How To... Call an External JSON Services Using the OData SDK (Windows)

Applicable Releases:

SAP Mobile Platform 3.0

Version 1.0

June 2014

|  |  |
| --- | --- |
| © Copyright 2014 SAP AG. All rights reserved.   1. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice. 2. Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors. 3. Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation. 4. IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation. 5. Linux is the registered trademark of Linus Torvalds in the U.S. and other countries. 6. Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries. 7. Oracle is a registered trademark of Oracle Corporation. 8. UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group. 9. Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc. 10. HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology. 11. Java is a registered trademark of Sun Microsystems, Inc. 12. JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape. 13. SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP BusinessObjects Explorer, StreamWork, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.   Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects Software Ltd. Business Objects is an SAP company.   1. Sybase and Adaptive Server, iAnywhere, Sybase 365, SQL Anywhere, and other Sybase products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Sybase, Inc. Sybase is an SAP company. | 1. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary. 2. The information in this document is proprietary to SAP. No part of this document may be reproduced, copied, or transmitted in any form or for any purpose without the express prior written permission of SAP AG. 3. This document is a preliminary version and not subject to your license agreement or any other agreement with SAP. This document contains only intended strategies, developments, and functionalities of the SAP® product and is not intended to be binding upon SAP to any particular course of business, product strategy, and/or development. Please note that this document is subject to change and may be changed by SAP at any time without notice. 4. SAP assumes no responsibility for errors or omissions in this document. SAP does not warrant the accuracy or completeness of the information, text, graphics, links, or other items contained within this material. This document is provided without a warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. 5. SAP shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials. This limitation shall not apply in cases of intent or gross negligence. 6. The statutory liability for personal injury and defective products is not affected. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third-party Web pages nor provide any warranty whatsoever relating to third-party Web pages.   SAP “How-to” Guides are intended to simplify the product implement­tation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using SAP NetWeaver. Should you wish to receive additional information, clarification or support, please refer to SAP Consulting.  Any software coding and/or code lines / strings (“Code”) included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.  Disclaimer  Some components of this product are based on Java™. Any code change in these components may cause unpredictable and severe malfunctions and is therefore expressively prohibited, as is any decompilation of these components.  Any Java™ Source Code delivered with this product is only to be used by SAP’s Support Services and may not be modified or altered in any way. |

Document History

|  |  |
| --- | --- |
| Document Version | Description |
| 1.00 | First official release of this guide |

Typographic Conventions

|  |  |
| --- | --- |
| Type Style | Description |
| Example Text | Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options.  Cross-references to other documentation |
| **Example text** | Emphasized words or phrases in body text, graphic titles, and table titles |
| Example text | File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools. |
| Example text | User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation. |
| <Example text> | Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system. |
| EXAMPLE TEXT | Keys on the keyboard, for example, F2 or ENTER. |

Icons

|  |  |
| --- | --- |
| Icon | Description |
|  | Caution |
|  | Note or Important |
|  | Example |
|  | Recommendation or Tip |

Table of Contents

[1. Business Scenario 1](#_Toc393401903)

[2. Background Information 1](#_Toc393401904)

[3. Prerequisites 1](#_Toc393401905)

[4. Step-by-Step Procedure 2](#_Toc393401906)

[4.1 Windows Project 2](#_Toc393401907)

[4.2 Retrieving data from SMP Server 3](#_Toc393401908)

[4.2.1 Steps involved in calling a whitelisted connection 3](#_Toc393401909)

[4.2.2 JSON Service request Process Flow 5](#_Toc393401910)

[4.2.3 Screen flow of the application 6](#_Toc393401911)

[4.2.4 GET JSON Service 7](#_Toc393401912)

[4.3 Running the application 9](#_Toc393401913)

[5. Appendix 11](#_Toc393401914)

[5.1 NuGet Package Manager 11](#_Toc393401915)

[5.1.1 Adding Windows SMP SDK package in Visual Studio 11](#_Toc393401916)

[5.1.2 Adding Windows SMP SDK references to the project 12](#_Toc393401917)

[5.2 Create whitelisted connections 14](#_Toc393401918)

[5.3 Set up proxy server in SMP Server 14](#_Toc393401919)

# Business Scenario

Travel Agency X would like to build an online mobile application for its customers, so they can book their flights anywhere, anytime from their devices. The **SAP Mobile Platform** provides a means for them to securely and efficiently access backend flight data exposed by their SAP NetWeaver Gateway system via OData-based REST services.

In addition to accessing backend flight data, the application also requires access to an external JSON non-OData service. The weather conditions of the airports are available as an external JSON service.

# Background Information

The goal of this exercise is not to show how to create a project from scratch and dissect every line of code. Instead, it shows the key pieces of code and information, along with a starter project template, so that developers understand how to leverage the OData SDK to on-board users in their own apps.

This exercise focuses on showing how to call and parse an external REST based service that returns JSON formatted content via the SAP Mobile Platform using the OData SDK. It also provides details on how to configure the external service as a whitelisted endpoint.

# Prerequisites

This exercise has the following prerequisites:

* Windows 8.1 operating system (for Windows Store applications)
* Visual Studio 2013 with Update 2 (for Windows Store and Desktop applications)
* .NET 4.5 (for Windows Desktop applications)
* To get the most out of this exercise, experience with Windows programming is recommended.

