SharePlus™

POWERED BY INFRAGISTICS

Developer's Guide

SharePlus™ 4.9 – Developer’s Guide 5.0.3

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.

Overview

SharePlus SDK allows you enhance the user experience by integrating web technologies like HTML, CSS, and JavaScript into the native User Interface. With that goal in mind, **SharePlus SDK has three major extension points**:

- **The start screen** – When you open the Application Home module
- **Sites** – When you navigate to a SharePoint site within SharePlus
- **List items** – When you add, edit, or visualize an item of a SharePoint list.

All the three extension points allow you to present a custom view to the user, which is developed using the technology described above.

SharePlus API

You can develop HTML-based packages by 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. HTML-based packages can communicate with the app through two means:

- Invoking native SharePlus actions by using **S+ links (SharePlus custom URL schema).**
- Reading/Writing SharePoint data using **SharePlus' JavaScript API.**

Main Features

Custom Screens

Building HTML ZIP packages is especially powerful, as you can create rich HTML pages including markup, images, CSS, and JavaScript frameworks like JQuery, to present the user with virtually any content you want to display. You can display customized screens in the three major extension points previously described.

Read/Write JavaScript APIs

Use SharePlus JavaScript APIs to build complex scenarios with full access to SharePoint content. You could achieve many scenarios, including:

- Adding, editing, deleting, or retrieving an item or document from SharePoint.
- Getting all the items from a SharePoint List.
- Getting user information of a SharePoint web.
- Managing the source and custom settings from SharePlus Mobile Workspaces.
- Downloading a resource from an URL and making it available when working offline.
- Checking whether SharePlus is online or not.

Remote Deployment

Your ZIP packages can be deployed using a centralized location, allowing a quick and easy distribution to all company devices. The content can be loaded either from an external URL or a local path inside the application.

Short Learning Curve

SharePlus SDK is not difficult to learn and you don't need further knowledge besides HTML5 and JavaScript. Also, you can use SharePlus Mobile Workspaces in combination with S+ links and in that scenario you don't even need JavaScript.

Introduction

SharePlus SDK allows you to present a custom view to the user in one or more of the application extension points. Customize the Application Home or any site (Mobile Workspace), or personalize how the items of a List are added, modified, or visualized (Custom Form). In all cases a specific ZIP file package is loaded as the source, and the file can be downloaded from a specified static URL or stored inside the SharePlus application. Your ZIP package interacts with SharePlus by invoking the application using S+ links or reading/writing SharePoint data using the JavaScript API.

Custom HTML page

Using an HTML page combined with **S+ links** (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 **S+ links** to navigate.

Remote Deployment

Your file 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. Your ZIP package can be available offline, and will later synchronize with the URL every time you start SharePlus. Caching is used to avoid downloading the same resource that is referenced by the URL every time. However, the source file is downloaded again when there are changes.

HTTP Caching for Mobile Workspaces

For HTTP, caching is not expected to take place. This might cause a performance impact, given that caching prevents the Mobile Workspace from being downloaded again if there are no changes. **It is therefore recommended to place the Mobile Workspace in a list/library that supports caching.**

Creating your first Customization with a ZIP Package

In this procedure you will create a new mobile workspace, which is displayed in the Application Home (start screen). Ultimately, you will have a **custom HTML page** that allows you to navigate to different parts of the SharePlus application, **using S+ links**.

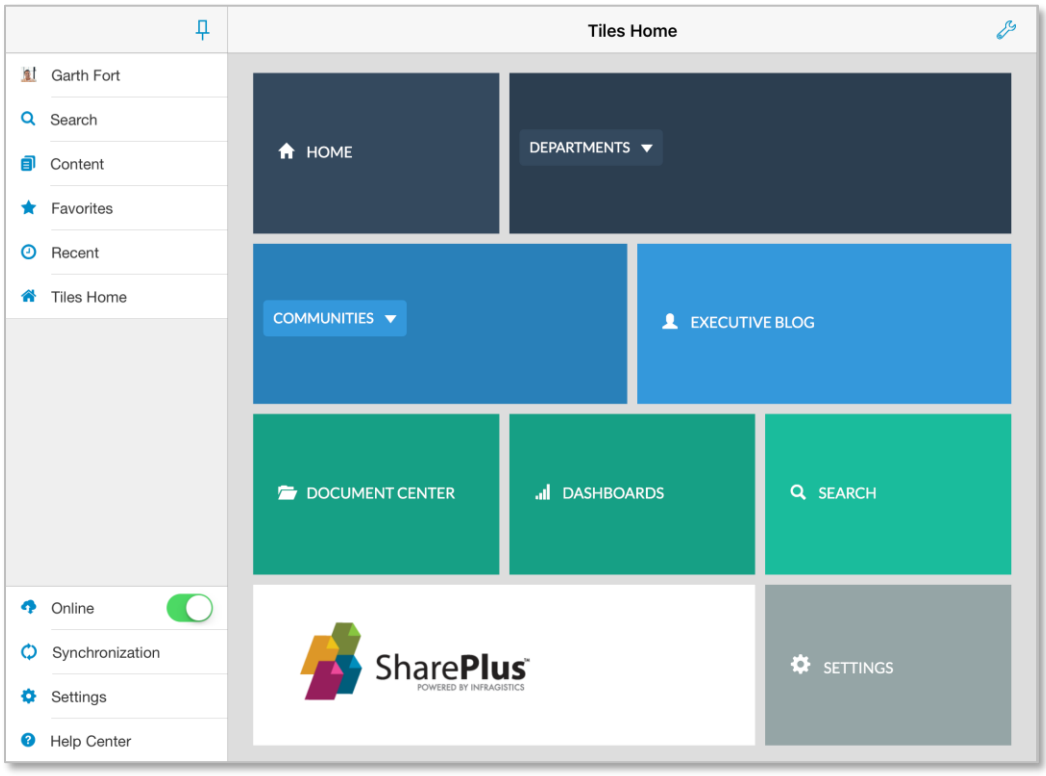
Overview

The process has the following steps:

1. Creating a custom mobile workspace.
2. Preparing your deployment package for SharePlus.
3. Displaying your custom mobile workspace in SharePlus.

Steps

1. **Create a custom mobile workspace**
    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 S+ links.
    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 can add behavior to your page by navigating or invoking actions within SharePlus. Below, you will find a few buttons with some possible customizations:

- Opening SharePlus Settings.

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

- Navigating to a static URL.

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

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 to your initial HTML page.

- Navigating to a Document Library.

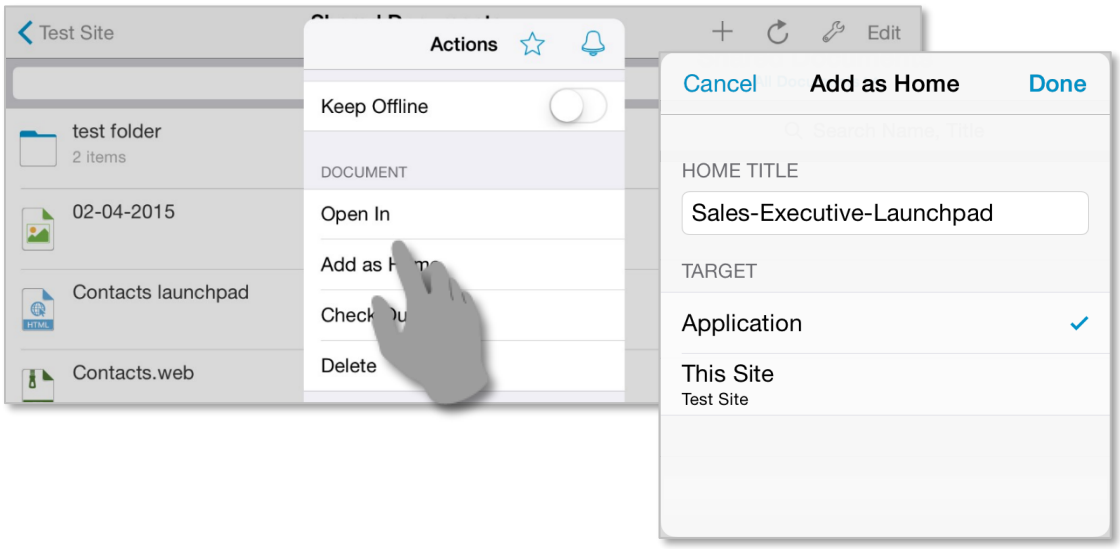
```
<a class="button" href="splus://MySharePointPortal/Site/Documents">DOCUMENT CENTER</a>
```

- Opening a Document.

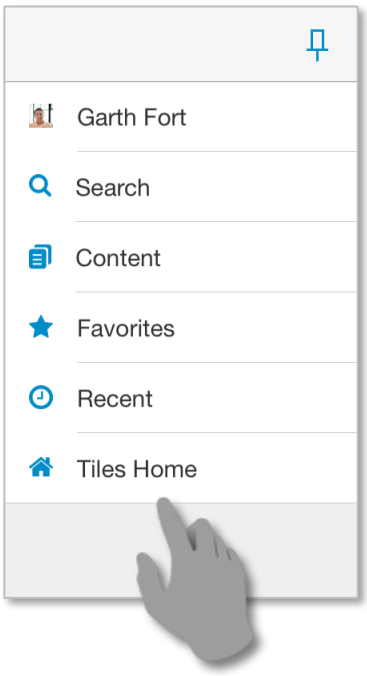
```
<a class="button" href="splus:// MySPPortal/Site/Documents/sampleDoc.docx?splus-action=viewdocument">DASHBOARDS</a>
```

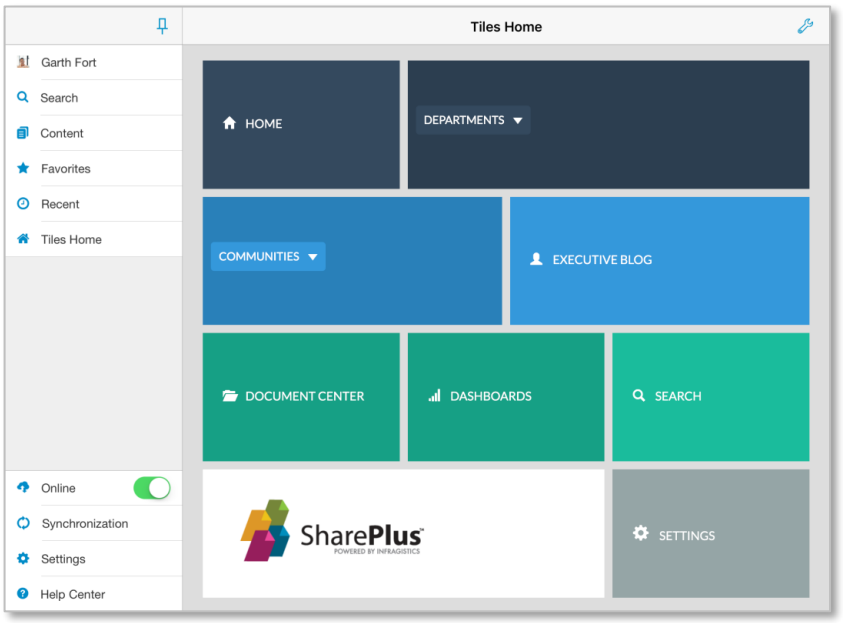
For further details about S+ links refer to the *Using SharePlus Links* section.

2. **Prepare your deployment package for SharePlus**
   a. **Create a ZIP file**
   You can use ZIP files to package and deploy your custom mobile workspaces.
      i. Add files to the compressed file.
      You need all the required files for your SharePlus Mobile Workspace to work as expected, including HTML, JavaScript, CSS, images, and other assets.
      ii. Get the ZIP file ready.
      Name the file using the **.web.zip** extension. E.g.: Demo.web.zip. During step 3, you will learn how to configure your mobile workspace using the *Add as Home* action through the UI.
      For further details, refer to the **Deployment** section.
   b. **Upload your file to SharePoint**
   Add your file to a document library, making it accessible from the SharePlus application.
3. **Display your custom mobile workspace in SharePlus**
   a. **Configure the Application Home**
   You can manually configure the Application Home through the User Interface.
      i. Access the actions menu for your item (ZIP file).
      Tap & hold over the item to open the *Actions* menu and select the *Add as Home* action.
      ii. Choose where to display your content.
      You can choose between two targets: Application (Application Home) or This Site (the site where the item is located).



   b. **Access your mobile workspace**
   Tap the Application Home module in the SideBar to display your SharePlus Mobile Workspace.


