contains>' +
                         '<FieldRef Name="FileLeafRef"/>' +
                         '<Value Type="File">ReportPlus</Value>'+
                   '</Contains>' +
                '</Or>';
options.orderBy = '<FieldRef Name="FileSizeDisplay"></FieldRef>';
options.queryOptions = '<QueryOptions>' +
                         '<IncludeMandatoryColumns>TRUE</IncludeMandatoryColumns>' +
                         '<DateInUtc>TRUE</DateInUtc>' +
                      '</QueryOptions>';
```

For further details, see the Lists.GetListItems Method MSDN topic.

Now you are ready to **call the List.getItemsWithOptions API method**, your code should be similar to the following code snippet:

```
SPlus.List.getItemsWithOptions (listUrl, options, function (items) {
    //Loop the items Array
    for (var i=0; i < items.length; i++) {
       //Getting a JSON object with the requested fields (item)
        var item = items[i];
       //Getting the fields for an item
        var title = item['ows_Title#displayValue'];
        var body = item['ows_Body#displayValue'];
        var created = item['ows_Created#displayValue'];
        // TODO: Display items somewhere in the launchpad
    }
}, function (errorResponse) {
    SPlus.Utility.showMessage('Get Items Error',errorResponse['error#displayValue']);
    //TODO: Implement how to handle errors
}, function (cancelResponse) {
       //TODO: Implement how to handle a user’s cancel
});
```

Appendix 1: Restrictions & Considerations

The following tables summarize the restrictions and considerations for the SharePlus 4.0 release.

Restrictions

Feature	Description
Dynamic URLs	When navigating to a dynamic URL (one that includes JavaScript or PHP scripts) you won't be able to return back to your Application Launchpad.

Considerations

What is it about?	Consideration
Launchpad source file	The launchpad source file must be any file that can be loaded by iOS' native Web View. These are the available source options: • A downloadable file from a static URL specified in the *config.plist* file • A file in the Local Files folder • A file in the application's resource bundle
iPhone startup screen	On iPhone, SharePlus can be configured to start with a module including an Application Launchpad, but as with any other module, the SideBar will be hidden due to space constraints.
Launchpad source API function	The *WebDashboard.setSource* API function only works for launchpads included in the Application Home.

Appendix 2: API Reference

The API Reference Guide offers a complete list of all available methods when working with Application Launchpads.

API Reference Table

Category	API Property	Introduced in version
Context	Current.siteTitle	4.0
	Current.siteUrl	4.0
	Current.webTitle	4.0
	Current.webUrl	4.0

Category	API Method	Introduced in version
Web	Web.getWebsAndLists	3.9
List	List.getItems	3.9
	List.getItemsWithOptions	3.9
	List.Item.download	3.9
User	User.getUser	3.9
	User.getProfileByName	3.9
Web Dashboard	WebDashboard.Settings.save	3.9
	WebDashboard.Settings.load	3.9
	WebDashboard.setSource	3.9
	WebDashboard.reload	3.9
	WebDashboard.navigate	3.9
Utility	Utility.getLocalizedString	3.9
	Utility.showMessage	3.9
	Utility.URL.open	3.9
	Utility.URL.download	3.9
Configuration	Configuration.setRemoteFileSource	3.9
	Configuration.getFeature	3.9
	Configuration.getVariables	3.9
Connection	Connection.isOnline	3.9
	Connection.isConnected	3.9

Context.Current.siteTitle

This property is used to get the title of the top-level site (portal).
The launchpad must be assigned to a portal or one of its sites. When the launchpad is assigned to the Application Home, an empty value is retrieved.

Syntax

```
SPlus.Context.Current.siteTitle;
```

Parameters

None

Context.Current.siteUrl

This property is used to get the URL of the top-level site (portal).
The launchpad must be assigned to a portal or one of its sites. When the launchpad is assigned to the Application Home, an empty value is retrieved.

Syntax

```
SPlus.Context.Current.siteUrl;
```

Parameters

None

Context.Current.webTitle

This property is used to get the title of the site to which the launchpad is assigned.
When the launchpad is assigned to the Application Home, an empty value is retrieved.

Syntax

```
SPlus.Context.Current.webTitle;
```

Parameters

None

Context.Current.webUrl

This property is used to get the URL of the site to which the launchpad is assigned.
When the launchpad is assigned to the Application Home, an empty value is retrieved.

Syntax