# Step-by-Step Procedure

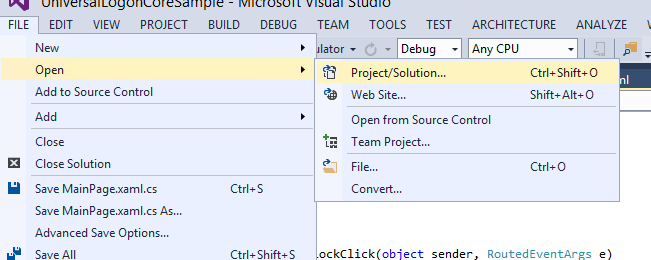
In the previous exercise, we submitted HTTP GET requests to the SMP Server to retrieve data from the CarrierCollection and the FlightCollection. We then parsed the HTTP response received from the SMP Server and displayed the results to the user.

Now we want to extend our app and connect to an external weather service that returns data on weather conditions for a chosen destination airport. This exercise provides details on how to add a whitelisted connection. This exercise also shows how to set up the SMP Server to use a proxy server to access the internet.

The following sections provide a detailed step-by-step procedure on how to connect to an external JSON service using the SAP Mobile Platform. Each exercise builds upon the previous exercise, so it is recommended that you complete each exercise before moving to the next.

## Windows Project

1. ...
   1. Open Visual Studio 2013 with Update 2 and open the solution RKT\_JSONService.sln



* 1. Windows SMP SDK uses Microsoft OData Parser libraries to parse the OData responses. So in addition to adding Windows SMP SDK libraries as references, you should also add Microsoft OData Parser libraries as references.
  2. Windows SMP SDK libraries are packaged as NuGet packages. See appendix on how to add the libraries as reference.
  3. See appendix on how to create white listed connections in the SMP Server.
  4. See appendix on how to set up SMP Server to use a proxy server to access the internet.

## Retrieving data from SMP Server

### Steps involved in calling a whitelisted connection

During the on-boarding process, the HTTP response from the SMP Server contains a wealth of information including the application connection id, the endpoint URL etc. However, the SMP Server does not return any information about the whitelisted connections. This is because the application can be edited any time to add a new whitelisted connection. Since registration is a onetime process, the device may not know about the whitelisted connection, even if the SMP Server sent it as part of the registration process. Because of this limitation, the developer should know beforehand the details of the whitelisted connection including name and the type of data that will be returned.

This particular whitelisted connection that we are using in this sample exercise is not an OData feed. Since we know this information before hand, we cannot use the ODataStore library to make the HTTP GET requests. The ODataStore is to be used only for OData feeds. However, we can use the SAP.Net.Http.HttpClient library to make the HTTP GET requests for any other type of connections.

To create an instance of the SAP.Net.Http.HttpClient, the developer can use either one of 3 constructors. For the most part, the developer should use the 3rd constructor that takes the HttpMessageHandler and a bool value. Using the HttpMessageHandler, you can pass in the credentials to connect to the SMP Server. The 2nd parameter allows the handler to be disposed after being used.

|  |
| --- |
| public HttpClient();  public HttpClient(HttpMessageHandler handler);  public HttpClient(HttpMessageHandler handler, bool disposeHandler); // Use this  // Create an HttpClient instance  var client = new SAP.Net.Http.HttpClient(  new System.Net.Http.HttpClientHandler()  {  Credentials = new System.Net.NetworkCredential( Globals.LogonCore.LogonContext.RegistrationContext.BackendUserName, Globals.LogonCore.LogonContext.RegistrationContext.BackendPassword),  },  true); // will be disposed by the store!  client.DefaultRequestHeaders.TryAddWithoutValidation("X-SMP-APPCID", connectionId);  client.DefaultRequestHeaders.TryAddWithoutValidation("X-SUP-APPCID", connectionId);  client.ShouldHandleXcsrfToken = true; |

The application connection id is added as a header to the client variable. The GetAsync method is then called asynchronously passing in the url as a parameter. The response from the SMP Server is processed using the native Windows JSON library.

|  |
| --- |
| // Send a request asynchronously continue when complete  HttpResponseMessage response = await client.GetAsync(url);    // Check that response was successful or throw exception  response.EnsureSuccessStatusCode();  // Read response asynchronously  var content = await response.Content.ReadAsStringAsync();  // Parse the content into a JsonObject  var jsonContent = JsonObject.Parse(content); |

### JSON Service request Process Flow

Create an instance of SAP.Net.Http.HttpClient

1. At design time or runtime – Set the DataContext of the xaml page to the instance of the class
2. Bind properties to UI controls

Create an instance of a class with values from the parsed response

Use native libraries to parse response from server

GetAsync is called passing url as parameter

Set the application connection id as headers

### Screen flow of the application

This sample application submits a GET request to the SMP Server to retrieve data from an external JSON Service. The weather information about the destination airport is retrieved from the JSON Service.

FlightDetails Page (FlightDetails.xaml) for a selected Flight

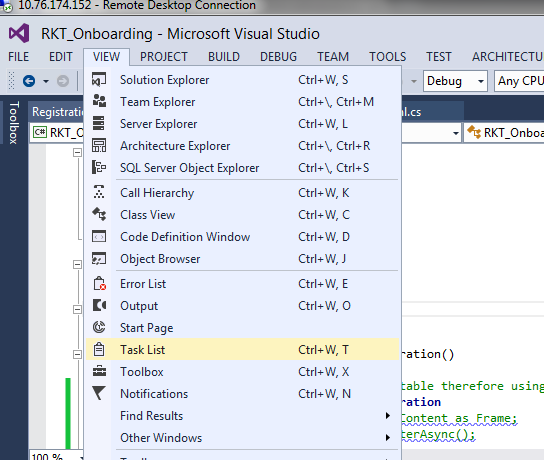
FlightCollection Page (Flights.xaml) for a selected Carrier