Garth Fort
Search
Content
Favorites
Recent
Tiles Home


Tiles Home
Garth Fort
Search
Content
Favorites
Recent
Tiles Home
Online
Synchronization
Settings
Help Center
HOME
DEPARTMENTS
COMMUNITIES
EXECUTIVE BLOG
DOCUMENT CENTER
DASHBOARDS
SEARCH
SharePlus
SETTINGS

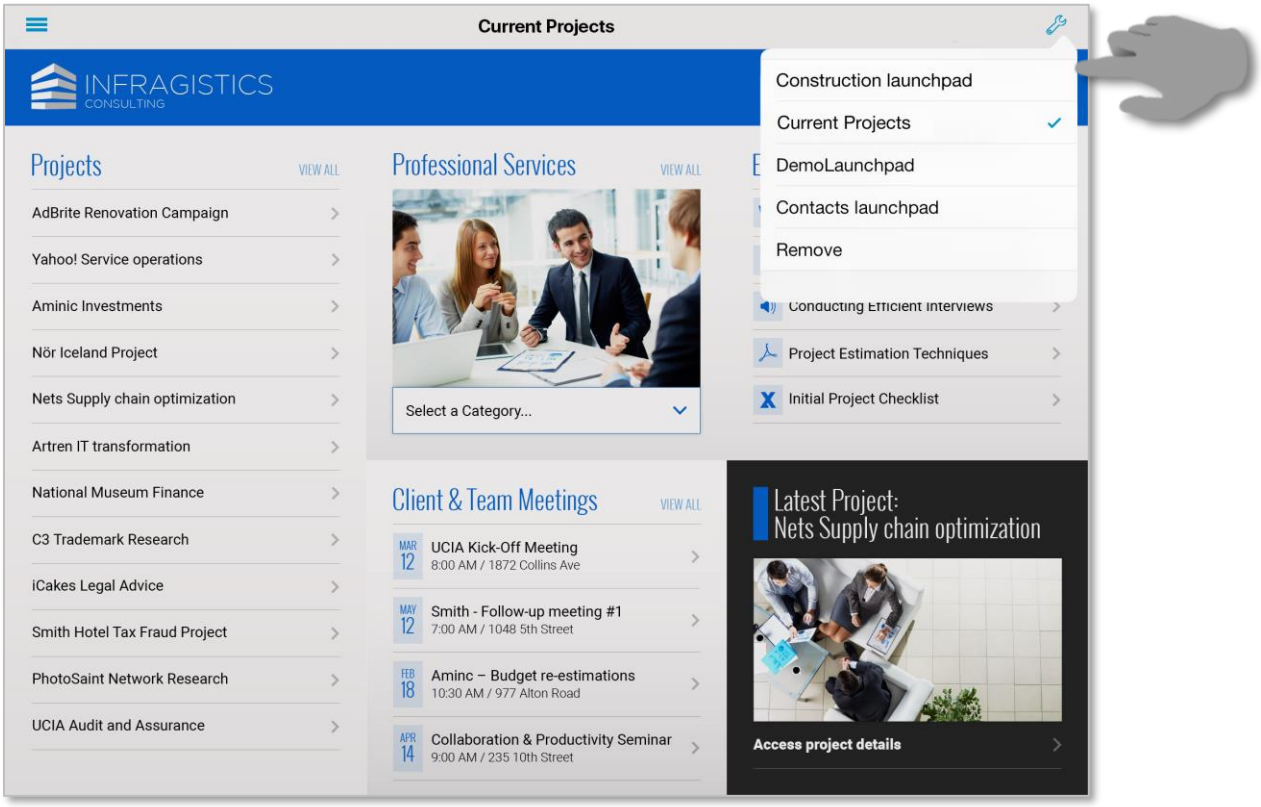
SharePlus Extension Points

SharePlus SDK has three major extension points where you can display custom content:

- **Sites** and the **start screen** – When you navigate to a SharePoint site or open the Application Home module, a ZIP package (Mobile Workspace), PDF, or ReportPlus dashboard can be displayed.
- **List items** – When you add, edit, or visualize an item of a SharePoint list, a ZIP package can be displayed to show a custom form to the user.

Displaying content

When accessing a custom form for list items, a Site, or the Application Home within SharePlus, you can configure different types of content to be displayed including ZIP packages, ReportPlus dashboards, and PDF files. When you have more than one content assigned, you can switch between different content by using a selector.



Mobile Workspaces

SharePlus allows you to display custom Mobile Workspaces by customizing Sites or the Application Home.

Application Home – This module can be accessed from the SideBar, and is sometimes referenced as the "Start Screen" because it's loaded by default when opening the app from scratch.

Sites – All sites can display custom content in SharePlus by presenting the user with a customized view for a given site. You can pre-configure and share this content across sites, displaying different content depending on the SharePoint site's context.

Custom Forms

As an alternative to SharePlus' native visualization, you can use a custom form to display, add, and edit items within a SharePoint list.

Using a custom form for multiple sites

Custom Forms work hierarchically. Applying a form to a site means that the custom form will be applied to all the site's subsites and lists for the content type you specify. Similarly, applying a custom form to a list means that the custom form will be applied to all items on that list depending on the content type you specify. For more information on content types, visit the Microsoft Base Content Type Hierarchy library.

Adding, editing, or just displaying items

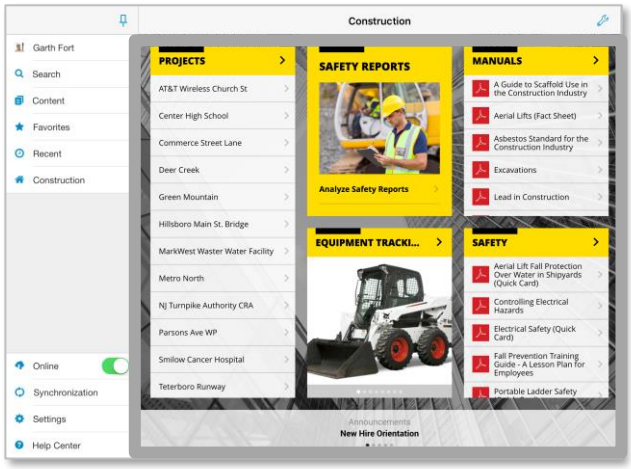
When you apply a form to a list, document library, or site, you need to specify for which user actions the form will be shown (i.e., adding, editing, and/or displaying items/documents).

About the Canvas Size

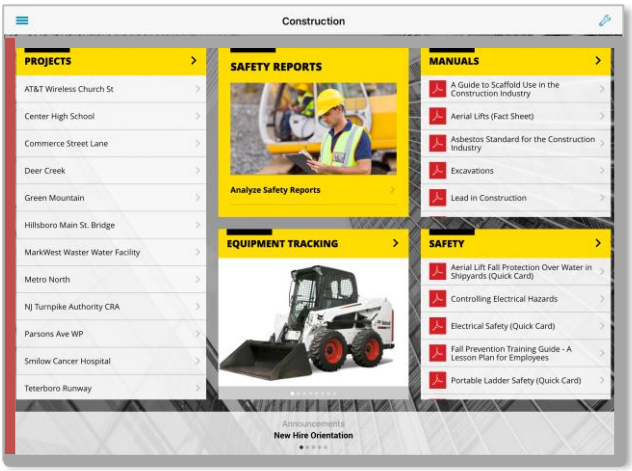
The canvas size where your ZIP package is displayed depends on:

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

Landscape Orientation

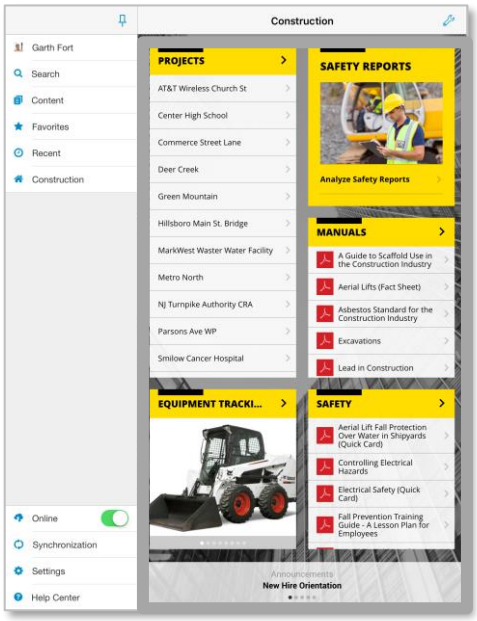


1608 x 1408 px

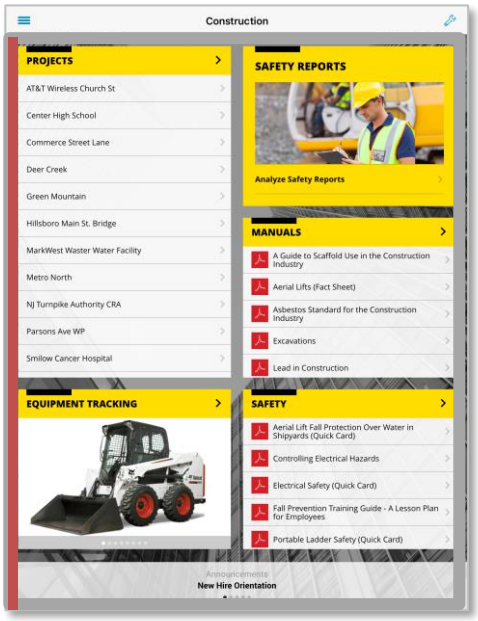


2048 x 1408 px

Portrait Orientation



1095 x 1919 px



1536 x 1919 px

Canvas Limitations

In SharePlus, you can switch between a visible or a 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 SharePlus cannot process gestures within that reserved section.

You can display ZIP packages (SharePlus Mobile Workspaces or Custom Forms) in different areas of the SharePlus application. The deployment process consists on preparing the deployment package and later configuring the package to be used in SharePlus.

Preparing your ZIP Deployment Package

The compressed file should contain all the required files for your custom package 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:

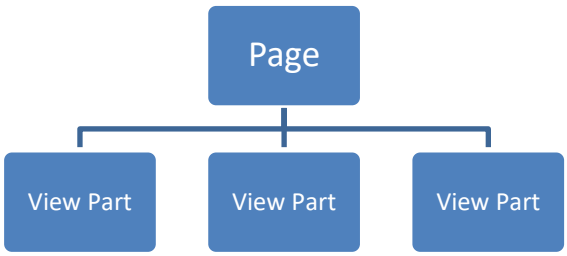
1. SharePlus searches the main folder for a file named *index.html*, *index.htm* , *default.html* or *default.htm.*
2. If no file was found, SharePlus searches the main folder for any file with *.html* or *.htm* extensions.
3. If both searches were unsuccessful, SharePlus searches the files in the first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

Note: The user can manually change the start screen (Application Home) by configuring ZIP files in-app, using the *Add as Home* action. In order to do this, just name the file using the **.web.zip** extension. E.g.: Demo.web.zip

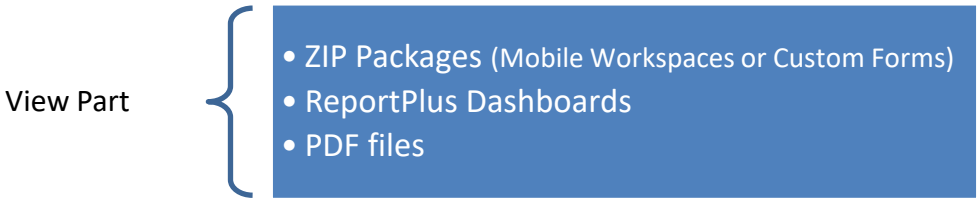
Configuring your ZIP Package

About 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 content by including one or more View Parts.



Each View Part can display a ZIP package, ReportPlus Dashboard, or a PDF file. When having more than one View Part, the user can swap between them through the User Interface.



View Parts are in charge of displaying content inside a Page and there are two types of View Parts OOTB, one for ZIP packages and another one for 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 content for different areas of the SharePlus application.

You can configure your ZIP packages in the following scenarios:

- In-app configuration – Configure a Site Home or the Application Home through the UI.
- Application Home – Configure the Application Home module in the Sidebar.
- Site Home – Configure a customized view for a given site.
- Default Home – Configure a default view to be used across all the sub-sites of a portal.
- Custom Form – Configure a customized form for a SharePoint list.

The following table lists the available scenarios; additional scenario information is available after the following table.

Scenario	Details	Configuration Method
In-App Configuration	Choose where to display your content through the *Add as Home* action from the Actions menu.	Actions menu – Add as Home
Application Home	Create a Page holding your content 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 your ZIP file.	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
Custom Form	Create a Page holding your content and reference the PageName in the CustomForms feature.	Configuration File – Pages, CustomForms

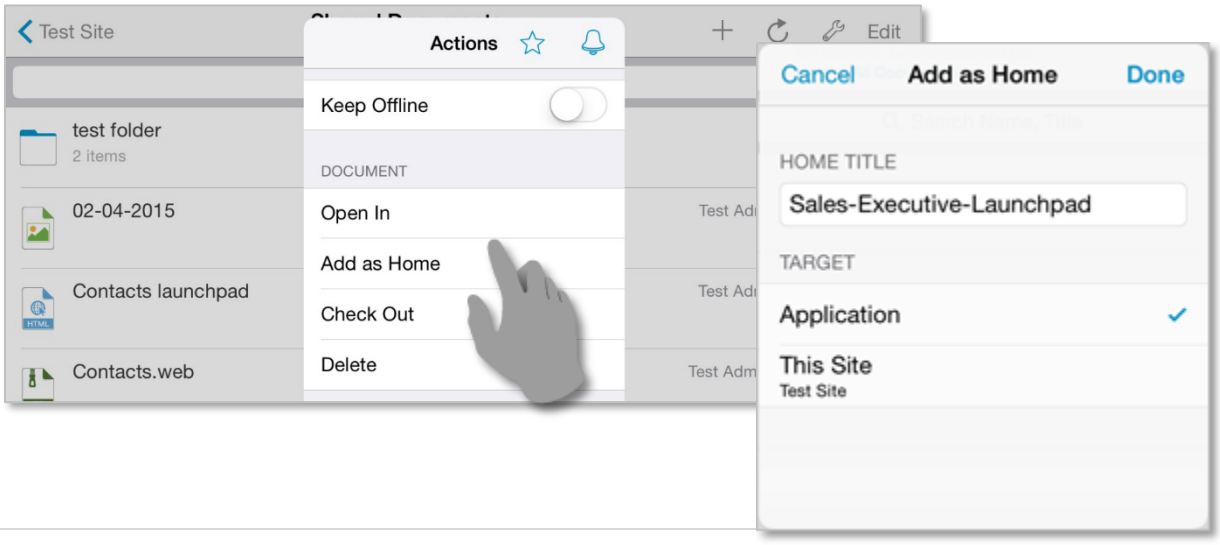
In-App Configuration

You can manually configure content for a Site Home or the Application Home module through the User Interface.

Steps

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

2. **Choose where to display your home content**
   You can choose between two targets: Application (Application Home module) or This Site (the site where the *.web.zip* file is located).



Application Home

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

Steps Overview

1. Creating a Page that includes a mobile workspace.
2. Configuring the Application Home.

Steps

1. **Create a Page that includes a SharePlus Mobile Workspace**
   Define a new page in the Configuration File or use an existing one. You need a View Part for mobile workspaces 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 a SharePlus Mobile Workspace.
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>customMobileWorkspace</string>
		<key>Settings</key>
		<dict>
			<key>Source</key>
			<string>Documents/Demo.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 ZIP Packages 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 package file. The source can be an URL or a local path inside the application.
- **SourceType** – Sets the source type of the SharePlus Mobile Workspace. 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 your file from Local Files, the path must start with "Documents/". This is a useful approach for debugging and testing new mobile workspaces.
- When downloading your file from a static URL, the URL location must be public or otherwise located in a server that shares existing credentials (Accounts) in SharePlus.

2. Configure the Application Home

a. Open the Configuration File.
Search for the *HomeManagement* feature in the Features section.

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 the 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 a SharePlus Mobile Workspace as custom content for a site. In addition, you can configure one or more mobile workspaces for the same site. Your Site Home can contain a ZIP package (SharePlus Mobile Workspace), a PDF document, a ReportPlus dashboard or any content that can be rendered in a web browser.