```
SPlus.Context.Current.webUrl;
```

Parameters

None

Web.getWebsAndLists

This method is used to get the sub-webs and lists of a SharePoint site.

Syntax

```
SPlus.Web.getWebsAndLists (url, function (webs, lists){
	for (var i=0; i < webs.length; i++) {//Loop the webs Array
		var web = webs [i];//Getting a JSON object with title and url properties
		var webTitle = web['title'];//Getting one of the properties of web
		//TODO: Implement what to do with each web element
	}
	for (var j=0; j < lists.length; j++) {//Loop the lists Array
		var list = lists [j];//Getting a JSON object with several properties
		var listTemplate = list['template'];//Getting one of the properties of list
		//TODO: Implement what to do with each list element
	}
}, function (errorResponse){
	var error = errorResponse['error#displayValue']; //Getting the error
	//TODO: Implement how to handle errors
}, function (cancelResponse){
	//TODO: Implement how to handle a user’s cancel
});
```

Parameters

Parameter	Type	Description
url	String	URL to the SharePoint site.
function (webs, lists)	Callback function	Success callback function to be implemented, which receives two Arrays (webs, lists) containing JSON Objects.

Parameters	Type	Description
webs	Array	Sub-webs included in the SharePoint site.

Web Object structure

Property	Type	Description
title	String	Title of the sub-web.
url	String	URL to the sub-web

Parameters	Type	Description
lists	Array	Lists included in the SharePoint site.

List Object structure

Property	Type	Description
title	String	Title of the sub-web.
url	String	URL to the sub-web.
template	Number	numeric value from SharePoint, see SPListTemplateType.
icon	String	Local image path for the icon.

Parameter	Type	Description
function (errorResponse)	Callback function	Error callback function to be implemented, which receives a JavaScript object with the error description.
function (cancelResponse)	Callback function	Cancel callback function to be implemented. Receives a JavaScript object with relevant information.

List.getItems

This method is used to get the items in a SharePoint List from a specified URL and list view. Only the field values included in the array passed to the query will be retrieved.

Syntax

```
SPlus.List.getItems (url, viewName, fields, function (items){
      for (var i=0; i < items.length; i++) {//Loop the items Array
            var item = items [i];//Getting a JSON object with the requested fields
            var title = item['ows_Title#displayValue']; //Getting one of the fields previously
                                                          specified for each item
            //TODO: Implement what to do with each item
      }
}, function (errorResponse){
      var error = errorResponse['error#displayValue']; //Getting the error
      //TODO: Implement how to handle errors
}, function (cancelResponse){
      //TODO: Implement how to handle a user's cancel
});
```

Parameters

Parameter	Type	Description
url	String	URL to the SharePoint list.
viewName	String	Name of the SharePoint list view.
fields	Array	Field values to be retrieved from each item. **fields Array structure example** `['ows_Title','ows_Created']`
function (items)	Callback function	Success callback function to be implemented, which receives an Array with the items (JavaScript Objects that represent an item).

Parameters	Type	Description
items	Array	Items from the SharePoint List. Each property corresponds to one of the requested fields; all field values are returned with String type.

Item Object structure example

Property	Type	Description
ows_Title	String	The item's title.
ows_Created	String	The item's creation date.

Parameter	Type	Description
function (errorResponse)	Callback function	Error callback function to be implemented, which receives a JavaScript object with the error description.
function (cancelResponse)	Callback function	Cancel callback function to be implemented. Receives a JavaScript object with relevant information.

Sample

For a sample with this API method, refer to the Integrating the SharePlusBridge to your procedure.

List.getItemsWithOptions

API Reference Table

This method is used to get the items in a SharePoint List, including a number of options to be used in the query to the SharePoint list. The options are passed through a custom options object detailed in the Parameters section below.

Syntax