CarrierCollection Page

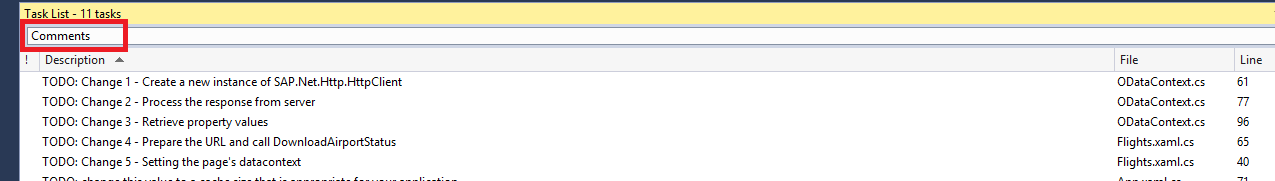
(Airlines.xaml)

### GET JSON Service

1. ...
   1. In Visual Studio 2013 + Update 2, click on View -> Task List



* 1. Sort the tasks alphabetically, to view all the tasks associated for this exercise. They should all begin with **// TODO: Change x.** There are 5 tasks for this exercise.



* 1. Open ODataContext.cs in the Shared project.
  2. Go to the DownloadAirportStatus method. In between the BEGIN and END //TODO Change1: markers enter the following code:

|  |
| --- |
| // Create an HttpClient instance  var client = new SAP.Net.Http.HttpClient(  new System.Net.Http.HttpClientHandler()  {  Credentials = new System.Net.NetworkCredential( Globals.LogonCore.LogonContext.RegistrationContext.BackendUserName, Globals.LogonCore.LogonContext.RegistrationContext.BackendPassword),  },  true); // will be disposed by the store!  client.DefaultRequestHeaders.TryAddWithoutValidation("X-SMP-APPCID", connectionId);  client.DefaultRequestHeaders.TryAddWithoutValidation("X-SUP-APPCID", connectionId);  client.ShouldHandleXcsrfToken = true; |

We create an instance of the SAP.Net.Http.HttpClient to submit the HTTP GET request. The HttpClientHandler is used to pass in the credentials.

* 1. In between the BEGIN and END //TODO Change2: markers enter the following code:

|  |
| --- |
| // Send a request asynchronously continue when complete  HttpResponseMessage response = await client.GetAsync(url);    // Check that response was successful or throw exception  response.EnsureSuccessStatusCode();  // Read response asynchronously as JsonValue  var content = await response.Content.ReadAsStringAsync();  // Parse the content into a JsonObject  var jsonContent = JsonObject.Parse(content); |

The HTTP request is submitted asynchronously using the GetAsync method. The response from the SMP Server is parsed using native Windows libraries. Note that the developer needs to know beforehand that the response is a JSON object.

* 1. In between the BEGIN and END //TODO Change3: markers enter the following code:

|  |
| --- |
| var airport = jsonContent.GetNamedValue("name").GetString();  jsonContent = jsonContent.GetNamedObject("weather");  var weather = jsonContent.GetNamedValue("weather").GetString();  var temp = jsonContent.GetNamedValue("temp").GetString();  var wind = jsonContent.GetNamedValue("wind").GetString();  SharedContext.JsonContext = new JsonContext(airlineId, flightNumber, flightDate, price, currency, airport, weather, temp, wind); |

The properties from the JSON object are used to create an instance of the JsonContext class which is assigned to a static JsonContext class. This static class is later used to bind to the UI controls.

* 1. Open Flights.xaml.cs in the Shared project.
  2. Go to the BtnAirportStatusClick event handler. In between the BEGIN and END //TODO Change4: markers enter the following code:

|  |
| --- |
| var url = "http://" + Globals.LogonCore.LogonContext.RegistrationContext.ServerHost + ":" + Globals.LogonCore.LogonContext.RegistrationContext.ServerPort + "/AirportStatus/" + (string)entity.Properties["flightDetails/airportTo"].Value + "?format=json";  try  {  SharedContext.Context.RingVisible = Visibility.Visible;  await SharedContext.Context.DownloadAirportStatus(url);  } |

The url is constructed here and then DownloadAirportStatus is called passing the constructed url. Note that the developer needs to know the name of the whitelisted connection beforehand.

* 1. Open FlightDetails.xaml.cs in the Shared project.
  2. Go to the constructor of the FlightDetails class. In between the BEGIN and END //TODO Change5: markers enter the following code:

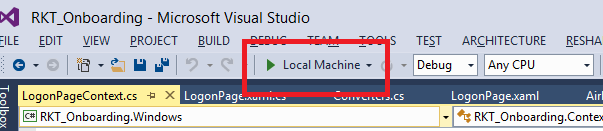
|  |
| --- |
| this.DataContext = SharedContext.JsonContext; |

The DataContext of the FlightDetails page is set to SharedContext.JsonContext. The TextBlocks in the FlightDetails.xaml page are bound to properties of the JsonContext class. To verify that the TextBlocks are bound to the JsonContext static class, open FlightDetails.xaml file. In the StackPanel, you should see the following DataContext="{Binding}" attribute…