Column	Type	Description
Title	Single line of text	To configure the site itself it should be set to "ThisWeb" or "ThisSite".
Is Web	Yes/No	To apply the settings to a site (web) it should be set to "Yes".
Home	Single line of text	A list of values separated by comma. Possible values are: • A URL - to reference a ZIP package 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. Next, you can find two Home column examples using the "default" value:
1. "default"
2. "customPageName,default"

For further details about the MobileNavigation list refer to Site Configuration aka “MobileNavigation” section in *SharePlus Administrator’s Guide*.

Default Home – Portal and sites

In SharePlus, Portals can present a pre-configured mobile workspace that can be shared with its sites. In this scenario, you use the same ZIP package but different content is displayed to the user depending on the site’s context.
You can configure a Site Home with a SharePlus Mobile Workspace 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 a SharePlus Mobile Workspace
2. (*Optional*) Enabling the default list view for the Portal
3. Configuring the Portal’s Home
4. Defining the Site Home by default (*DefaultSitePageName*)
5. Adjusting inheritance settings (*InheritParentPage*)

Steps

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

2. **(*Optional*) Enable the default list view for the Portal**
   When pre-configuring home content as the Portal Home, the default content is replaced.
   A View Part with SharePlus default list view needs to be added, so you can switch between content (default list view and the new home content) 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 SharePlus Mobile Workspace 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 the Portal's Home**
   a. Open the Configuration File.
   Search your Portal configuration in the Sites section.

   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).**
   In, *HomeManagement*, 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.

Custom Form

Because the Custom Forms feature works hierarchically, applying a form to a site means that the custom form will be applied to all the site's subsites and lists for the content type you specify. Similarly, applying a custom form to a list means that the custom form will be applied to all items on that list depending on the content type you specify.

This also means, for example, that if you use 0x01 (item) as a content type, the custom form will be applied to items on the list or site, but also to all content types included below the 0x01 hierarchy (0x0101 – Documents, 0x0102 – Events, 0x0103 – Issue, etc.). For more information on content types, visit the Microsoft Base Content Type Hierarchy library. You can pre-configure a custom form to be used within SharePlus in the Configuration File using an existing Page.

Steps Overview

1. Creating a Page that includes a Custom Form package.
2. Configuring the Custom Form. During this step, you will specify where the custom form will be applied.

Steps

1. **Create a Page that includes a Custom Form package**
   Define a new page in the Configuration File or use an existing one. You need a View Part containing your custom form package.
   a. Open the Configuration File and navigate to the Pages section to create a new Page.
   b. Define a Page.

```
<key>Pages</key>
<array>
      <dict>
            <key>PageName</key>
            <string>MyCustomForm</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 Custom Form.
- **ShowSelector** - Does nothing, the first View Part is always used.
- **ViewParts** – Array of View Part elements defined for the page. Only the fist View Part will be used.

   c. Specify a View Part containing a custom form.

Each page has an array of view Part elements that you must specify.

```
<key>ViewParts</key>
<array>
     <dict>
         <key>Settings</key>
          <dict>
              <key>Source</key>
                   <string>http://mySharePoint.com/.../sitename/libraryname/CustomForm.zip</string>
              <key>SourceType</key>
                <integer>1</integer>
            </dict>
         <key>Title</key>
         <string>Construction Project</string>
         <key>ViewControllerID</key>
         <integer>0</integer>
     </dict>
</array>
```

The four elements are:

- **ViewControllerID** – Specifies the ID defined in the app.plist for the View Controller to be used. For Custom Forms, Mobile Workspaces and PDF files, the value 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 package file. The source can be a URL or a local path inside the application.
- **SourceType** – Sets the source type of the Custom Form package. Possible values are:
    - **0** – to reference a 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 your file from Local Files, the path must start with "Documents/". This is a useful approach for debugging and testing new mobile workspaces.
- When downloading your file 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 the Custom Form**
    a. Open the Configuration File.
    Search for the *CustomForms* feature in the Features section.

    b. Reference the Page.
    You need to reference the page you created by name in the *PageName* key value.

The Configuration File should be similar to the one below:

```
<key>CustomForms</key>
<dict>
      <key>Enabled</key>
      <true/>
      <key>Settings</key>
      <dict>
         <key>CustomFormsDefinitions</key>
         <array>
            <dict>
               <key>BaseContentType</key>
               <string>0x01</string>
               <key>BaseUrl</key>
               <string>http://yourSharePoint.com/.../sitename/libraryname</string>
               <key>PageName</key>
               <string>customPageName</string>
               <key>CustomFormActions</key>
               <array>
                  <dict>
                     <key>CustomFormAction</key>
                     <string>New</string>
                  </dict>
                  <dict>
                     <key>CustomFormAction</key>
                     <string>Edit</string>
                  </dict>
                  <dict>
                     <key>CustomFormAction</key>
                     <string>Display</string>
                  </dict>
               </array>
               <key>ShowSaveButton</key>
               <false/>
            </dict>
            ...
         </array>
      </dict>
</dict>
```

The four elements are:

- **BaseContentType**: the SharePoint ContentType for the list items being modified, added or displayed. For more information on content types, visit the Microsoft Base Content Type Hierarchy library.
- **BaseURL**: the site collection or specific list to which you will apply the Custom Form.
    - Applying a custom form to a site collection means applying the form to **all** lists and their items under that site collection for the *Content Type* you specify.
    - Applying a custom form for a specific list means applying the form to **only** that list and its items for the *Content Type* you specify.

- **PageName**: the name used to reference the page when configuring a Custom Form for the application or sites.
- **CustomFormActions**: Specify at least one of the actions (New, Edit, Display) in which the Custom Form will be shown for that list and content type. I.e., when adding, editing, or displaying an item for that list.
- **(Optional) ShowSaveButton**: by default, the native SharePlus "Save" button will always appear for Custom Forms in the top right corner. However, if the *ShowSaveButton* is included in the config.plist file and is set to <false/>, the native "Save" button will disappear. The ShowSaveButton functionality is available only for CustomFormsActions "New" and "Edit".

Introduction

A SharePlus link (S+ Link) is a SharePlus feature which allows users to perform certain actions within SharePlus, invoking these actions from HTML content. A S+ Link is basically a custom URL scheme that starts with **splus://** (HTTP) or **spluss://** (secure HTTPS channel) and is followed by the resource's URL without the HTTP or HTTPS protocols. They can be used to modify the application configuration when building the URL with a set of required parameters.

S+ links are useful for a number of application-related behaviors:

- SharePoint Navigation
- Edition
- SharePlus Basic
- Configuration Change
- Search

For more information on any of these, go to Appendix 2 (SharePlus Links Reference section).

SharePlus Links Structure

The structure of SharePlus links always follows this order:

1. The link starts with the **scheme** on the left-most of the URL (**splus://** or **spluss://**).
2. *If applicable*, it continues with the **referenced resource**. If invoking application actions, you do not need to include any resources.
3. If you are working with dynamic webpages (.aspx), parameters can be included in the form of key-value pairs.
4. SharePlus actions and their parameters (if any) are locate on the right-most of the URL.

Your resulting URL will look like the following one:

<scheme>://<resource URL>?<resource parameters><SharePlus action and parameters>

Note: A referenced resource may need a parameter named "action". That is not an issue as SharePlus engine searches for the **splus-action** SharePlus link parameter. The **action** is deprecated but still works to avoid compatibility issues.

SharePlus Link Types

As mentioned, there are several ways to use a SharePlus link within the application. Below are some of the most common scenarios:

Actions Group	Description
SharePoint Navigation	Navigation to Webs, Lists, Items or Documents
Edition	Add/Edit Items or Documents
SharePlus Basic	Navigation within SharePlus components (e.g., Local Files, Favorites, Help)
Configuration Change	Change the Remote Configuration URL or SharePlus Mobile Workspace URL.
Search	Open a query through the Advanced List Search or Enterprise Search (Web).

SharePoint Navigation Actions

When invoking navigation actions to SharePoint Webs or Lists, there are three different modes in which you can access the content:

- **InWeb** – A web view is opened from SharePlus displaying the navigation action.
- **InSafari** – iOS Safari browser is invoked by SharePlus, opening with the navigation action.
- **Native** (Default mode) – The navigation action is displayed within SharePlus

Web Samples

- Navigating to a SharePoint Web, opening Safari and requesting the mode to the user.

```
splus://<portal>/site?splus-action=view&mode=InSafari

splus://<portal>/site/multimedia?splus-action=view&mode=AskUser
```

When navigating sites, the user is prompted to select the mode to be used (SharePlus, Safari, or Web).

List Samples

- Navigating to a SharePoint List, to the default view and also to a specific List View.

```
splus://<portal>/site/calendar?splus-action=view

splus://<portal>/site/calendar?splus-action=view&viewName=All%20Events
```

- Browsing a list's folder contents, both samples below can be used indistinctly.

```
splus://<portal>/site/Shared%20Documents/RSS%20Samples

splus://<portal>/site/Shared%20Documents/RSS%20Samples?splus-action=query
```

Item Sample

- Navigating to the details of an item.

```
splus://<portal>/site/Tasks/ID=4
```

Document Samples

- Navigating to the details of a SharePoint Document.

```
splus://<portal>/site/multimedia/Far%20Away.mp3?splus-action=viewitem
```

- Opening a *SharePoint Document.*

```
splus://<portal>/site/multimedia/Away.mp3?splus-action=viewdocument
```

Editing Actions

These actions may include initial field values to be loaded when opening the SharePoint Add/Edit view. When initial field values are included, there are three important considerations:

- The *field names* must match the names used in SharePoint, including the ows_prefix.
- All *field values* must be URL escaped.
- All field values must be included in a specific format, as shown below.

Field type	Format	Examples
Single/Multiple lines of text, Numbers, Currency, Hyperlink	These values need no format.	**Text:** ows_TextField=Text%20Value **Number:** ows_NumberField=15
Choice	Values must be separated by "**;#**" characters (%3B%23 when escaped).	**Single:** ows_ChoiceField=%3B%23Value1%3B%23 (Non-escaped: ows_ChoiceField=;#Value1;#) **Multi:** ows_ChoiceField=%3B%23Value1%3B%23Value2%3B%23 (Non-esc: ows_ChoiceField;#Value1;#Value2;#)
Lookup, Person or Group	ItemID;#Name	ows_LookupField=103%3B%23Test (Non-escaped: ows_LookupField =103;#Test)
Date and Time	yyyy-MM-dd'T'HH:mm:ss'Z'	ows_DateField=2012-12-27T16%3A15%3A31Z (Non-esc: ows_DateField =2012-12-27T16:15:31Z)
Yes/No	TRUE or FALSE	ows_YesNoField=TRUE

Samples

- Adding a new SharePoint Item or Document.

```
splus://<portal>/site/multimedia/Away.mp3?splus-action=additem
&contenttype=audio

splus://<portal>/site/multimedia/Home.mp3?splus-action=additem
&contenttype=audio&ows_comments=Added%20with%20URL%20schemes
```

- Editing a new SharePoint Document.

```
splus://<portal>/site/multimedia/Away.mp3?splus-action=edititem

splus://<portal>/site/multimedia/Away.mp3?splus-action=edititem
&ows_title=New%20title
```

SharePlus Application Actions

- Invoking a *SharePlus component*.

```
splus://?splus-action=home

splus://?splus-action=localfiles&folder=Logs

splus://?splus-action=favorites

splus://?splus-action=settings

splus://?splus-action=help

splus://?splus-action=feedback
```

Configuration Change Actions

These actions let you modify the application's configuration and the SharePlus Mobile Workspace source for the Application Home. The *URLSchemes* feature and its settings *AllowConfigurationUpdate* and *AllowWebDashboardUpdate* respectively must be enabled in the application's configuration (Configuration File) for these actions to work.

Samples

- Modifying the Remote Configuration section

```
splus://?splus-action=configurationURL&url=
https%3A%2F%2Fportal%2FConfigurationFiles%2FCustomConfiguration.plist
```

- Modifying the source of a SharePlus Mobile Workspace in the Application Home

```
splus://?splus-
action=webdashboard&source=http%3A%2F%2Fportal%2Fsite%2FSiteAssets%2FDemo.web.zip&title=MyCu
stomMobileWorkspace
```

Values	Description
source	The URL to the resource or the local path to an existing web resource. An empty value will disable the SharePlus Mobile Workspaces feature.
title	This is an optional value and sets a new title.

Search Actions

Using these actions, you can open a query to a list in the Advanced List Search, as you would from the UI. The Advanced Search feature filters items by the specified fields' value, using a specific operator. Only one field can be filtered at a time. E.g., ows_Title:contains(SharePlus)

Format to be used:

```
fieldname:operator(value)
```

Operators	Description
equals	The field value must be exactly the same
notequals	The field value must be different
greater	The field value must be greater
greaterorequal	The field value must be greater or equal
lower	The field value must be lower
lowerorequal	The field value must be lower or equal
isnull	The field value must not be specified (isnull())
isnotnull	The field value must not be specified (isnotnull())
beginswith	The field value must start with the value specified
contains	The field value must contain the value specified

The Search Module (Enterprise) can also be opened from a search action, and search queries can be constructed using the keyword query syntax. Advanced filtering can be achieved through property restrictions, for further details refer to the Property Restriction Keyword Queries MSDN article.

Samples

- Performing a search in the Search Module on a specific site.

```
splus://<portal>/site?splus-action=search

splus://<portal>/site?splus-action=search&query=pdf
```

- Performing a custom search on a SharePoint List.