```
SPlus.List.getItemsWithOptions (url, options, function (items){
      for (var i=0; i < items.length; i++) {//Loop the items Array
            var item = items [i];//Getting a JSON object with the requested fields
            var title = item['ows_Title#displayValue']; //Getting one of the fields for an item
            //TODO: Implement what to do with each item
      }
}, function (errorResponse){
      var error = errorResponse['error#displayValue']; //Getting the error
      //TODO: Implement how to handle errors
}, function (cancelResponse){
      //TODO: Implement how to handle a user’s cancel
});
```

Parameters

Parameter	Type	Description
url	String	URL to the SharePoint list.
options	JavaScript Object	The custom Object options includes information to be used in the query to the SharePoint list. The Query schema of CAML is used to define queries against list data.

Property	Type	Description
viewName	String	Name of the SharePoint List view.
fields	Array	Array of field values to be retrieved for each item
where	String	Where condition used in the query.
orderBy	String	OrderBy condition used in the query.
queryOptions	XML node	XML node with several properties used in the query.

Parameter	Type	Description
function (items)	Callback function	Success callback function to be implemented, which receives an Array with the items (JavaScript Objects that represent an item).

Parameters	Type	Description
items	Array	Items from the SharePoint List. Each property corresponds to each field specified in the options Object; all field values are returned with String type.

Item Object structure example

Property	Type	Description
ows_Title	String	The item’s title.
ows_Created	String	The item’s creation date.

Parameter	Type	Description
function (errorResponse)	Callback function	Error callback function to be implemented, which receives a JavaScript object with the error description.
function (cancelResponse)	Callback function	Cancel callback function to be implemented, which receives a JavaScript object with relevant information.

For a sample with this API method, refer to the **Getting items from a list, using a query** bridge scenario.

List.Item.download

This method is used to download a document from SharePoint, returning its file path for local use. Offline support is automatically enabled if the list is synchronized.

Syntax

```
SPlus.List.Item.download (url, function (filePath){
	//TODO: Implement what to do with filePath
}, function (errorResponse){
	var error = errorResponse['error#displayValue']; //Getting the error
	//TODO: Implement how to handle errors
}, function (cancelResponse){
	//TODO: Implement how to handle a user’s cancel
});
```

Parameters

Parameter	Type	Description
url	String	URL to the SharePoint list.
function(filePath)	Callback function	Success callback function to be implemented, which receives a String with the document’s file path. Parameters: filePath — Type: String — Description: File path to the downloaded document.
function (errorResponse)	Callback function	Error callback function to be implemented, which receives a JavaScript object with the error description.
function (cancelResponse)	Callback function	Cancel callback function to be implemented, which receives a JavaScript object with relevant information.

Nested table in the function(filePath) row:

Parameters	Type	Description
filePath	String	File path to the downloaded document.

User.getUser

This method is used to get the SharePoint user information from a web’s URL. The site containing that web needs to be configured in SharePlus and already been accessed within the application at least once.

Syntax

```
SPlus.User.getUser (url, function (userProfile){
      var username = userProfile['loginName'];//Getting one of the user properties
      //TODO: Get more properties
      //TODO: Implement what to do with the user profile information
}, function (errorResponse){
      var error = errorResponse['error#displayValue']; //Getting the error
      //TODO: Implement how to handle errors
});
```

Parameters

Parameter	Type	Description
url	String	URL to the SharePoint web.
function (userProfile)	Callback function	Success callback function to be implemented, which receives a JavaScript Object with the current SharePoint user properties. Retrieved properties depend entirely on the current user and may be a subset of the following:

Parameters	Type	Description
ID	String	Member ID for the user.
Sid	String	Unique security ID for the network account of the user.
Name	String	Display name of the user.
loginName	String	User name of the user.
Email	String	E-mail address of the user.
Notes	String	Notes for the user.
isSiteAdmin	Boolean	Specifies whether the user is a site collection administrator.
isDomainGroup	Boolean	Indicates whether the user is a domain group.

Parameter	Type	Description
function (errorResponse)	Callback function	Error callback function to be implemented, which receives a JavaScript object with the error description.

For further details about the user properties, see the SPUser Properties MSDN topic.

User.getProfileByName

Method used to search the user profiles from a web's URL, returning the available information for a specific user profile.

Syntax