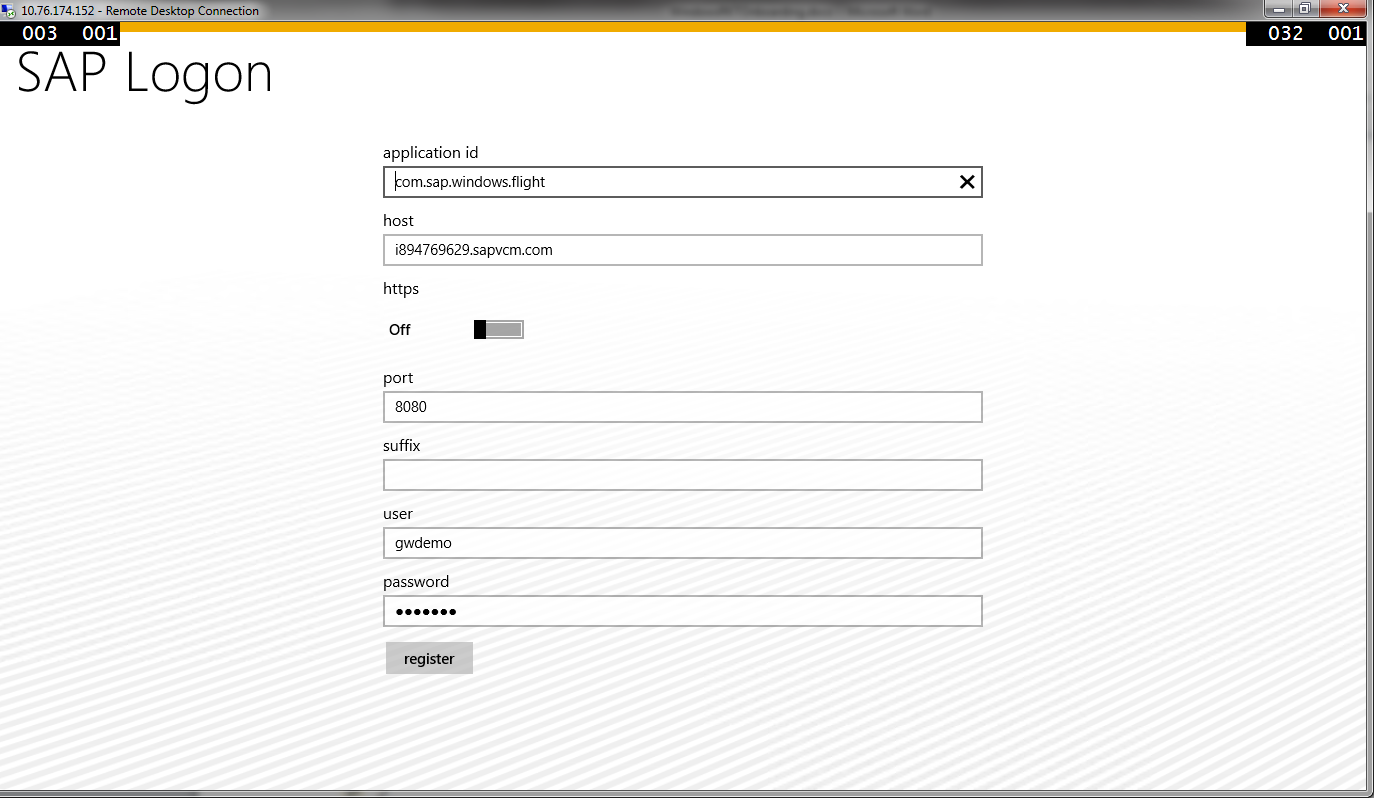
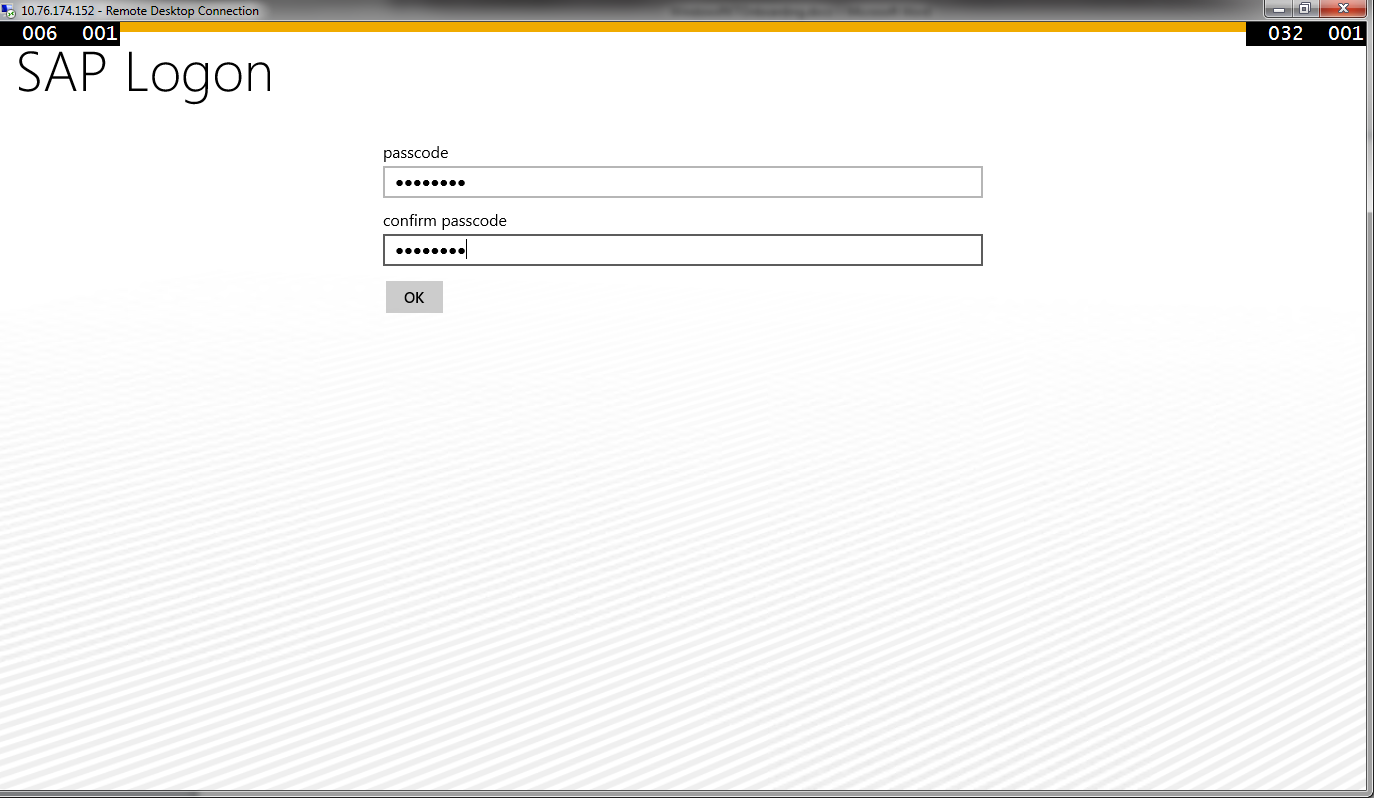
|  |
| --- |
| <StackPanel DataContext="{Binding}">  <StackPanel Orientation="Horizontal">  <TextBlock Width="200" FontWeight="Bold" Text="Aiport" />  <TextBlock Text="{Binding DestinationAirport}"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <TextBlock Width="200" FontWeight="Bold" Text="Weather" />  <TextBlock Text="{Binding Weather}"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <TextBlock Width="200" FontWeight="Bold" Text="Temperature" />  <TextBlock Text="{Binding Temperature}"/>  </StackPanel>  </StackPanel> |