```
splus://<portal>/site/multimedia?splus-
action=query&filter=ows_Title:contains(Text)&filtertitle=Text%20Filter&includesubfolders=true
```

Integrating SharePlus Links to your ZIP Packages

Following this procedure, you will be able to add S+ links with different functionalities (adding a direct link to a document, site or list or even launching a detailed search within a site).

Overview

The process has 2 required steps:

1. Defining the S+ link type you will use
2. Enabling the SharePlus link in the index.html file
3. (Optional) Testing the resulting link in SharePlus

Steps

1. **Define the S+ link type you will use**
   The URL structure is a different one for each SharePlus link type; therefore, choose the one that is most suitable to your needs.

2. **Enable the S+ link in the index.html file**
   The SharePlus link needs to be included in the index.html file.
   a. Locate the place where you want the S+ link and create the link URL structure you need, including resources, parameters, and SharePlus actions.
   b. Include the S+ link. Remember to use **splus** or **spluss** (as applicable) instead of HTTP/HTTPS.

3. (Optional) **Test the resulting link**
   Make sure the S+ link is working as expected by testing your zip package.

SharePlus Link Samples

Invoking actions	S+ link
… over a SharePoint resource	splus://portal/site/library/document.docx?splus-action=view&mode=InSafari
… over a dynamic web page	splus://dynamicWebServer/page.aspx?color=red&splus-action=view&mode=InSafari
… over a ReportPlus dashboard with parameters	splus://portal/site/library/reportPlusDashboard.rplus?country=USA&splus-action=view
…a SharePlus application action	splus://?splus-action=settings

URL Encoding

When working with SharePlus links, if the resource URL or the action's parameters include special characters that are not part of the ASCII character-set, you need to encode them.
For example, "=" must be replaced with "%3d".
You can encode/decode URL using free tools like http://meyerweb.com/eric/tools/dencoder/

Introduction

A bridge file called SharePlusBridge allows **communication between SharePlus and your Mobile Workspaces.** SharePlusBridge.js is a JavaScript file that includes a number of pre-defined APIs that you will use to achieve specific and controlled behavior. It is included in the WebDashboard section of the SharePlus Configuration File.

Invoking API methods

The SPlus prefix is always used to invoke the API and is placed before the methods.

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

Adding custom logic before your Mobile Workspace is displayed

When a SharePlus Mobile Workspace has finished loading, a function is called to notify that is ready and JavaScript functions are enabled. That function is *SharePlusOnLoad* and it allows you, for example, to load any custom settings that you may need.

Forcing the API to bypass SharePlus' cache

When invoking API methods, you can force them to bypass SharePlus cache and get SharePoint content directly from the server. For further details, refer to DataManagerContext.

Integrating SharePlus APIs to your Mobile Workspace

With this procedure, you will add functionality to display 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 mobile workspace to the bridge.

Overview

The process has 4 required steps:

1. Enabling the bridge in the Configuration File
2. Add a custom HTML page
3. (*Optional*) Including the *SharePlusOnLoad* function
4. Adding JavaScript
5. (*Optional*) Checking connection and URL availability
6. Displaying your SharePlus Mobile Workspace in 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 procedure.

Note: The *BridgeEnabled* property is enabled by default in most configuration files.

```
<key>WebDashboard</key>
<dict>
	<key>Settings</key>
	<dict>
		<key>BridgeEnabled</key>
		<true/>
	</dict>
</dict>
```

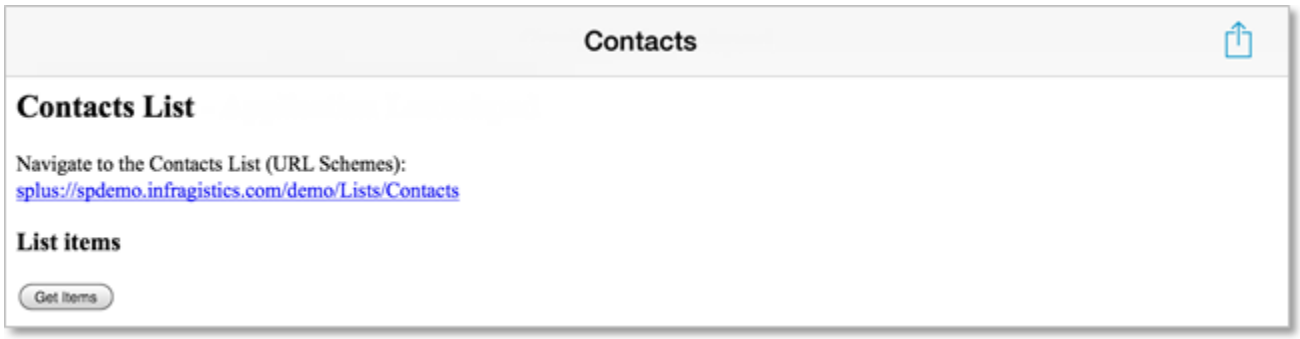
2. **Add a custom HTML page**
   In this example, you will create a SharePlus Mobile Workspace that communicates with **List.getItems** API function to retrieve information from the Contacts list in SharePoint's Demo site. These contact items will be displayed in a table. The URL Scheme to navigate to 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</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>
```

   Keep in mind that if a link to a SharePoint site is included in the HTML, that site **must** be configured within SharePlus before the content can be accessed.

   The HTML page should look similar to this one:



3. **(*Optional*) Include the *SharePlusOnLoad* function**
   This function is called to notify that the SharePlus Mobile Workspace is loaded and JavaScript functions are enabled. You can include JavaScript code here to add logic that needs to be executed before the mobile workspace is displayed. For example, you could 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. Add JavaScript

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 displayed
	}
}, 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 note that you can work without an internet connection. When working in offline mode and having all required resources already downloaded, you may work offline and synchronize your changes later.

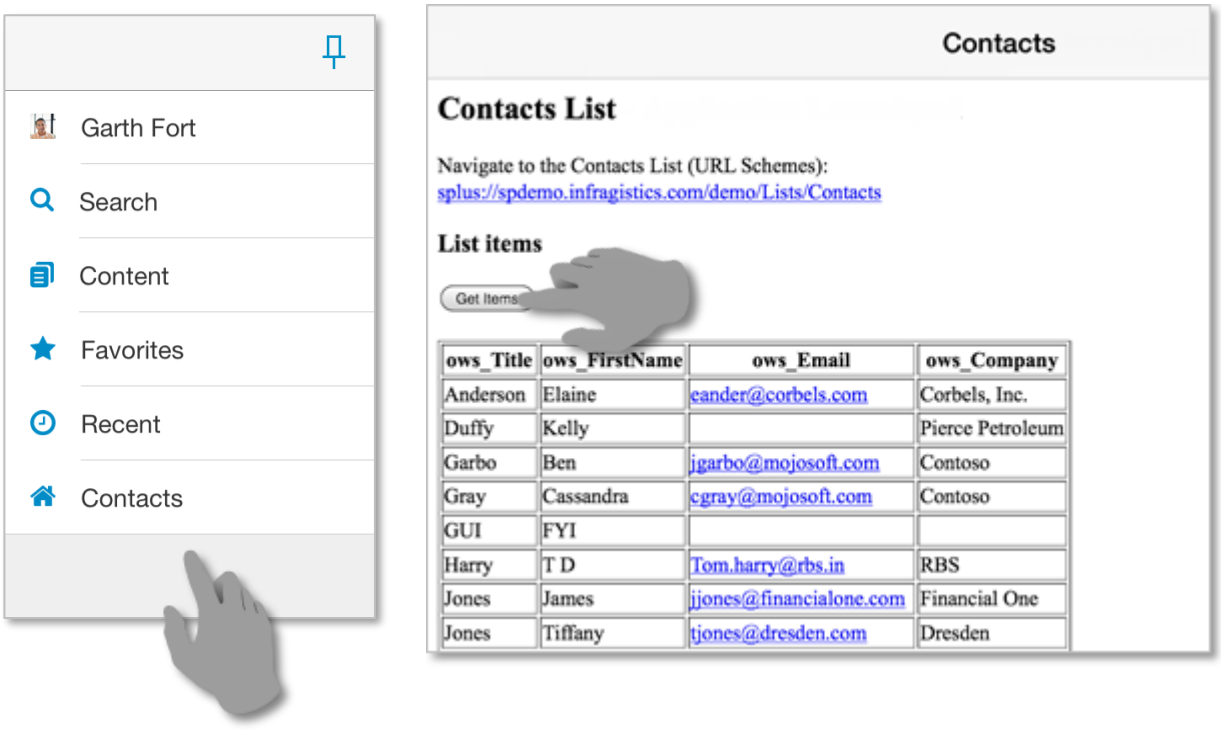
```
var url = 'http://spdemo.infragistics.com/demo';

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 SharePlus Mobile Workspace in SharePlus**

   a. Upload the ZIP file to SharePoint and configure it as the Application Home.
   For details about this refer to the last step of Creating your First SharePlus Mobile Workspace.

   b. Access the Application Home module in the SideBar.



This section includes two complete procedures that show you how to add behavior to your SharePlus implementation.

Summary

The following table provides high-level information about two full-fledged API scenarios.

Scenario	Details
Adding an item to a SharePoint list from your Mobile Workspace	With read/write SharePlus APIs, you can add, modify, or delete SharePoint content from Mobile Workspaces.
Integrating a custom form to a SharePoint list in SharePlus	As an alternative to SharePlus' native visualization, you can use a custom form to display, add, and edit items in a SharePoint list.

Adding an Item to a SharePoint list from your Mobile Workspace

With this procedure, you will create a new item in a specific SharePoint list from a Mobile Workspace. You will invoke one of the pre-defined API methods, *List.addItem*, and will learn how to make the connection from your workspace to the SharePoint list.

Overview

There are five steps in this process:

1. Adding a Mobile Workspace with custom HTML.
2. Retrieving List Information
3. Getting Ready List Information for *List.addItem*
4. Calling the *List.addItem* function.
5. Displaying your SharePlus Mobile Workspace in SharePlus.

Steps

1. **Add a Mobile Workspace with custom HTML**
   You will need a Mobile Workspace with, at least, an input field and a button to submit your information. 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>Please Send Us Your Feedback</h2>
	   <label for="task_name">Comments:</label>
	   <input id="task_name" type="text">
	   <button id="add_item_btn" onclick="addItem()" type="submit">Save</button>
	</body>
</html>
```

Keep in mind that if a link to SharePoint content is included in the HTML, that site **must** be configured within SharePlus before the content can be accessed. The JavaScript file with the addItem() method will be explained in depth during step 3.

The HTML page should look similar to this one:

2. **Retrieving List Information**

Creating an item from a Mobile Workspace implies you will not have any contextual information at the moment of creation (that is, which list you want the item to be created in, what type of item, etc.). You will therefore need to provide list information and the new item's field values with the following parameters:

- The internal name of the SharePoint list (GUID, **not** the display name).
- The SharePoint content type ID for the new item. This ID is unique to the items within the list.
- The URL to the SharePoint site containing the list where the item will be added.
- A JavaScript Object (attributes) with all the attributes for the new item.

2.1. Get the list GUID

If you don't have the list internal name, you can get it using the *Web.getWebsAndLists* API method. This method receives one parameter:

- URL to the SharePoint site containing the list.

The code snippet should be similar to the one below:

```
url = 'http://yourSharePoint.com/sitename';
SPlus.Web.getWebsAndLists (url, function (webs, lists){
      for (var j=0; j < lists.length; j++) {//Loop the lists Array
            var list = lists [j];//Getting a JSON object with several properties
            var listGUID = list['listName'];//Getting one of the properties of list
      }
}, function (errorResponse){
      //TODO: Implement how to handle errors
}, function (cancelResponse){
      //TODO: Implement how to handle a user's cancel
});
```

2.2. Get the list information

Use the *List.getListMetadata* API method, which receives a JavaScript Object as parameter with:

- GUID of the SharePoint list
- URL to the SharePoint site containing the list

Sample code snippet:

```
var args = {"listName": listName, //The GUID of the list value get from Web.getWebsAndLists
            "webUrL": webUrl};
SPlus.List.getListMetadata (args, function (result){
       var contentTypesArray = result['contentType'];
       for (var i=0; i< contentTypesArray.length; i++){ //Loop the array of content types
              var contentType = contentTypesArray[i]; //Get an object with one content type
              //For this example let's assume there is only one content type
              contentTypeID = contentType['id']; //Get the content type ID
              contentTypeFields = contentType['fields']; //Get the array of fields
              for (var j=0; j< contentTypeFields.length; j++){ //Loop the array of fields
                     var field = contentTypeFields [i]; //Get an object with one field
                     fieldName = field['name'];
                     fieldType = field['type'];
                     fieldRequired = field['required']; //Find out if a field is required
              }
       }
}, function (eResponse){
       //TODO: Implement how to handle error information
}, function (){
       //TODO: Implement how to handle a user's cancel
});
```

As shown above, using the *List.getListMetadata* API method you can get:

- The content type ID you want
- Information about the fields, including which one is required or not.

3. **Getting Ready List Information for *List.addItem***

You need to provide the list information and the new item's field values.

- The SharePoint *contentTypeId* for the new item being added.
- The GUID of the SharePoint list
- The *webUrl* for the SharePoint site containing the list where the item will be added.
- A JavaScript Object (attributes) with all the attributes for the new item as name:value pairs (properties).

The code snippet for these properties should be similar to the one below:

```
var taskName = document.getElementById("task_name").value; //Get Value to be used for the
                                                            Title field. The "task_name" id
                                                            was assigned to the input field
                                                            in step 1.
var attributesDictionary = {"Title": taskName}
var args = {"contentTypeId": contentTypeId, //Retrieved using List.getListMetadata
            "listName": listGUID, //Retrieved using Web.getWebsAndLists
            "webUrL": webUrl, //URL to the SharePoint site containing the list
            "attributes": attributesDictionary}; //New item's attributes (value:name pairs)
```

4. **Call the *List.addItem* method**

   You will call the List.addItem method from the API in order to create a new item in the SharePoint list. Include the following parameters:

   - A **success callback function** that receives a JavaScript Object (args) with several properties.
   - An **error callback function** that receives a JavaScript Object with the error description.
   - A **cancel callback function** that receives a JavaScript Object with cancel information.