```
SPlus.User.getProfileByName (accountName, url, function (userProfileProperties){
	//Loop the Array of user profile properties
	for (var i=0; i < userProfileProperties.length; i++) {
		//Getting a JSON object with one of the properties
		var property = userProfileProperties [i];
		//Getting the name of the current property
		var name = property ['name'];
		//Getting the displayValues Array of the current property
		var displayValues = property ['displayValues'];
		//Loop the Array of displayValues of the current property
		for (var j=0; j < displayValues.length; j++) {
			//TODO: Handle the Array of displayValues
		}
		//TODO: Implement what to do with each user profile property
	}
}, function (errorResponse){
	var error = errorResponse['error#displayValue']; //Getting the error
	//TODO: Implement how to handle errors
});
```

Parameters

Parameter	Type	Description
accountName	String	User account name to search for.
url	string	URL to the SharePoint web.
function (userProfileProperties)	Callback function	Receives an Array with user profile properties (JavaScript Objects that contain three properties).

Parameters	Type	Description
userProfileProperties	Array	User profile properties SharePoint information for the user account name provided.

User profile Object structure

Property	Type	Description
name	String	Name of the user profile property.
values	Array	String Array with all values.
displayValues	Array	String Array with all display values.

Parameter	Type	Description
function (errorResponse)	Callback function	Error callback function to be implemented, which receives a JavaScript object with the error description.

For further details about user profile properties, see the Plan user profiles article.

Note: When your SharePoint authentication is configured through Active Directory, the user account name parameter sometimes needs to be passed in a claims-encoded format.

Examples:

Username	Encoded parameter
johnSmith (windows-based)	i:0#.w|mydomain\\\\johnSmith
jSmith@acompany.onmicrosoft.com (office 365)	i:0#.f|provider|jSmith@acompany.onmicrosoft.com

For further information refer to SharePoint 2013: Claims Encoding article.

WebDashboard.Settings.save

This method is used to save custom settings for the launchpad. The settings can be any JSON object and they persist even if SharePlus is shut down.

Syntax

```
SPlus.WebDashboard.Settings.save (settings, key, function (){
      //TODO: Implementation
}, function (errorResponse){
      var error = errorResponse['error#displayValue']; //Getting the error
      //TODO: Implement how to handle errors
});
```

Parameters

Parameter	Type	Description
settings	JSON Object	Custom launchpad settings to be saved.
key	String	Sets the unique key for custom launchpad settings.
function ()	Callback function	Success callback function to be implemented.
function (errorResponse)	Callback function	Error callback function to be implemented, which receives a JavaScript object with the error description.

Sample

For a sample with this API method, refer to the **Persisting Application Launchpad Settings** bridge scenario.

WebDashboard.Settings.load

This method is used to load custom settings that were previously saved.

Syntax

```
SPlus.WebDashboard.Settings.load (key, function (settings){
      //TODO: Implement what to do with the settings
}, function (errorResponse){
      var error = errorResponse['error#displayValue']; //Getting the error
      //TODO: Implement how to handle errors
});
```

Parameters

Parameter	Type	Description
key	String	Unique key for the custom launchpad settings to be loaded.
function (settings)	Callback function	Success callback function to be implemented, which receives a JSON Object with the configuration settings loaded for the Application Launchpad.
function (errorResponse)	Callback function	Error callback function to be implemented. Receives a JavaScript object with the error description.

Sample

For a sample with this API method, refer to the Persisting Application Launchpad Settings bridge scenario.

WebDashboard.setSource

This method is used to set a new source for a launchpad in the Application Home. The change is persistent even if SharePlus is shut down and will replace all previously configured launchpads in the Application Home.

Syntax

```
SPlus.WebDashboard.setSource (source, sourceType, function (){
      //TODO: Implementation.
}, function (errorResponse){
      var error = errorResponse['error#displayValue']; //Getting the error
      //TODO: Implement how to handle errors
});
```

Parameters

Parameter	Type	Description
source	String	The source of the launchpad.
sourceType	Number	The source type of the launchpad to be loaded.
function ()	Callback function	Success callback function to be implemented.
function (errorResponse)	Callback function	Error callback function to be implemented, which receives a JavaScript object with the error description.

Sample

For a sample with this API method, refer to the **Changing the source of a** bridge scenario.

Note: The *WebDashboard.setSource* API function only works for launchpads included in the Application Home.

WebDashboard.reload

This method is used to reload the launchpad. If the source has been changed, the new launchpad is downloaded.

Syntax

```
SPlus.WebDashboard.reload();
```

Parameters

None

WebDashboard.navigate

This method is used to jump to the specified launchpad. The change is temporary and will remain until SharePlus is shut down or you jump to another launchpad.

Syntax