## Running the application

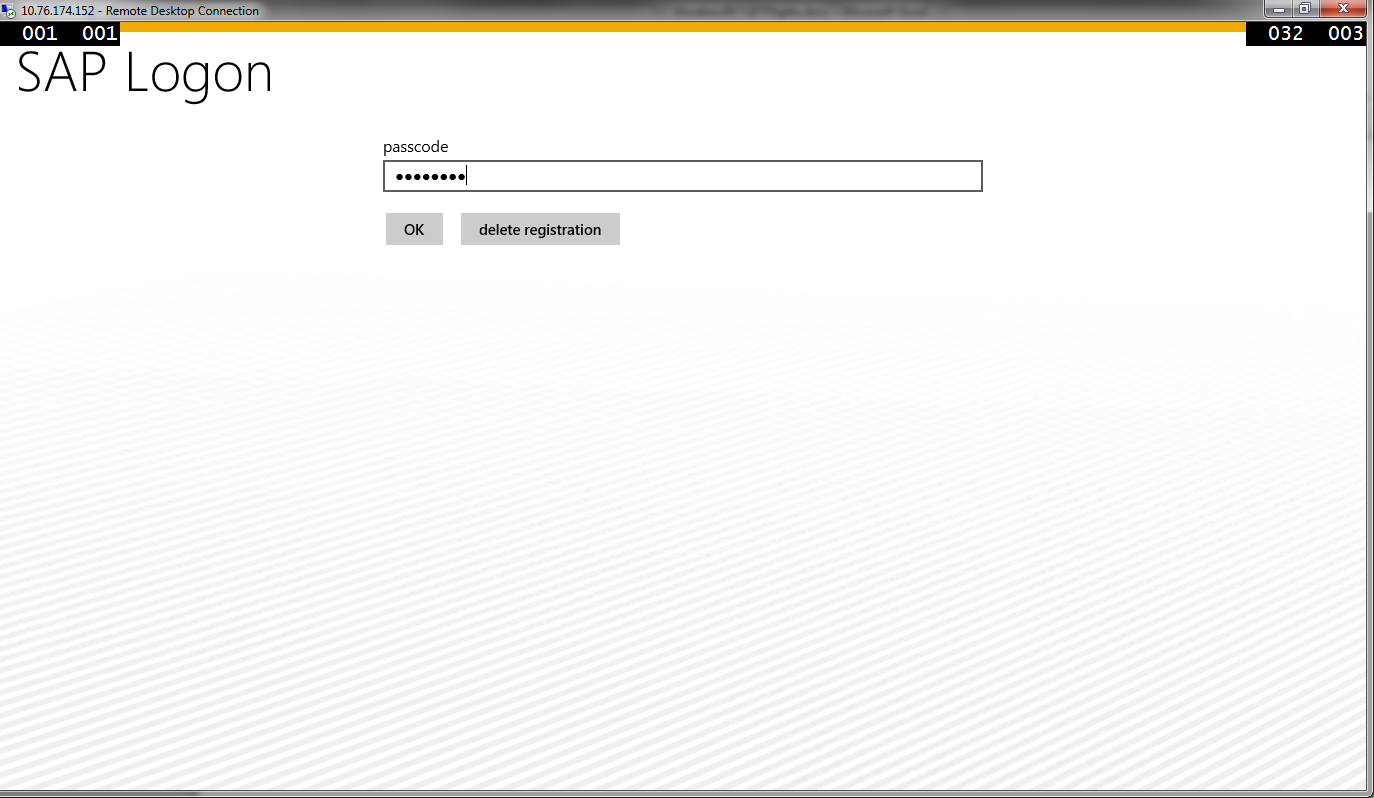
1. Run the application by clicking Run from within Visual Studio 2013 Update 2.