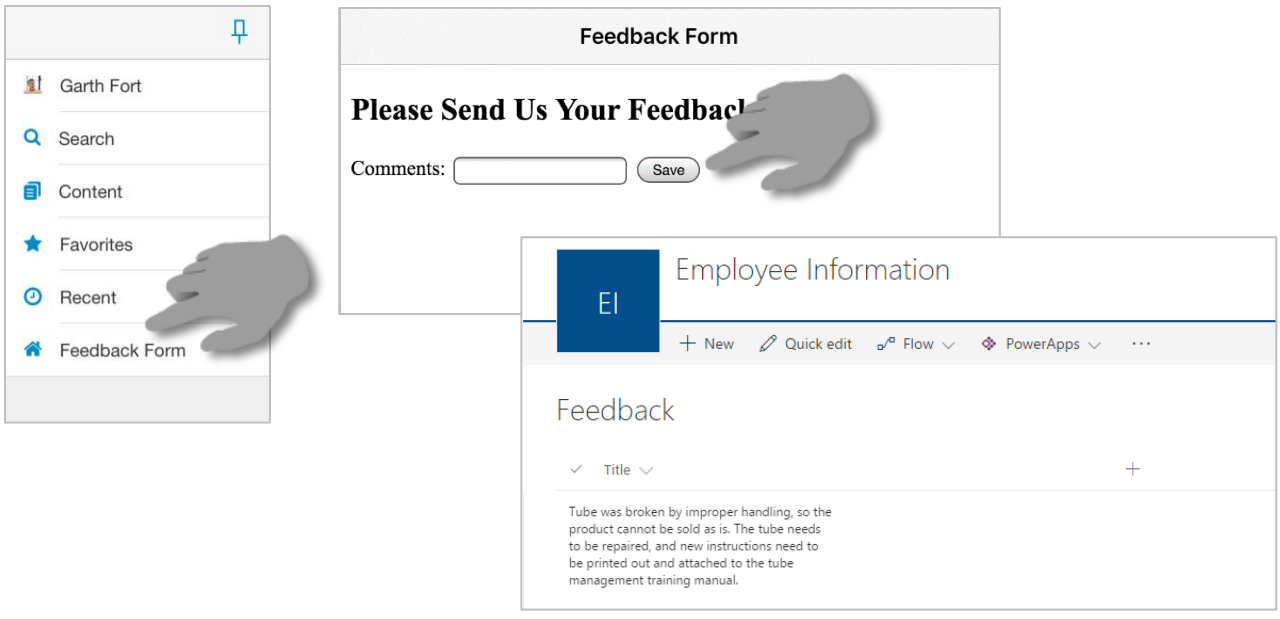
   Syntax of the API method:

```
SPlus.List.addItem (args, function (result){
      //TODO: Implement what to do when a new item was successfully added

}, function (eResponse){
      var errorInformation = eResponse['error#displayValue']; //Get error information
      var fieldErrors = eResponse['fieldErrors']; //Get a dict with all fields with errors
      for (var field in fieldErrors){ //Go through every field with errors
            for (var i=0; i<field.length; i++){ //Loop the array of errors for every field
                  var errorsArray = field[i]; //Get a JSON object with one of the errors
                  var errorDescription = errorsArray['description'];
                  var fieldName = errorsArray['field'];
                  var errorCode = errorsArray['code'];
                  //TODO: Implement what to do with the error information
            }
      }
}, function (){
      //TODO: Implement how to handle a user's cancel

});
```

5. **Display your SharePlus Mobile Workspace in SharePlus**

   c. Upload the ZIP file to SharePoint and configure it as the Application Home.
   For details about this refer to the last step of [Creating your First SharePlus Mobile Workspace.]

   d. Access the Application Home module in the SideBar.



Integrating a Custom Form to a SharePoint list in SharePlus

With this procedure, you will add functionality to display a Custom HTML Form for a specific SharePoint list within SharePlus. You will use pre-defined API methods *like SharePlusOnLoad*, *Forms.close, SharePlusCustomFormsOnCancel*, *List.addItem*, and *List.Item.update*. The Custom Form in this example is used to add or modify items in a Tasks list that has only two fields: Title and Status.

Overview

The process has seven steps:

1. Creating a Custom HTML Form.
2. Getting SharePlus contextual infomation.
3. Displaying the item being edited in your Custom Form.
4. Saving the Custom Form.
5. (*Optional*) Canceling the Custom Form.
6. Preparing the deployment package for SharePlus.
7. Configuring your Custom Form in SharePlus.

Steps

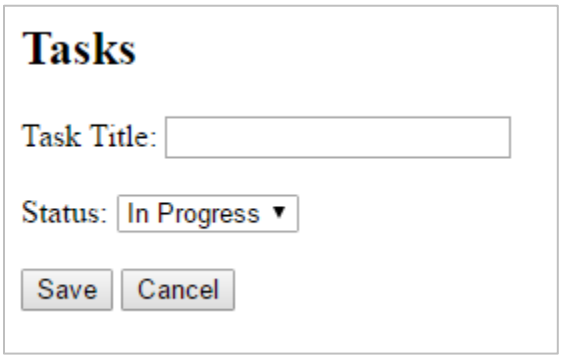
1. **Create a Custom HTML Form**
   Using a Custom Form, you will replace SharePlus' native visualization when displaying, adding, or modifying items in a SharePoint list. Create an *index.html* page with HTML content to add/modify items in a Tasks list.

   Sample code snippet for *index.html*:

```
...
	<body>
		<h2>Tasks</h2>
		//Use two labels to gather values (Title and Status)
		<label for="task_title">Task Title:</label>
		<input id="task_title" type="text">
		<br><br>
		<label for="task_status">Status:</label>
		<select id="task_status">
			<option>Not Started</option>
			<option selected>In Progress</option>
			<option>Completed</option>
		</select>
		<br><br>
		//Add two buttons (Save and Cancel)
		<button id="save_btn" onclick="saveForm()" type="submit">Save</button>
		<button id="cancel_btn" onclick="cancelForm()" type="button">Cancel</button>
	</body>
...
```

   Keep in mind that this will be a simple form with no CSS, images, or other assets. However, it can still be used to add new items, or modify existing ones, and call JavaScript code invoked from the buttons' onclick events.

Your custom form will look similar to this one:



2. **Get SharePlus contextual information**

Once you have the form created, you need to add functionality to the *SharePlusOnLoad* function. This function is called just before the custom form is displayed, and it's where you need to gather contextual information.

```
function SharePlusOnLoad(){
      initializeCustomForm();
};
```

Using the *Context.Current.Forms* property, you can get contextual information from the user's navigation. As you will see in the following code snippet, we retrieve two important pieces of information:

- result.customFormActiontype – The action (new, edit, display) triggered by the user through the UI.
- result.item – The item to be edited or an empty value if the user wants to add a new item.

```
var custom_form_action_mode;

...

function initializeCustomForm() {

    var contextObject = SPlus.Context.Current.Forms;
    //Get the user action (New, Edit, or Display)
    custom_form_action_mode = contextObject.customFormActionType;
    //Get a Dictionary with item information
    var item = contextObject.item;
    if (!item) {
       return;
    }
    //When an item is retrieved, display it to the user in your custom form
    displayItem(item);
}
```

In the code snippet above, a global variable, *custom_form_action_mode*, is used to identify the user action. If an item was retrieved, it means the user navigated through the UI to that specific item. Therefore, you need to display that item in you custom form so that the user can visualize or edit the item. To do this, the sample displayItem(item) function is invoked.

3. **Displaying the item being edited in your Custom Form**
   This function is not needed to add new items; only to display existing ones. When you are in Edit or Display mode, retrieve the item's attributes and load them in your custom form.

```
function displayItem(item) {
    //Get all attributes from the item Dictionary
    var attributes = item.attributes;
    if (!iattributes) {
        return;
    }
    //Load values in your custom form
    $("#task_title").val(attributes["Title"]);
    $("#task_status").val(attributes["Status"]);
    //When not editing, disable your labels and hide your buttons
    if (custom_form_action_mode == "Display") {
        $("#task_title").prop('disabled', true);
        $("#task_status").prop('disabled', true);
        $("#save_btn").hide();
        $("#cancel_btn").hide();
}
```

4. **Saving the Custom Form**

Now is the time to add JavaScript code to the sample saveForm() method. You need to retrieve the custom form values and send them as parameters to the SharePlus API methods (List.Item.updateItem or List.addItem).

```
function saveForm() {
    //Get values from your custom form
    var title = $("#task_title").val();
    var status = parseInt($("#task_status").val());
    //Get ready the args Object, used as parameter for List.Item.updateItem / List.addItem
    var attributes = {
        "Title": title,
        "Status": status
    };
    var args = {
        "attributes": attributes
    }
    //Call List.Item.updateItem or List.addItem depending on the action the user is
performing
    if (custom_form_action_mode == "New") {
          SPlus.List.addItem(args, function(result) {
              //On success, call the API method to close the web view and refresh the list
               SPlus.Forms.close();
          }, function(response) {
               //TODO: Implement how to handle errors
          }, function(response) {
               //TODO: Implement how to handle the user's cancel
          });
    } else {
          SPlus.List.Item.updateItem(args, function(result) {
              //On success, call the API method to close the web view and refresh the list
               SPlus.Forms.close();
          }, function(response) {
               //TODO: Implement how to handle errors
          }, function(response) {
               //TODO: Implement how to handle the user's cancel
          });
    }
}
```

The args Object can include several properties. The only required property is a dictionary with the new attributes of the item. The other properties can provide contextual information like the site or list URLs, and they are relevant when there is no user navigation and you are adding/editing an item without context.

The *Forms.close* API method is used to close the web view in which SharePlus displays the custom form. The list is refreshed automatically after the form is closed.

5. **(Optional) Canceling the Custom Form**

   You can add code to a cancelForm () function invoked from the *Cancel* button.

```
function cancelForm(){
      SPlus.Forms.close();
};
```

Alternatively, you may want to perform some actions every time you leave the custom form; for example, when the user navigates back to the list. In that case, you can, for example, display a message alerting the user that all changes will be lost or just ask for a confirmation. To do this, you need to implement the *SharePlusCustomFormsOnCancel* API method.

```
function SharePlusCustomFormsOnCancel(){
      If (confirm ("Are you sure")){
            SPlus.Forms.close(); //Close the custom form
      }
};
```

When the *SharePlusCustomFormsOnCancel* function is implemented, the custom form won't be closed automatically after navigating back through the UI. Instead, you have to close it manually using the *Forms.close* function.

6. **Prepare your deployment package for SharePlus**
   a. Create a ZIP file
      Prepare your deployment package by adding all necessary files for your Custom Form to work as expected. In our example, you will only have *index.html* in the compressed ZIP file.
   b. Upload your file to SharePoint
      Add your ZIP file to a SharePoint Document Library, making it accessible from the SharePlus app.
      For further details, refer to the **Deployment** section.
7. **Configuring your Custom Form in SharePlus**

   Once you have your ZIP file ready and accessible, you have to configure the package as a custom form for the Tasks list in the Configuration File. To do this, you need to:
   a. Create a new Page in the config file that includes your custom form.
   b. Reference the new Page and specify where the custom form will be applied.

   For further details about this, refer to **Custom Forms**.

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 the 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
Attaching files to an item	You can prompt the user to select a file and later add, update, or remove attachments from items.	*Utility.pickDocument, List.addItem, List.Item.updateItem*
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 SharePlus Mobile Workspaces Settings	Settings can be modified, stored in SharePlus, and later loaded when needed to.	*WebDashboard.Settings.save, WebDashboard.Settings.load*
Changing the sources of a SharePlus Mobile Workspace	You can change it permanently or temporary, depending on your needs.	*WebDashboard.setSource, WebDashboard.navigate*

Attaching files to an item

Overview

The Utility.pickDocument JavaScript API method allows you to display the iOS document picker to the user to select new files. Every time a file is added to the files collection, the method returns the file name to be used later with List.addItem or List.Item.updateItem. When invoking List.addItem, you will include the attachments property in the JavaScript object parameter sent to the method.

Code

Use *Utility.pickDocument* to add files to the files collection and store the filename(s) in a JavaScript array. If you want, you can display the filename to the user to let him know the file will be attached to the item.

The code should be similar to the following code snippet:

```
// You need to define the array of filenames
var attachmentsArray = [];

function addAttachment() {
	SPlus.Utility.pickDocument(function(result) {
		// get the String with the file name and add the filename to the array
		var filename = result;
		attachmentsArray.push(filename);
		// display the aray of filenames in a label to the user
		$("label[for='attachment_filenames']").html(attachmentsArray.toString());
	}, function(response) {
		//TODO: Implement how to handle errors
	}, function(response) {
		//TODO: Implement how to handle a user's cancel
	});
}
```

Once you selected the file(s) to be attached to the item, you will include the attachments property in the JavaScript object parameter sent to *List.addItem* or *List.Item.updateItem* as shown below:

```
//Create a dictionary with all the item's attributes
var attributesDictionary = {"Title": taskName, "StartDate": startDate};

//Create a dictionary with the attachments' information
var attachmentsDictionary = new Object()
for (var i = 0; i < attachmentsArray.length; i++){
      filename = attachmentsArray[i];
      attachmentsDictionary[filename] = "Add";
}

//Get ready the args parameter that will be used with List.addItem
var args = {"attachments": attachmentsDictionary, "attributes": attributesDictionary}
```

As seen above, you need to create an object with the attachments of the new item as filename:operation pairs. The object structure is similar to the following:

```
{filename1:'Add',filename2:'Add',filename3:'Add'}
```

But when editing an item with *List.Item.updateItem*, you may need to change or delete a file from that item. To achieve those, you need to use "Update" or "Remove" instead of "Add". Below you can find an example:

```
{filename1:'Add',filename2:'Update',filename3':Remove',filename4:'Add'}
```

As a reference you have the following **operations** available:

- Add – Attach the referenced file to the item.
- Update – Change an existing file for another one.
- Remove – Delete an existing file referenced by filename.

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.

You are now 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 the items somewhere
    }
}, 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
});
```

Persisting SharePlus Mobile Workspaces Settings

Overview

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

Code

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

- Create the settings *JSON Object*.
- Specify the mobile workspace'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 = 'MobileWorkspaceOne'; // unique key, identifies specific 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 mobile workspaces 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 = 'MobileWorkspaceOne'; // Key used to identify the specific 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 a specific SharePlus Mobile Workspace.
- An error callback function, which receives a *JavaScript object* with the error description.

Changing the source of a SharePlus Mobile Workspace

Overview

The **WebDashboard.setSource** and **WebDashboard.navigate** JavaScript API methods allow you to change your mobile workspace. 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 mobile workspace that will be persistent even if SharePlus is shut down.

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

Code

In the code snippet below, you:

- Specify the static URL to download the file.
- 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.

```
var source = ‘http://spdemo/demo/SiteAssets/MobileWorkspaces/Demo.web.zip‘;
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 mobile workspaces that are included in the Application Home.