```
SPlus.WebDashboard.navigate (source, sourceType, function (){
       //TODO: Implementation.
}, function (errorResponse){
       var error = errorResponse['error#displayValue']; //Getting the error
       //TODO: Implement how to handle errors
});
```

Parameters

Parameter	Type	Description
source	String	The source of the launchpad.
sourceType	Number	The source type of the launchpad to be loaded.
function ()	Callback function	Success callback function to be implemented.
function (errorResponse)	Callback function	Error callback function to be implemented, which receives a JavaScript object with the error description.

Sample

For a sample with this API method, refer to the Changing the source of a bridge scenario.

Utility.getLocalizedString

This method is used to localize a text string using the SharePlus resource files.

Syntax

```
SPlus.Utility.getlocalizedString (text, function (localizedText){
      //TODO: Implement what to do with localizedText
});
```

Parameters

Parameter	Type	Description
text	String	Text to be localized.
function (localizedText)	Callback function	Success callback function to be implemented, which receives a String with the localized text string.

Parameters	Type	Description
localizedText	String	Text string localized using SharePlus resource files.

Utility.showMessage

This method is used to display an alert, which includes a title and descriptive text.

Syntax

```
SPlus.Utility.showMessage (title, message);
```

Parameters

Parameter	Type	Description
title	String	Title of the alert.
message	String	Message to be displayed in the alert.

Sample

For a sample with this API method, refer to the **Integrating the SharePlusBridge to your** procedure.

Utility.URL.open

This method is used to open an URL as if it was accessed through a link.

Syntax

```
SPlus.Utility.URL.open (url);
```

Parameters

Parameter	Type	Description
url	String	URL to be opened.

Utility.URL.download

This method is used to download a resource from an URL, returning its file path for local use. The resource is stored locally, making it available when working offline.

Syntax

```
SPlus.Utility.URL.download (resourceUrl, useCache, function (filePath){
      //TODO: Implement what to do with filePath
}, function (errorResponse){
      var error = errorResponse['error#displayValue']; //Getting the error
      //TODO: Implement how to handle errors
}, function (cancelResponse){
      //TODO: Implement how to handle a user’s cancel
});
```

Parameters

Parameter	Type	Description
url	String	URL where the resource is located.
useCache	Boolean	Determines whether to search locally for the resource before downloading it or not. Using this parameter is very useful when you work offline. Option / Result: YES: Searches the local storage for the resource, only downloading if needed. NO: The resource is downloaded.
function (filePath)	Callback function	Success *callback function* to be implemented, which receives a String with the file path of the resource. Parameters / Type / Description: filePath / String / File path to the downloaded resource.
function (errorResponse)	Callback function	Error *callback function* to be implemented, which receives a JavaScript object with the error description.
function (cancelResponse)	Callback function	Cancel *callback function* to be implemented, which receives a JavaScript object with relevant information.

Utility.URL.isAvailable

This method is used to determine if an URL is available or not. If there is no available connection, the URL will never be available as well.

Syntax

```
SPlus.Utility.URL.isAvailable (url, function (available){
      //TODO: Implement what to do when the URL is available or not
}, function (errorResponse){
      var error = errorResponse['error#displayValue']; //Getting the error
      //TODO: Implement how to handle errors
}, function (cancelResponse){
      //TODO: Implement how to handle a user’s cancel
});
```

Parameters

Parameter	Type	Description
url	String	URL to be checked for availability.
function (available)	Callback function	Success callback function to be implemented, which receives a Boolean value (True or False) depending on the URL availability. Parameters | Type | Description available | Boolean | Notifies whether the URL is available or not.
function (errorResponse)	Callback function	Error callback function to be implemented, which receives a JavaScript object with the error description.
function (cancelResponse)	Callback function	Cancel callback function to be implemented, which receives a JavaScript object with relevant information.

Sample

For a sample with this API method, refer to the **Integrating the SharePlusBridge to your** procedure.

Configuration.setRemoteFileSource

API Reference Table

This method is used to set a new source for the Configuration File, but does not trigger a Configuration File reload.

Syntax

