Contents

[SharePoint 2013: Store and retrieve SharePoint list items on a Windows Phone 2](#_Toc329691200)

[Description of the sample 2](#_Toc329691201)

[Prerequisites 3](#_Toc329691202)

[Key components of the sample 3](#_Toc329691203)

[Configure the sample 3](#_Toc329691204)

[Run and test the sample 4](#_Toc329691205)

[Troubleshooting 4](#_Toc329691206)

[Change log 5](#_Toc329691207)

[Related content 5](#_Toc329691208)

# SharePoint 2013: Store and retrieve SharePoint list items on a Windows Phone

This documentation is preliminary and is subject to change.

Summary:  Implement a mechanism for offline editing of SharePoint list items in a Windows Phone app.

One of the most important considerations in the development of Windows Phone apps is the management of state information, both for the overall application and for individual pages or data items within the application. Developers of Windows Phone apps must assume that users of their apps can lose connectivity to network resources (such as SharePoint lists). The development infrastructure for Windows Phone apps provides mechanisms for handling state information at various stages in the life cycle of an app. In a Windows Phone app that gets data from a SharePoint list, the data used on the phone from session to session can be retrieved from SharePoint Server if the server is available. But continuous connectivity to a SharePoint Server may not be available for a Windows Phone device because of variations in service coverage and other factors. To provide users of your app with access to data when connectivity with SharePoint Server is lost, or simply to save data to persistent storage between sessions of the app regardless of server availability, you can take advantage of the Closing and Launching events of the PhoneApplicationService class.

## Description of the sample

After implementing handlers for the Closing and Launching events in your app, SharePoint list data that was retrieved from the server when connectivity was available can be displayed in your app even if connectivity to the server is lost in a subsequent session of the app, because the list items are retrieved from local persistent storage on the phone. However, list items that are displayed offline can't be edited and saved to the server unless connectivity is restored. In the following procedure, you add a mechanism to your app to store edited versions of list items locally when connectivity is unavailable. When connectivity to the server is again available, you can retrieve these edited list items and save your changes to the server.

A specific instance of the EditItemViewModel class represents a SharePoint list item that is being edited on the phone. A list item that has been edited can be considered metaphorically as a "draft item" before changes are saved to the server. In the code in this class, the AddDraftItem method adds a specific instance of the EditItemViewModel class (that is, a draft item) as a value to a Dictionary object, associating the EditItemViewModel in the Dictionary with a key based on the identifier for the given list item.

Note



An identifier is assigned by SharePoint Server to each item in a list. In a project based on the Windows Phone SharePoint List Application template, that identifier is stored in the **ID** property of the ViewModel class, such as **EditItemViewModel** or **DisplayItemViewModel**, that represents the list item.

The RemoveDraftItem method removes an EditItemViewModel from the Dictionary object based on a specified identifier. Both of these methods use the GetDraftItemCollection method to retrieve the Dictionary object containing the EditItemViewModel objects from isolated storage and both methods use the SaveDrafts method to save the modified Dictionary object (with a draft item either added to it or removed from it) back to isolated storage. The GetDraftItemCollection method first determines whether a "Drafts" Dictionary object has been saved to isolated storage. If so, the method returns that Dictionary object; otherwise, the method initializes and returns a new Dictionary object. The Drafts property of the class provides access to the Dictionary of draft items by returning a list (that is, an object based on the List<T> generic) of draft items as EditItemViewModel objects. The GetDraftItemById method returns a given draft item from the Dictionary object based on a specified identifier value.

## Prerequisites

This sample requires the following:

* Visual Studio 2010 Express with the new SharePoint templates
* An installation of SharePoint 2013 Preview, with administrative privileges

## Key components of the sample

The sample contains the following:

* SPListAppLocalStorage project, which contains the App.xaml.cs file and the CreateProductOrdersList.ps1 Windows PowerShell script.
* App.xaml.cs   This file is autogenerated by the Windows Phone SharePoint List Application template .The App.xaml file represents the overall Windows app. The associated code-behind file, App.xaml.cs, includes procedural code to handle life-cycle events for the app. App.xaml.cs contains references to SharePoint Server and the list title.
* CreateProductOrdersList.ps1   You can run this Windows PowerShell script from the SharePoint Management Shell to create the SharePoint list on which this project is based.

## Configure the sample

This sample assumes that you are working in a Windows Phone app project that was created from the Windows Phone SharePoint List Application template and that your app is based on a Product Orders list created from the Custom List template on the server and containing the columns and field types shown in Table 1.

Table 1. Sample Product Orders list

| Column | Type | Required |
| --- | --- | --- |
| Product (i.e., Title) | Single line of text (Text) | Yes |
| Description | Single line of text (Text) | No |
| Quantity | Number | Yes |
| Order Date | Date and Time (DateTime) | No |
| Fulfillment Date | Date and Time (DateTime) | No |
| Contact Number | Single line of text (Text) | No |

Follow these steps to configure the sample.

1. Update the value of TargetSiteUrl in the App.xaml.cs file of the SPListAppLocalStorage solution with the URL of the home page of your SharePoint website.
2. Update the value of the <List Title> element in the App.xaml.cs file of the SPListAppLocalStorage solution with the title of the target SharePoint list.

## Run and test the sample

* Choose the F5 key to build and run the app.

## Troubleshooting

The following table lists common configuration and environment errors that prevent the sample from running or deploying properly and how to solve them.

| Problem | Solution |
| --- | --- |
| While running the SharePoint List wizard from Visual Studio 2010 Express, an error may occur if developer does not have sufficient privilege on SharePoint site. | Give sufficient privilege to the user account with which developer is running the wizard. |
| Form-based authentication error. | Form-based authentication is not enabled by default. To enable basic form-based authentication for the web application, follow these steps.   * Navigate to Central Administration and ensure you have administrator rights on the server. * Under Application Management, choose Manage Web Applications. * Choose your web application (on which you have your SharePoint site, which you are accessing from your mobile app). * From the ribbon, choose Authentication Providers. * In the Authentication Provider dialog box, choose Default to edit the authentication. * In the Edit Authentication Model window under Claims Authentication Types, choose Basic Authentication. |

## Change log

| Version | Date |
| --- | --- |
| First version | July 16, 2012 |

## Related content

* **How to: Store and retrieve SharePoint list items on a Windows Phone**
* **How to: Create a Windows Phone SharePoint 2013 list app**
* **Overview of Windows Phone SharePoint 2013 application templates in Visual Studio**