API Reference Table

Category	API Property	Introduced in version
List	List.addItem	4.6
	List.addDocument	4.6
	List.Item.updateItem	4.6
	List.Item.updateDocument	4.6
	List.Item.delete	4.6
	List.getListMetadata	4.6
	List.Item.download	3.9
	List.getItems	3.9
	List.getItemsWithOptions	3.9
Forms	Forms.close	4.6
Context	Context.Current.Forms	4.6
	Context.Current.siteTitle	4.0
	Context.Current.siteUrl	4.0
	Context.Current.webTitle	4.0
	Context.Current.webUrl	4.0
Hook methods	SharePlusCustomFormsOnCancel	4.6
	SharePlusOnLoad	4.6
Object Type	DataManagerContext	4.2
Utility	Utility.pickDocument	4.6
	Utility.getLocalizedString	3.9
	Utility.showMessage	3.9
	Utility.URL.open	3.9
	Utility.URL.download	3.9
Search	Search.query	4.1
	Search.queryWithOptions	4.1
	Search.getScopes	4.1

Category	API Property	Introduced in version
Taxonomy	Taxonomy.getTermsByLabel	4.1
	Taxonomy.getTermsByLabelWithOptions	4.1
	Taxonomy.getTermsInWeb	4.1
Connection	Connection.isOnline	3.9
	Connection.isConnected	3.9
	Connection.setOnLine	4.1
Web	Web.getWebsAndLists	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
Configuration	Configuration.setRemoteFileSource	3.9
	Configuration.getFeature	3.9
	Configuration.getVariables	3.9
	Configuration.reload	3.9

List.addItem

This method is used to create a new item in a SharePoint List. As parameter, it receives a JavaScript object that can include several properties. When the parameter has no properties with contextual information, this method relies on user navigation to get the context. For example, after the user navigates to a specific list within a site and tries to add an item, SharePlus can get the site URL, the listName, and the content type.

You can use List.addItem in two different ways:

1. From a Mobile Workspace, where you need to provide contextual information like the site URL and list name. In this case you can use List.getListMetadata to get the content types and required fields from the list. In the case that you need the GUID of the list, you can get it using the Web.getWebsAndLists method.
2. From a Custom Form defined for a list, where you already have contextual information available from the user navigation through the UI.

Syntax

```
SPlus.List.addItem (args, function (result){
	//TODO: Implement what to do when a new item was successfully added
}, function (eResponse){
	//TODO: Implement how to handle error information
}, function (){
	//TODO: Implement how to handle a user's cancel
});
```

List.addItem Parameters

Parameter	Type	Description
args	JavaScript Object	The args Object includes a property with the attributes of the new item. And, optionally, contextual information to be used by SharePlus to find the list and add the new item.

args Object Structure

Property	Required	Type	Description
contentTypeId	See note	String	SharePoint content type's ID that is unique within a site collection.
listName	See note	String	GUID of the SharePoint list (**not** the display name).
webUrl	See note	String	URL to the SharePoint site containing the list.
attachments	No	Object	The attachments of the new item as filename:operation pairs (properties)
attributes	**Yes**	Object	The attributes of the new item as name:value pairs (properties)

Note: Not required when using a custom form defined for a list, where you already have contextual information available from the user navigation through the UI.

args.attributes Object structure example

```
{'Title':taskName,'StartDate':sDate,'Priority':priority,'Assigned_x0020_To':assigned}
```

Note: When the attribute you are including is a lookup field, you need to use a custom data type.

```
var lookupid = country_id;
var lookupvalue = country_value;
var countryField = new SPlusBridgeDataType.SPLookupValue(lookupid, lookupvalue);
```

Example with a lookup field:

```
{'Title':taskName,'StartDate':sDate,'Country':countryField}
```

args.attachments Object structure example

```
{filename1:'add',filename2:'add',filename3:'add'}
```

onSuccess callback function

Parameter	Description
function (result)	Success callback function to be implemented. When successful, *result* returns as an empty dictionary.

onError callback function

Parameter	Description
function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.
fieldErrors	Dictionary	Dictionary with all the fields with errors.

eResponse.fieldErrors Object structure example

```
{'field1':errorsArray,'field2':errorsArray, ... 'fieldN':errorsArray}
```

eResponse.fieldErrors.errorsArray Object structure

Property	Type	Description
description	String	Description of the error
field	String	Field name.
code	Number	Numeric code for the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

List.addDocument

This method is used to create a new document in a Document Library. As parameter, it receives a JavaScript object that can include several properties. When the parameter has no properties with contextual information, this method relies on user navigation to get the context. For example, after the user navigates to a specific library within a site and tries to add a document, SharePlus can get the site URL, the library GUID, and the content type.

You can use List.addDocument in two different ways:

1. From a Mobile Workspace, where you need to provide contextual information like the site URL and library GUID. In this case you can use List.getListMetadata to get the content types and required fields from the library. In the case that you need the GUID of the library, you can get it using the Web.getWebsAndLists method.
2. From a Custom Form defined for a library, where you already have contextual information available from the user navigation through the UI.

Syntax

```
SPlus.List.addDocument (args, function (result){
      //TODO: Implement what to do when a new document was successfully added
}, function (eResponse){
      //TODO: Implement what to do with the error information
}, function (){
      //TODO: Implement how to handle a user's cancel
});
```

List.addDocument Parameters

Parameter	Type	Description
args	JavaScript Object	The args Object includes a property with the file name and the attributes of the new document. And, optionally, contextual information to be used by SharePlus to find the library and add the new document.

args Object Structure

Property	Required	Type	Description
contentTypeId	See note	String	SharePoint content type's ID that is unique within a site collection.
listName	See note	String	GUID of the SharePoint library (**not** the display name).
webUrl	See note	String	URL to the SharePoint site containing the library.
document	**Yes**	String	File name that references the file to be used as document.
attributes	**Yes**	Object	The attributes of the new document as name:value pairs (properties)

Note: Not required when using a custom form defined for a library, where you already have contextual information available from the user navigation through the UI.

args.attributes Object structure example

```
{'Name':documentTitle,'Status':status,'Assigned_x0020_To':assigned}
```

onSuccess callback function

Parameter	Description
function (result)	Success callback function to be implemented. When successful, *result* returns as an empty dictionary.

onError callback function

Parameter	Description
function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.
fieldErrors	Dictionary	Dictionary with all the fields with errors.

eResponse.fieldErrors Object structure example

```
{'field1':errorsArray,'field2':errorsArray, ... 'fieldN':errorsArray}
```

eResponse.fieldErrors.errorsArray Object structure

Property	Type	Description
description	String	Description of the error
field	String	Field name.
code	Number	Numeric code for the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

List.Item.updateItem

This method is used to modify an existing item in a SharePoint List. As parameter, it receives a JavaScript object that can include several properties. When the parameter has no properties with contextual information this method relies on user navigation to get the context. For example, after the user navigates to a specific list within a site and tries to modify an item, SharePlus can get the site URL, the list GUID, the content type, and the item's ID.

You can use List.Item.update in two different ways:

1. From a Mobile Workspace, where you need to provide contextual information like the site URL and list name. In this case you can use List.getListMetadata to get the content types and required fields from the list. In the case that you need the GUID of the list, you can get it using the Web.getWebsAndLists method.
2. From a Custom Form defined for a list, where you already have contextual information available from the user navigation through the UI.

Syntax

```
SPlus.List.Item.updateItem (args, function (result){
	//TODO: Implement what to do when an item was successfully edited
}, function (eResponse){
	//TODO: Implement what to do with the error information
}, function (){
	//TODO: Implement how to handle a user's cancel
});
```

List.Item.updateItem Parameters

Parameter	Type	Description
args	JavaScript Object	The args Object includes a property with the attributes of the item to be edited. And, optionally, contextual information to be used by SharePlus to find the list and the item.

args Object Structure

Property	Required	Type	Description
listName	See note	String	GUID of the SharePoint list (**not** the display name).
webUrl	See note	String	URL to the SharePoint site containing the list.
itemId	See note	String	ID of the SharePoint item to be edited.
attachments	No	Object	The attachments of the item as filename:operation pairs (properties)
attributes	**Yes**	Object	The attributes of the item as name:value pairs (properties)

Note: Not required when using a custom form defined for a list, where you already have contextual information available from the user navigation through the UI.

args.attributes Object structure example

```
{'Title':taskName,'StartDate':sDate,'Priority':priority, Assigned_x0020_To':assigned}
```

Note: When the attribute you are including is a lookup field, you need to use a custom data type. For further details about using lookup fields refer to the **Note** in the List.AddItem method.

args.attachments Object structure example

```
{filename1:'add',filename2:'update',filename3:'remove',filename4:'add'}
```

onSuccess callback function

Parameter	Description
function (result)	Success callback function to be implemented. When successful, *result* returns as an empty dictionary.

onError callback function

Parameter	Description
function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.
fieldErrors	Dictionary	Dictionary with all the fields with errors.

eResponse.fieldErrors Object structure example

```
{'field1':errorsArray,'field2':errorsArray, ... 'fieldN':errorsArray}
```

eResponse.fieldErrors.errorsArray Object structure

Property	Type	Description
description	String	Description of the error
field	String	Field name.
code	Number	Numeric code for the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

List.Item.updateDocument

This method is used to modify an existing document in a SharePoint Document Library. As parameter, it receives a JavaScript object that can include several properties. When the parameter has no properties with contextual information this method relies on user navigation to get the context. For example, after the user navigates to a specific list within a site and tries to modify a document, SharePlus can get the site URL, the library GUID, the content type, and the document's ID.

You can use List.Item.update in two different ways:

1. From a Mobile Workspace, where you need to provide contextual information like the site URL and library GUID. In this case you can use List.getListMetadata to get the content types and required fields from the library. In the case that you need the GUID of the library, you can get it using the Web.getWebsAndLists method.
2. From a custom form defined for a library, where you already have contextual information available from the user navigation through the UI.

Syntax

```
SPlus.List.Item.updateDocument (args, function (result){
	//TODO: Implement what to do when an item was successfully edited
}, function (eResponse){
	//TODO: Implement what to do with the error information
}, function (){
	//TODO: Implement how to handle a user's cancel
});
```

List.Item.updateDocument Parameters

Parameter	Type	Description
args	JavaScript Object	The args Object includes a property with the file name and the attributes of the document. And, optionally, contextual information to be used by SharePlus to find the library and the existing document.

args Object Structure

Property	Required	Type	Description
contentTypeId	See note	String	SharePoint content type's ID that is unique within a site collection.
listName	See note	String	GUID of the SharePoint library (**not** the display name).
webUrl	See note	String	URL to the SharePoint site containing the library.
itemId	See note	String	ID of the SharePoint item to be edited.
document	**Yes**	String	File name that references the new file to be used as document.
attributes	**Yes**	Object	The attributes of the item as name:value pairs (properties)

Note: Not required when using a custom form defined for a library, where you already have contextual information available from the user navigation through the UI.

args.attributes Object structure example

```
{'Name':documentTitle,'Status':status,'Assigned_x0020_To':assigned}
```

onSuccess callback function

Parameter	Description
function (result)	Success callback function to be implemented. When successful, *result* returns as an empty dictionary.

onError callback function

Parameter	Description
function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.
fieldErrors	Dictionary	Dictionary with all the fields with errors.

eResponse.fieldErrors Object structure example

```
{'field1':errorsArray,'field2':errorsArray, ... 'fieldN':errorsArray}
```

eResponse.fieldErrors.errorsArray Object structure

Property	Type	Description
description	String	Description of the error
field	String	Field name.
code	Number	Numeric code for the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

List.Item.delete

This method can be used to delete either an existing item in a SharePoint List or a document in a Document Library. As parameter, it receives a JavaScript object that can include several properties. When the parameter has no properties with contextual information this method relies on user navigation to get the context. For example, after the user navigates to a specific list within a site and tries to delete an item or document , SharePlus can get the site URL, the list or library GUID, the content type, and the item or document GUID.
You can use List.Item.delete in two different ways:

1. From a Mobile Workspace, where you need to provide contextual information like the site URL and list name.
2. From a Custom Form defined for a list, where you already have contextual information available from the user navigation through the UI.

Syntax

```
SPlus.List.Item.delete (args, function (result){
       //TODO: Implement what to do when an item/document was successfully deleted
}, function (eResponse){
       //TODO: Implement what to do with the error information
}, function (){
       //TODO: Implement how to handle a user's cancel
});
```

List.Item.delete Parameters

Parameter	Type	Description
args	JavaScript Object	The args Object includes a property with the GUID of the item or document to be deleted. And, optionally, contextual information to be used by SharePlus to find the list and the existing item.

args Object Structure

Property	Required	Type	Description
listName	See note	String	GUID of the SharePoint list/library (**not** the display name).
webUrl	See note	String	URL to the SharePoint site containing the list/library.
itemId	See note	String	ID of the SharePoint item to be deleted.

Note: Not required when using a custom form defined for a library, where you already have contextual information available from the user navigation through the UI.

onSuccess callback function

Parameter	Description
function (result)	Success callback function to be implemented. When successful, *result* returns as an empty dictionary.

onError callback function

Parameter	Description
function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

List.getListMetadata

This method is used to get information from a SharePoint list. As parameter, it receives a JavaScript object with two properties: the GUID of the SharePoint list and the URL to the SharePoint site containing that list.

You can use List.getListMetadata to get valuable information from a list before adding or updating an item to that list. For example, you get the content types and required fields from the list. In the case that you need the GUID of the list, you can get it using the Web.getWebsAndLists method.

Syntax

```
SPlus.List.getListMetadata (args, function (result){
      //TODO: Implement what to do with the list information
}, function (eResponse){
      //TODO: Implement how to handle error information
}, function (){
      //TODO: Implement how to handle a user's cancel
});
```

List.getListMetadata Parameters

Parameter	Type	Description
args	JavaScript Object	The args Object includes two properties, the GUID of the SharePoint list and the URL to the SharePoint site containing that list.

args Object Structure

Property	Required	Type	Description
listName	**Yes**	String	GUID of the SharePoint list (**not** the display name).
webUrl	**Yes**	String	URL to the SharePoint site containing the list.

onSuccess callback function

Parameter	Description
function (result)	Success callback function to be implemented, which retrieves the result JavaScript Object

result Object Structure

Property	Type	Description
contentType	Array	Array of existing content types.

result.contentType Object structure

Property	Type	Description
id	String	Content type identifier.
name	String	Content type name.
description	Number	Content type description.
fields	Array	Array of existing fields

result.contentType.fields Object structure

Property	Type	Description
name	String	The field's name.
type	String	The field's type.
required	Boolean	True when the field is required.
…	…	…

onError callback function

Parameter	Description
function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

Utility.pickDocument

This method is used to select a document, add it to the files collection and return the filename for later use.
The standard iOS document picker is displayed to the user to select a new file. Once the file was added to the files collection, the onSuccess part of the method returns a String with the file name to be displayed and/or used later. The file name is later needed to add an attachment to an item or to add a new document.