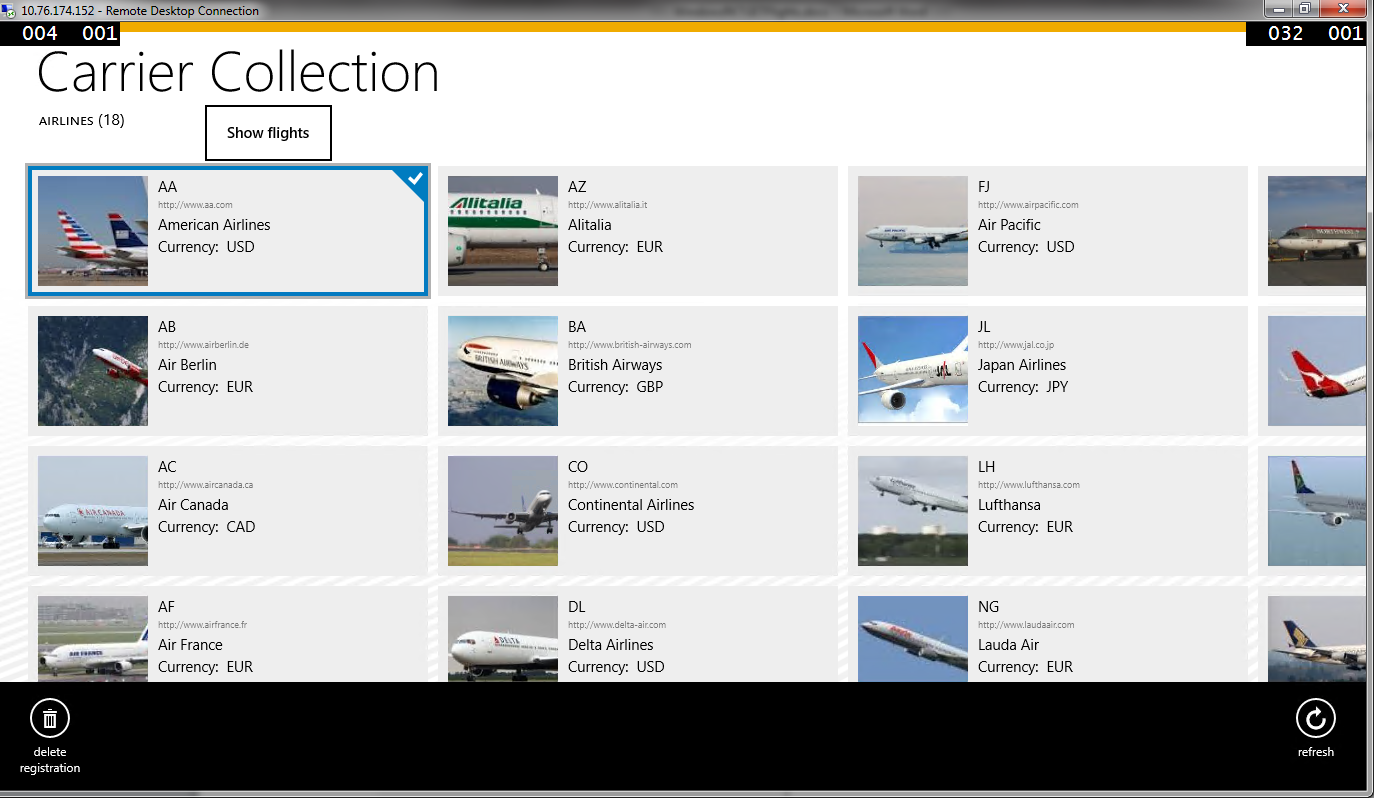
1. If you are not registered, click register. Enter passcode for data vault and click Ok.

If you are already registered, enter the passcode to unlock the data vault.