```
SPlus.Configuration.setRemoteFileSource (configSource, function (){
      //TODO: Implementation.
}, function (errorResponse){
      var error = errorResponse['error#displayValue']; //Getting the error
      //TODO: Implement how to handle errors
});
```

Parameters

Parameter	Type	Description
configSource	String	URL to the remote Configuration File location
function ()	Callback function	Success callback function to be implemented.
function (errorResponse)	Callback function	Error callback function to be implemented, which receives a JavaScript object with the error description.

Note: To enforce security, this API function can be disabled by configuration.
In the Configuration File, the AllowConfigurationUpdate setting (URLSchemes feature) must be set to false. For details about this setting refer to *Configuring Security* > *Configuration Injection* > *SharePlus link* in **SharePlus Administrator Guide.**

Configuration.reload

This method is used to force a Configuration File update.

Syntax

```
SPlus.Configuration.reload ();
```

Parameters

None

Configuration.getFeature

This method is used to get the configuration settings of a feature in the Configuration File (*config.plist*).

Syntax

```
SPlus.Configuration.getFeature (featureKey, function (featureConfig){
       var enabled = featureConfig['Enabled'];
       var settings = featureConfig['Settings'];
       var _title = settings ['Title']
       //TODO: Get all settings
       //TODO: Implement what to do with the configuration settings
}, function (errorResponse){
       var error = errorResponse['error#displayValue']; //Getting the error
       //TODO: Implement how to handle errors
});
```

Parameters

Parameter	Type	Description
featureKey	String	The feature's Key
function (featureConfig)	Callback function	Success callback function to be implemented, which receives a JavaScript Object with the complete configuration settings for the specified feature. **feature structure example** <table><tr><th>Property</th><th>Type</th><th>Description</th></tr><tr><td>enabled</td><td>Boolean</td><td>Depends on whether the feature is enabled or not</td></tr><tr><td>settings</td><td>JSON Object</td><td>Object containing all the settings as properties</td></tr></table>
function (errorResponse)	Callback function	Error callback function to be implemented. Receives a JavaScript Object with the error description.

Configuration.getVariables

This method is used to get the variables specified in the Configuration File (*config.plist*).

Syntax

```
SPlus.Configuration.getVariables (function (variables){
      var myCustomVariable = variables['customVariable']; //Getting one of variables
      //TODO: Implement what to do with each variable
});
```

Parameters

Parameter	Type	Description
function (variables)	Callback function	Success callback function to be implemented, which receives a JavaScript Object with the variables specified in the Configuration File (config.plist).

Connection.isOnline

This method is used to determine the connection mode and is closely related to the **go online/offline** button.

Syntax

```
SPlus.Connection.isOnline (function (online){
      //TODO: Implement what to do when SharePlus is online or not
});
```

Parameters

Parameter	Type	Description
function (online)	Callback function	Success callback function to be implemented, which receives a Boolean value (True or False) depending on the connection mode. Parameters | Type | Description online | Boolean | Notifies whether SharePlus is online or not.

Connection.isConnected

This method is used to determine if there is an internet connection available or not, depends entirely on the device.

Syntax

```
SPlus.Connection.isConnected (function (connected){
      //TODO: Implement what to do when there is an available connection or not
});
```

Parameters

Parameter	Type	Description
function (connected)	Callback function	Success callback function to be implemented. Receives a Boolean value (True or False) depending on the connection availability.

Parameters	Type	Description
connected	Boolean	Notifies whether SharePlus has a working connection or not.

Sample

For a sample with this API method, refer to the **Integrating the SharePlusBridge to your** procedure.

Appendix 3: Document Changelog

Version	Section	Description
1.1	Introducing the Web SDK	Screenshot update in Custom Home Screen.
	Getting Started	Screenshots update in Creating your First Launchpad.
	API Reference	Added a Note about the Configuration.setRemoteFileSource API method.
	Extending the API	Section removed (not suitable for this guide).
1.2	-	Guide renamed to Launchpads Developer Guide.
	Launchpads Deployment	The Home Configuration section was relocated and renamed to Launchpads Deployment.
		The Site Home configuration scenario was updated and now allows multiple values in the Home column of the Site Configuration List.
		The Default Homes configuration scenario has minor fixes and was renamed to Default Home – Portal and sites.
	Getting Started, Launchpads Deployment	Sections were updated to include launchpads packaging and deployment using ZIP files. Webarchive format is deprecated.
	API Reference	Added new context properties (.Context).