Syntax

```
SPlus.Utility.pickDocument (function (result){
	var filename = result;
	//TODO: Implement what to do with the filename of a file successfully added to the files
	collection

}, function (eResponse){
	var errorInformation = eResponse['error#displayValue']; //Get error information
	//TODO: Implement what to do with the error information
}, function (){
	//TODO: Implement how to handle a user's cancel
});
```

onSuccess callback function

Parameter	Description
function (result)	Success callback function to be implemented. When successful, result returns file name of the file added to the files collection.

result Object Structure

Property	Type	Description
filename	String	File name of the file successfully added to the files collection.

onError callback function

Parameter	Description
function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

Forms.close

This method is used to close the web view in which SharePlus displays a custom form.

Syntax

```
SPlus.Forms.close ();
```

Context.Current.Forms

This property is used to get contextual information from the user's navigation within SharePlus. E.g., After the user navigates to a specific list within a site and tries to edit an item, SharePlus can get the site URL, the listName, the content type, and the item's GUID.

Syntax

```
var contextObject = SPlus.Context.Current.Forms; // Get the object with contextual information
```

Returned Object Structure

Property	Type	Description
customFormActionType	String	The action type the user is performing (New, Edit, or Display)
web	Dictionary	Dictionary with web information.
contentType	Dictionary	Dictionary with content type information.
list	Dictionary	Dictionary with list information.
item	Dictionary	Dictionary with item information.

.web Object Structure

Property	Type	Description
url	String	URL to the SharePoint site containing the list.

.contentType Object Structure

Property	Type	Description
id	String	Content type identifier.
name	String	Content type name.
description	String	Content type description.
fields	Array	Array of fields.

.contentType.field Object Structure

Property	Type	Description
name	String	The field's name.
type	String	The field's type.
required	Boolean	True when the field is required.
…	…	…

.list Object Structure

Property	Type	Description
name	String	The list's name
title	String	The list's title
…	…	…

.item Object Structure

Property	Type	Description
id	String	GUID of the SharePoint item.

attributes	Dictionary	Dictionary with values for every field
attachments	Dictionary	Dictionary with the filename of every attachment.

.item.attributes Object Structure example

```
{'field1':value1,'field2':value2, ... 'fieldN':valueN}
```

.item.attachments Object structure example

```
{filename1,filename2,filename3, ... filenameN}
```

SharePlusCustomFormsOnCancel

This hook method can be implemented or not. When implemented, the method is called after the user navigates back in a Custom Form through the UI. In this case, the form displayed in SharePlus won't be automatically closed. Instead, you have to close the form manually using the *Forms.close* method.

Syntax

```
function SharePlusCustomFormsOnCancel (){
	//TODO: Validate custom form cancel
	SPlus.Forms.close (); //Close the custom form.
}
```

SharePlusOnLoad

This hook method can be implemented or not. When implemented, the method is called to notify that your Mobile Workspace or Custom Form hs finished loading and JavaScript functions are enabled. You can use SharePlusOnLoad to add custom logic before your Mobile Workspace or Custom Form is displayed on screen.

Syntax

```
function SharePlusOnLoad (){
	//TODO: Add custom logic before the Custom Form or Mobile Workspace is displayed
}
```

Search.query

This method is used to search resources within a SharePoint site and can receive an optional text (String) to search for.

Syntax

```
SPlus.Search.query (url, text, function (resources){
	for (var i=0; i < resources.length; i++) {//Loop the resources Array
		var resource = resources [i];//Getting a JSON object with the requested fields
		var title = resource['title']; //Getting one of the fields
		var properties = resource['properties']; //Getting the properties array
		for (var i=0; i < properties.length; i++){
		//TODO: Implement what to do with each one of the resource's properties
		}
	//TODO: Implement what to do with each resource and its properties
	}
}, 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
});
```

Search.query Parameters

Parameter	Type	Description
url	String	URL to the SharePoint site.
text	String	Optional text parameter used to search the site.

onSuccess callback function

Parameter	Description
function (resources)	Success callback function to be implemented, which retrieves the resources array containing JSON Objects with the results from the SharePoint site.

resources Array

Property	Type	Description
resource	Array	Array of existing resources retrieved from the SharePoint site.

resources.resource Object structure

Property	Type	Description
url	String	URL to the resource
title	String	Title of the resource
properties	Array	Resource properties

resources.resource.properties Object structure

Property	Type	Description
size	String	Resource size in bytes
path	String	Path to the resource
write	String	Last time the resource was modified
author	String	The resource's creator.
isdocument	String	"true" when the resource is a document
fileextension	String	File's suffix, e.g., "DOCX"
contentclass	String	List or library type
title	String	Title of the resource

onError callback function

Parameter	Description
function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

Search.queryWithOptions

This method is used to search resources within a SharePoint site. Receives a number of parameters to refine the search.

Syntax

```
SPlus.Search.queryWithOptions (url, options, function (resources){
	for (var i=0; i < resources.length; i++) {//Loop the resources Array
		var resource = resources [i];//Getting a JSON object with the requested fields
		var title = resource['title']; //Getting one of the fields
		var properties = resource['properties']; //Getting the properties array
		for (var i=0; i < properties.length; i++){
		}
	//TODO: Implement what to do with each resource and its properties
	}
}, 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
});
```

Search.queryWithOptions Parameters

Parameter	Type	Description
url	String	URL to the SharePoint site.
options	JavaScript Object	The custom Object options includes optional information to be used in the query to the SharePoint list.

options Object structure

Property	Type	Description
text	String	Specific text to search for
author	String	The document's creator.
resultType	String	The expected type, e.g.: "Documents", "Items", "All Results", "Word Documents", "PDF".
modifiedDate	String	Last time the resource was modified
scope	String	The relevant scope to search, e.g.: "People", "All Sites", "This Site".

onSuccess callback function

Parameter	Description
function (resources)	Success callback function to be implemented, which retrieves the resources array containing JSON Objects with the results from the SharePoint site.

resources Array

Property	Type	Description
resource	Array	Array of existing resources retrieved from the SharePoint site.

resources.resource Object structure

Property	Type	Description
url	String	URL to the resource
title	String	Title of the resource
properties	Array	Resource properties

resources.resource.properties Object structure

Property	Type	Description
size	String	Resource size in bytes
path	String	Path to the resource
write	String	Last time the resource was modified
author	String	The resource's creator.
isdocument	String	"true" when the resource is a document
fileextension	String	File's suffix, e.g., "DOCX"
contentclass	String	List or library type
title	String	Title of the resource

onError callback function

Parameter	Description
function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

Note: When using the Search.queryWithOptions method, you may want to specify the scope. As scopes vary depending on the site, it is highly recommended that you first retrieve all scopes for a site using the Search.getScopes method.

Search.getScopes

This method is used to retrieve the existing scopes for a SharePoint site.

Syntax

```
SPlus.Search.getScopes (url, function (scopes){
	for (var i=0; i < scopes.length; i++) {//Loop the scopes Array
		var scope = scopes [i];//Getting one of the requested scopes
		//TODO: Implement what to do with the scopes
	}
}, 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
});
```

Search.getScopes Parameters

Parameter	Type	Description
url	String	URL to the SharePoint site.

onSuccess callback function

Parameter	Description
function (scopes)	Success callback function to be implemented, which receives an Array with the scopes.

scopes Array

Property	Type	Description
scopes	Array	Retrieved results from the SharePoint site. Each scope corresponds to one of the site's scopes and values are returned with String type.

onError callback function

Parameter	Description
Function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

Taxonomy.getTermsByLabel

This method is used to retrieve managed metadata from a SharePoint site. You can get all the existing terms containing a specific text.

Syntax

```
SPlus.Taxonomy.geTermsByLabel (url, label, function (terms){
	for (var i=0; i < terms.length; i++) {//Loop the terms Array
		var term = terms [i];//Getting one of the requested terms
		//TODO: Implement what to do with the terms
	}
}, 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
});
```

Taxonomy.getTermsByLabel Parameters

Parameter	Type	Description
url	String	URL to the SharePoint site.
label	String	Text (letters or words) to be searched in the existing terms.

onSuccess callback function

Parameter	Description
function (terms)	Success callback function to be implemented, which receives an Array with the terms.

terms Array

Property	Type	Description
terms	Array	Retrieved results from the SharePoint site. Each term corresponds to one of the site's terms and values are returned with String type.

onError callback function

Parameter	Description
Function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

Taxonomy.getTermsByLabelWithOptions

This method is used to retrieve managed metadata from a SharePoint site. You can get all the existing terms that contain a specific text or match it exactly. Also, when the search is unsuccessful you can optionally add the term to the site.

Syntax

```
SPlus.Taxonomy.geTermsByLabelWithOptions (url, label, exactMatch, addIfNotFound, function
(terms){
	for (var i=0; i < terms.length; i++) {//Loop the terms Array
		var term = terms [i];//Getting one of the requested terms
		//TODO: Implement what to do with the terms
	}
}, 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
});
```

Taxonomy.getTermsByLabelWithOptions Parameters

Parameter	Type	Description
url	String	URL to the SharePoint site.
label	String	Text (letters or words) to be searched in the existing terms.
exactMatch	Boolean	When true, the returned term must match the label exactly.
addIfNotFound	Boolean	When true and the term is not found, you add the term to the site.

onSuccess callback function

Parameter	Description
function (terms)	Success callback function to be implemented, which receives an Array with the terms.

terms Array

Property	Type	Description
terms	Array	Retrieved results from the SharePoint site. Each term corresponds to one of the site's terms and values are returned with String type.

onError callback function

Parameter	Description
Function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

Taxonomy.getTermsInWeb

This method is used to retrieve managed metadata from a SharePoint site. You can get all the existing terms for a site.

Syntax

```
SPlus.Taxonomy.getTermsInWeb (url, termSetId, storeId, function (terms){
	for (var i=0; i < terms.length; i++) {//Loop the terms Array
		var term = terms [i];//Getting one of the requested terms
		//TODO: Implement what to do with the terms and it's child terms
	}
}, 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
});
```

Taxonomy.getTermsInWeb Parameters

Parameter	Type	Description
url	String	URL to the SharePoint site..
termSetId	String	Term Set collection identifier. This value specifies a set of Term objects (related terms).
storeId	String	Term store identifier. This value references a database that contains all the metadata.

onSuccess callback function

Parameter	Description
function (terms)	Success callback function to be implemented, which receives an Array with the terms.

terms Array

Property	Type	Description
terms	Array	Retrieved results from the SharePoint site. Each **term** corresponds to one of the site's terms and values are returned with String type.

terms.term Object structure

Property	Type	Description
title	String	The term's text.
id	String	Guid that identifies the term.
childs	Array	Array of child term objects.
path	String	Path to the current Term object.
wssid	String	The term's Id in the list of managed terms in use (hidden).

terms.term.child Object structure

Property	Type	Description
title	String	The child term's text
Id	String	Guid that identifies the term
path	String	Path to the current Term object
wssid	String	The term's Id in the list of managed terms in use (hidden)

onError callback function

Parameter	Description
function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

Context.Current.siteTitle

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

Syntax

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

Context.Current.siteTitle Parameters

None

Context.Current.siteUrl

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

Syntax

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

Context.Current.siteUrl Parameters

None

Context.Current.webTitle

This property is used to get the title of the site to which the SharePlus Mobile Workspace is assigned.
When the mobile workspace is assigned to the Application Home, an empty value is retrieved.

Syntax

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

Context.Current.webTitle Parameters

None

Context.Current.webUrl

This property is used to get the URL of the site to which the SharePlus Mobile Workspace is assigned.
When the mobile workspace is assigned to the Application Home, an empty value is retrieved.

Syntax

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

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
});
```

Web.getWebsAndLists Parameters

Parameter	Type	Description
url	String	URL to the SharePoint site.

onSuccess callback function

Parameter	Description
function (webs, lists)	Success callback function to be implemented, which receives two Arrays (webs, lists) containing JSON Objects.

webs Array

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

webs.web Object structure

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

lists Array

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

lists.list Object structure

Property	Type	Description
title	String	Title of the list.
url	String	URL to the list.
listName	String	GUID of the list.
Template	Number	Numeric value from SharePoint, see SPListTemplateType.
icon	String	Local image path for the icon.

onError callback function

Parameter	Description
function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

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
});
```

List.getItems Parameters

Parameter	Type	Description
url	String	URL to the SharePoint site.
viewName	String	Name of the SharePoint list view.

fields Array

Property	Type	Description
Fields	Array	Field values to be retrieved from each item.

fields Array Structure example

```
['ows_Title','ows_Created']
```

onSuccess callback function

Parameter	Description
function (items)	Success callback function to be implemented, which receives two Arrays (web, lists) containing JSON Objects.

items Array

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

Items.item Object structure

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

onError callback function

Parameter	Description
function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

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

List.getItemsWithOptions

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){
      //TODO: Go through the items array and implement what to do with each item
}, function (errorResponse){
      //TODO: Implement how to handle errors
}, function (cancelResponse){
      //TODO: Implement how to handle a user's cancel
});
```

List.getItemsWithOptions 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.

options Object Structure

Property	Type	Description
listName	String	GUID of the SharePoint list (**not** the display name).
webUrl	String	URL to the SharePoint site containing the list.
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.

Notes:

1. To search inside subfolders you need to set the *scope* property from queryOptions to "Recursive All".
   Sample snipett :
   ```
   Options.queryOptions =
   '<QueryOptions>' +
         '<IncludeAttachmentUrls>TRUE</IncludeAttachmentUrls>' +
         '<ViewAttributes Scope="RecursiveAll" />' +
   '</QueryOptions>';
   ```
2. When a *where* clause is present, the *viewName* value is ignored by List.getItemsWithOptions and only the CAML query is used to filter content.

onSuccess callback function

Parameter	Description
function (items)	Success callback function to be implemented, which receives an Array with the items (JavaScript Objects that represent an item). Each property corresponds to each field specified in the options Object; all field values are returned with String type.

item Object Structure

Property	Type	Description
Title	String	The item's title.
Created	String	The item's creation date.
…	…	…

onError callback function

Parameter	Description
function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

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
});
```

List.Item.download Parameters

Parameter	Type	Description
url	String	URL to the SharePoint site.

onSuccess callback function

Parameter	Description
function (filePath)	Success callback function to be implemented, which receives a String with the document's file path.

filePath Array

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

onError callback function

Parameter	Description
Function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

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
});
```

User.getUser Parameters

Parameter	Type	Description
url	String	URL to the SharePoint site.

onSuccess callback function

Parameter	Description
function (userProfile)	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 properties in the table below.

***userProfile** Sample subset of retrieved user properties*

Property	Type	Description
String	String	File path to the downloaded document.
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.

onError callback function

Parameter	Description
Function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

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
});
```

User.getProfileByName Parameters

Parameter	Type	Description
accountName	String	User account name to search for.
url	String	URL to the SharePoint web.

onSuccess callback function

Parameter	Description
function (userProfileProperties)	Success callback function to be implemented, which received an Array with user profile properties (JavaScript Objects that contain three properties).

userProfileProperties Array

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

userProfileProperties.UserProfile Object Structure

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

onError callback function

Parameter	Description
Function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

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 SharePlus Mobile Workspace. 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
});
```

WebDashboard.Settings.save Parameters

Parameter	Type	Description
settings	JSON Object	Custom settings to be saved.
key	String	Sets the unique key for custom settings.

onSuccess callback function

Parameter	Description
function ()	Success callback function to be implemented.

onError callback function

Parameter	Description
Function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

Sample

For a sample with this API method, refer to the Persisting SharePlus Mobile Workspaces 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
});
```

WebDashboard.Settings.load Parameters

Parameter	Type	Description
key	String	Unique key for the custom settings to be loaded.

onSuccess callback function

Parameter	Description
function (settings)	Success callback function to be implemented, which receives a JSON Object with the configuration settings loaded.

onError callback function

Parameter	Description
Function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

Sample

For a sample with this API method, refer to the Persisting SharePlus Mobile Workspaces Settings bridge scenario.