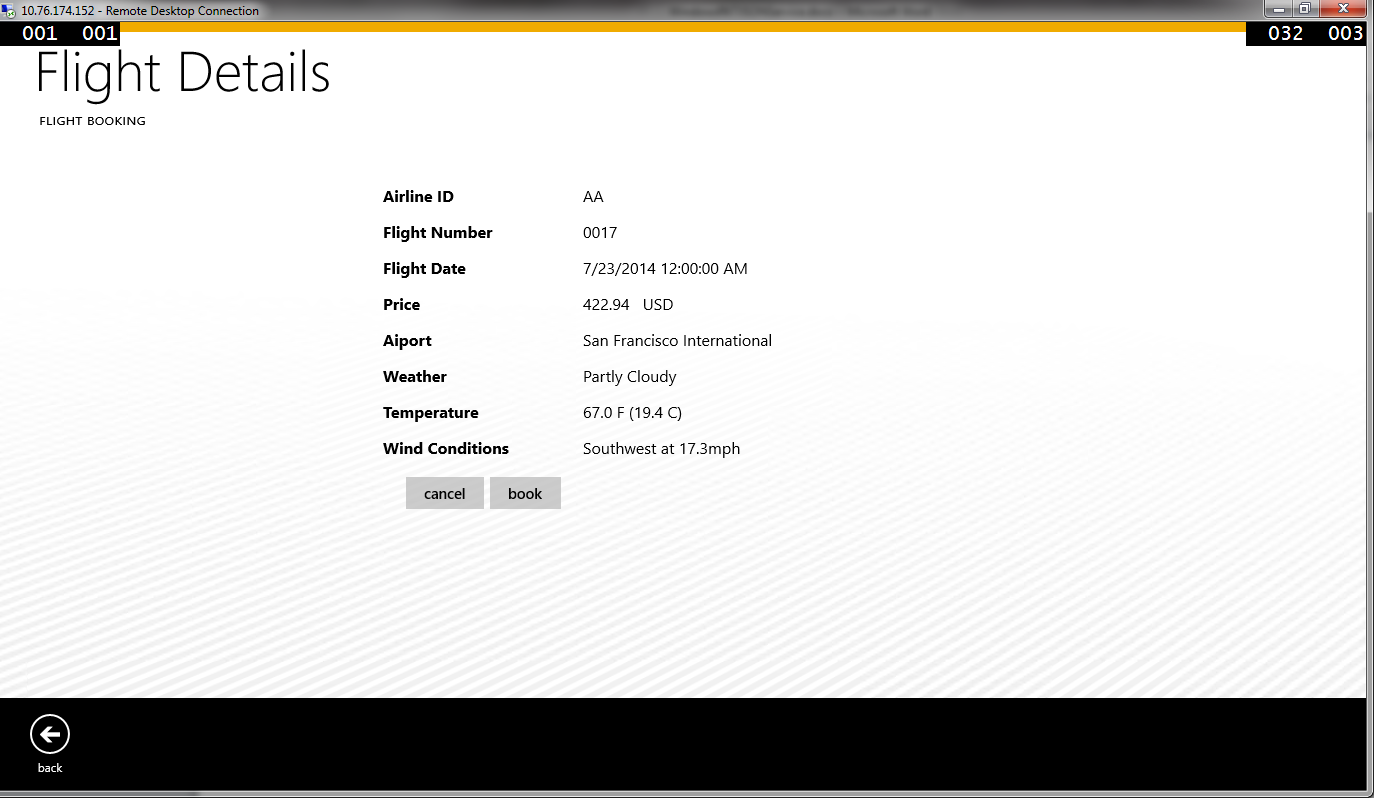
1. The application should now open up. Right click on any Carrier and click on Show flights.



1. You should now be taken to the Flights page. Right click on a future flight and click on Airport Status.



1. You should now be taken to the FlightDetails page.



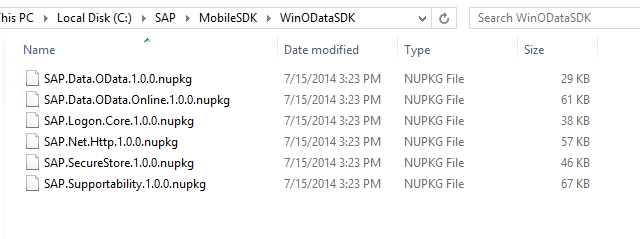
# Appendix

## NuGet Package Manager

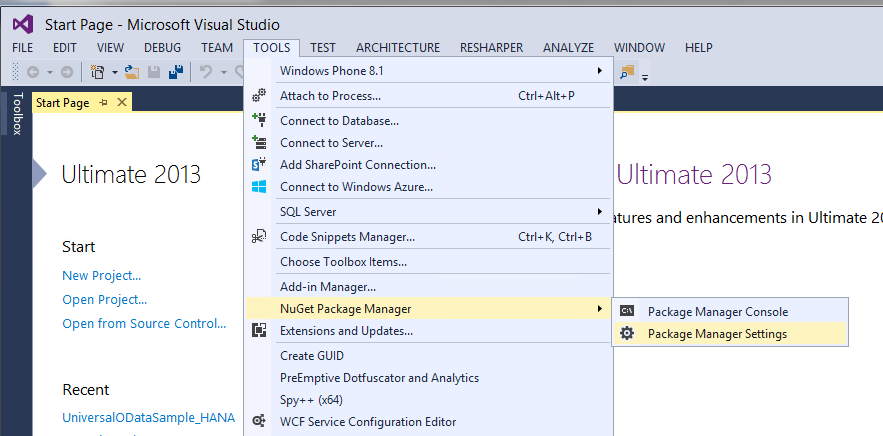
NuGet is the package manager for the Microsoft development platform including .NET. The NuGet client tools provide the ability to produce and consume packages. Starting with Visual Studio 2012, NuGet is included in every edition (except Team Foundation Server) by default. Updates to NuGet can be found through the Extension Manager.

### Adding Windows SMP SDK package in Visual Studio

1. Find the location of the Windows SMP SDK files in your local development machine (default location is C:\SAP\MobileSDK3\NativeSDK\ODataFramework\Windows). (The .nupkg file contains libraries for both Windows Store and Windows desktop)



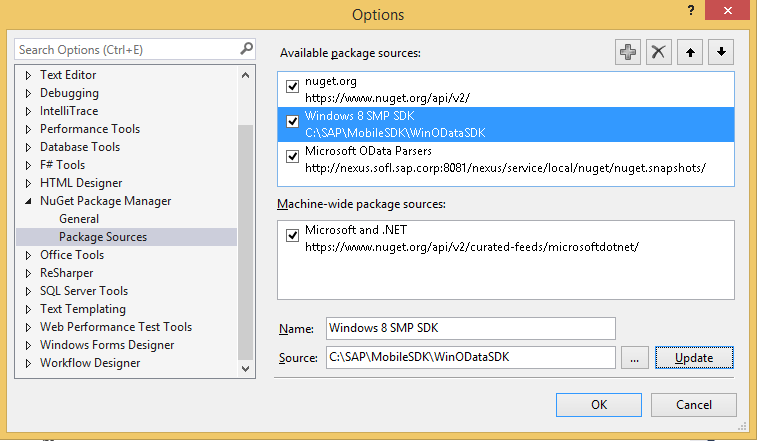
1. Open Visual Studio 2013 with Update 2. Click Tools -> NuGet Package Manager -> Package Manager Settings



1. Click on NuGet Package Manager -> Package Sources on the left pane. On the right pane, click the + sign on the right pane to add a new package source. Enter a name for the package and browse to the source of the unzipped Windows SMP SDK files. Click Update.