WebDashboard.setSource

This method is used to set a new source for a SharePlus Mobile Workspace in the Application Home. The change is persistent even if SharePlus is shut down and will replace all previously configured workspaces 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
});
```

WebDashboard.setSource Parameters

Parameter	Type	Description
source	String	The source of the mobile workspace.
sourceType	Number	The source type of the mobile workspace to be loaded.

onSuccess callback function

Parameter	Description
function ()	Success callback function to be implemented.

onError callback function

Parameter	Description
Function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

Sample

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

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

WebDashboard.reload

This method is used to reload the SharePlus Mobile Workspace. If the source has changed, the new mobile workspace is downloaded.

Syntax

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

WebDashboard.reload Parameters

None

WebDashboard.navigate

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

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
});
```

WebDashboard.navigate Parameters

Parameter	Type	Description
source	String	The source of the mobile workspace.
sourceType	Number	The source type of the mobile workspace to be loaded.

onSuccess callback function

Parameter	Description
function ()	Success callback function to be implemented.

onError callback function

Parameter	Description
Function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

Sample

For a sample with this API method, refer to the Changing the source of a SharePlus Mobile Workspace 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
});
```

Utility.getLocalizedString Parameters

Parameter	Type	Description
Text	String	Text to be localized.

onSuccess callback function

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

localizedText String

Property	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);
```

Utility.showMessage 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 SharePlus APIs to your Mobile Workspace 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);
```

Utility.URL.open 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
});
```

Utility.URL.download Parameters

Parameter	Type	Description
url	String	URL where the source 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.

useCache Boolean values

Option	Result
YES	Searches the local storage for the resource, only downloading if needed.
NO	The resource is downloaded.

onSuccess callback function

Parameter	Description
function (filePath)	Success callback function to be implemented, which receives a String with the file path of the resource.

filePath String

Property	Type	Description
filePath	String	File path to the downloaded resource.

onError callback function

Parameter	Description
Function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description

error#DisplayValue	String	Gets information about the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

Utility.URL.isAvailable

This method is used to determine if a 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
});
```

Utility.URL.isAvailable Parameters

Parameter	Type	Description
url	String	URL to be checked for availability.

onSuccess callback function

Parameter	Description
function (available)	Success callback function to be implemented, which receives a Boolean value (True or False= depending on the URL availability.

available String

Property	Type	Description
Available	String	Notifies whether the URL is available or not.

onError callback function

Parameter	Description
Function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

onCancel callback function

Parameter	Description
function (result)	Cancel callback function to be implemented.

Sample

For a sample with this API method, refer to the **Integrating SharePlus APIs to your Mobile Workspace** procedure.

Configuration.setRemoteFileSource

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
});
```

Configuration.setRemoteFileSource Parameters

Parameter	Type	Description
configSource	String	URL to the remote Configuration File location.

onSuccess callback function

Parameter	Description
function ()	Success callback function to be implemented.

onError callback function

Parameter	Description
Function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

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 ();
```

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
});
```

Configuration.getFeature Parameters

Parameter	Type	Description
featureKey	String	The feature's Key.

onSuccess callback function

Parameter	Description
function (featureConfig)	Success callback function to be implemented, which receives a JavaScript Object with the complete configuration settings for the specified feature.

featureConfig.feature Object structure example

Property	Type	Description
enabled	Boolean	Depends on whether the feature is enabled or not.
Settings	JSON Object	Object containing all the settings as properties.

onError callback function

Parameter	Description
Function (eResponse)	Error callback function to be implemented, which receives the eResponse JavaScript Object.

eResponse Object Structure

Property	Type	Description
error#DisplayValue	String	Gets information about the error.

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
});
```

onSuccess callback function

Parameter	Description
function (variables)	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
});
```

onSuccess callback function

Parameter	Description
function (online)	Success callback function to be implemented, which receives a Boolean value (True or False) depending on the connection mode.

online Boolean

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
});
```

onSuccess callback function

Parameter	Description
function (connected)	Success callback function to be implemented, which receives a Boolean value (True or False) depending on the connection availability.

connected Boolean

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 SharePlus APIs to your Mobile Workspace procedure.

Connection.setOnLine

This method is used to switch SharePlus to the Online mode.

Syntax

```
SPlus.Connection.setOnLine ();
```

Connection.setOnline Parameters

None

Connection.setOffLine

This method is used to switch SharePlus to the Offline mode.

Syntax

```
SPlus.Connection.setOffLine ();
```

Connection.setOffLine Parameters

None

DataManagerContext

You can use an instance of this object type to force an existing API method to bypass SharePlus cache and get SharePoint content straight from the server.
Create a new DataManagerContext object, then set its **forceOnline** property to true, and finally send the new object as a parameter to the API method.

Syntax

```
var context = new DataManagerContext(); // Create a new DataManagerContext object
context.forceOnline = true; // Set the forceOnline property to true

// Specify the web's URL.
var webUrl = " http://spdemo.infragistics.com/demo";

// Invoke Web.getWebsAndLists API method.
SPlus.Web.getWebsAndLists(webUrl, function(webs, lists) {
     // Get all webs and lists and implement what to do with them
}, function(response) {
     // Implement how to handle errors
}, function(response) {
     // Implement how to handle a user's cancel
}, context); // Send the DataManagerContext to the API method
```

API Methods which accept DataManagerContext as parameter

Category	API Property	Parameters accepted by the Method
Search	getScopes	url, onSuccess, onError, onCancel, **context**
	query	url, text, onSuccess, onError, onCancel, **context**
	queryWithOptions	url, options, onSuccess, onError, onCancel, **context**
Web	getWebsandLists	url, onSuccess, onError, OnCancel, **context**
List	getItems	url, viewName, fields, onSuccess, onError, onCancel, **context**
	getItemswithOptions	url, options, onSuccess, onError, onCancel, **context**
	updateViewCollectionMetadata	url, onSuccess, onError, onCancel, **context**
	listItem.download	url, onSuccess, onError, onCancel, **context**
User	getUser	url, onSuccess, onError, onCancel, **context**
	getProfileByName	accountName, url, onSuccess, onError, onCancel, **context**

Appendix 2: SharePlus Links Reference

CATEGORY	ACTIONS (?)	PARAMETERS (&)	SAMPLES
SharePoint Navigation	**view** – Navigate to SharePoint (sites, lists, libraries, documents, or items)	OPTIONAL - *mode* – InWeb, InSafari, Native (default) - *<list/library parameter>* - Add list/libraries parameters to make navigation more specific	**SITES** splus://<portal>/site?**action=view**&*mode=InSafari* splus://dynamicWebServer/page.aspx?*color=red*&**action=view**&*mode=InSafari* **LISTS AND LIBRARIES** splus://<portal>/site/calendar?**action=view** splus://<portal>/site/calendar?**action=view**&*viewName=All%20Events*
	viewdocument – Open a SharePoint document	OPTIONAL - *mode* – InWeb, InSafari, Native (default) - *filter* for ReportPlus (must be existing filter within ReportPlus)	**DOCUMENTS** splus://portal/site/library/document.docx?**action=viewdocument**&*mode=InSafari* splus://<portal>/site/multimedia/Away.mp3**?action=viewdocument** **REPORTPLUS DASHBOARD** splus://portal/site/library/reportPlusDashboard.rplus?*country=USA*&**action=viewdocument**
	viewitem – Open the properties of a document	OPTIONAL - *mode* – InWeb, InSafari, Native (default)	splus://<portal>/site/Tasks/**ID=4** splus://<portal>/site/multimedia/Away.mp3**?action=viewitem** splus://<portal>/site/list/item.docx?**action=viewitem**
SharePoint Search	**search** – Search for a document on a site	-	splus://<portal>/site?**action=search**
	query – Custom search on a SharePoint List	OPTIONAL - *filters (*SEE TABLE 1 for filter formatting)	splus://<portal>/site/multimedia?**action=query**&*filter=ows_Title:contains(Text)&filtertitle=Text%20Filter&includesubfolders=true*
SharePoint Edition	**additem** – Add new item to SharePoint	-	plus://<portal>/site/multimedia/Away.mp3?**action=additem**&contenttype=audio
	edititem – Edit existing item in SharePoint	OPTIONAL - *filters* (SEE TABLE 2 for operators)	splus://<portal>/site/multimedia/Away.mp3?**action=edititem** splus://<portal>/site/multimedia/Home.mp3?**action=additem**&contenttype=audio&*ows_comments=Added&with&20URL&20schemes&ows_title=New%20title*
SharePlus actions	**localfiles** - Go to Local Files	OPTIONAL - *folder* – go to specific folder	splus://**?action=localfiles** splus://**?action=localfiles**&*folder=logs*
	favorites - Go to Favorites	-	splus://**?action=favorites**
	help - Go to Help	-	splus://**?action=help**
	settings -Go to Settings	-	splus://?**action=settings**
	home -Go to Home	-	splus://?**action=home**
SharePlus configuration	**configurationURL** – remote config update	MANDATORY - ***URL*** – link to new config File OPTIONAL - *user-agent, timeout*	splus://?**action=configurationURL**&***url=https%3A%2F%2Fportal%2FConfigurationFiles%2FCustomConfiguration.plist***
	reloadconfig – reload remote config	-	splus://?**action=reloadconfig**
	webdashboard – modify the source of a mobile workspace in the Application Home	MANDATORY - ***source*** – URL to resource or local path to web resource. OPTIONAL - *title*	splus://?**action=webdashboard**&***source=http%3A%2F%2Fportal%2Fsite%2FSiteAssets%2FCustomMobileWorkspace.web.zip***&*title=MyMobileWorkspace*

Table 1: Query Operators

Operators	Description
equals	The field value must be exactly the same
notequals	The field value must be different
greater	The field value must be greater
greaterorequal	The field value must be greater or equal
lower	The field value must be lower
lowerorequal	The field value must be lower or equal
isnull	The field value must not be specified (isnull())
isnotnull	The field value must not be specified (isnotnull())
beginswith	The field value must start with the value specified
contains	The field value must contain the value specified

Table 2: Editing Formatting

Field type	Format	Examples
Single/Multiple lines of text, Numbers, Currency, Hyperlink	These values need no format.	**Text:** ows_TextField=Text%20Value **Number:** ows_NumberField=15
Choice	Values must be separated by "**;#**" characters (%3B%23 when escaped).	**Single:** ows_ChoiceField=%3B%23Value1%3B%23 (Non-escaped: ows_ChoiceField=;#Value1;#) **Multi:** ows_ChoiceField=%3B%23Value1%3B%23Value2%3B%23 (Non-esc: ows_ChoiceField;#Value1;#Value2;#)
Lookup, Person or Group	ItemID;#Name	ows_LookupField=103%3B%23Test (Non-escaped: ows_LookupField =103;#Test)
Date and Time	yyyy-MM-dd'T'HH:mm:ss'Z'	ows_DateField=2012-12-27T16%3A15%3A31Z (Non-esc: ows_DateField =2012-12-27T16:15:31Z)
Yes/No	TRUE or FALSE	ows_YesNoField=TRUE

Appendix 3: Document Changelog

Version	Section	Description
5.0.3	-	Updated version to SharePlus 4.9. No new changes.
5.0.2	Complete API Scenarios	Fixed two uppercase typos (item.attribute and function) in step 3 of the *Integrating a custom form to a SharePoint list in SharePlus* scenario
	API Reference (AddItem)	Added information about how to work with lookup fields.
5.0.1	Introducing SharePlus SDK	Updated screenshots with the new sidebar.
	Getting Started	
	SharePlus Extension Points	
	Getting Started with the API	
	Complete API Scenarios	
5.0	-	Guide restructured to better present new features.
	Getting Started	Added new information on the custom form functionality.
	SharePlus Extension Points	
	Deployment & Configuration	
	Complete API Scenarios	
	Partial API Scenarios	
	API Reference	Updated API Reference with new methods for custom forms.
4.3	Appendix 2	Corrected actions for the SharePoint Navigation section.
4.2	Using SharePlus Links	Updated information with *splus-action* and *home* parameters.
	Appendix 3	Updated information with *home* parameter.
4.1	Getting Started	Added brief information for HTTP Caching.
	Getting Started with the API	Added new SharePlusBridge.js accepted method (DataManagerContext).
4.0	-	Guide renamed to SharePlus Developer's Guide. The term "Launchpad" is outdated, we now use "SharePlus Mobile Workspace".
3.1	All sections	Updated all screenshots to reflect new icons.
3.0.2	Using SharePlus Links	SharePlus Link Types moved from Appendix 3 to this section.
	Appendix 3	New Quick Reference table added with existing actions and parameters.
3.0.1	Getting Started with the API	Added clarifications for the SharePlus bridge and the SPlus prefix
3.0	Appendix 3	New appendix added to include SharePlus reference information
	Using SharePlus Links	New section added to include instructions on how to use SharePlus links
2.0.1	Getting Started	Minor fix in the Note about creating launchpads with more than HTML code.
2.0	API Reference	Added new API methods under the Search, Taxonomy and Connection categories.

1.2	-	Guide renamed to Launchpads Developer Guide.
1.2	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).
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).