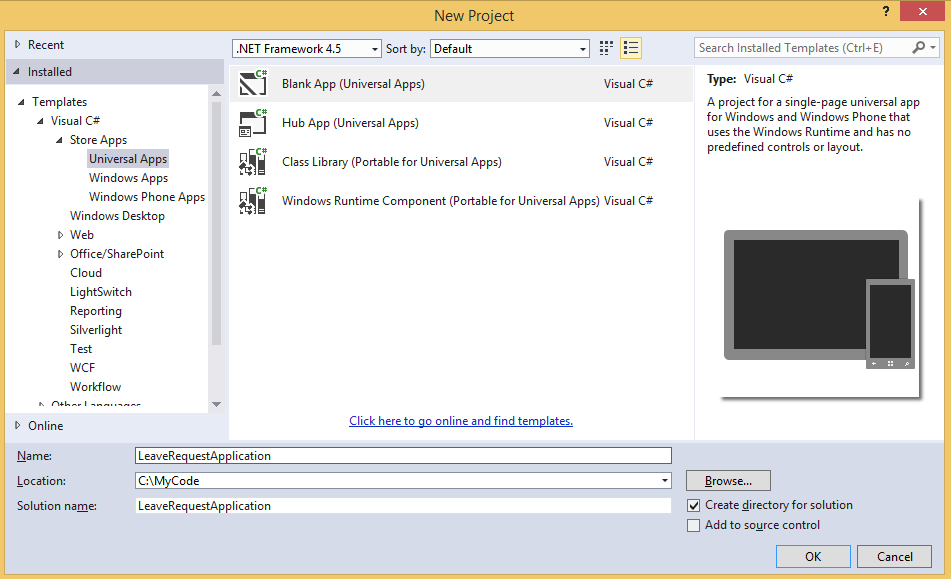
**Name:** Windows SMP SDK

**Source:** C:\SAP\MobileSDK3\NativeSDK\ODataFramework\Windows

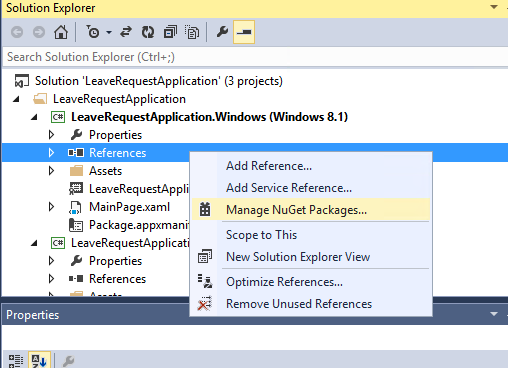


### Adding Windows SMP SDK references to the project

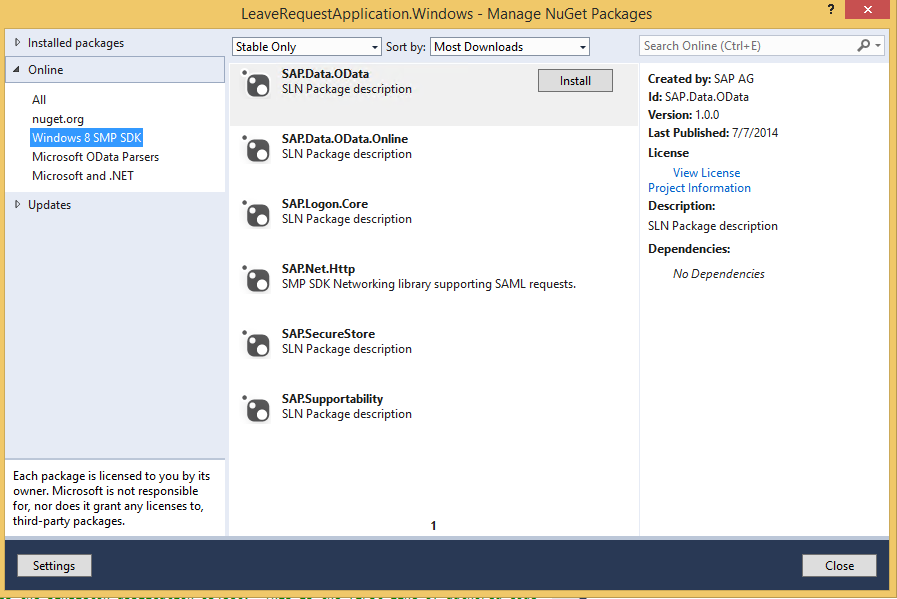
1. Create a new Windows Store Universal project by clicking on the New Project… link. Enter a name for the project.



1. To add references to the Windows project, right click on References and select Manage NuGet Packages…



1. Select the package source on the left pane that you created previously. Select the package that you need to add as a reference and click Install. NuGet Package Manager installs all dependent packages for you automatically. In addition, the proper package for the specified platform is installed.



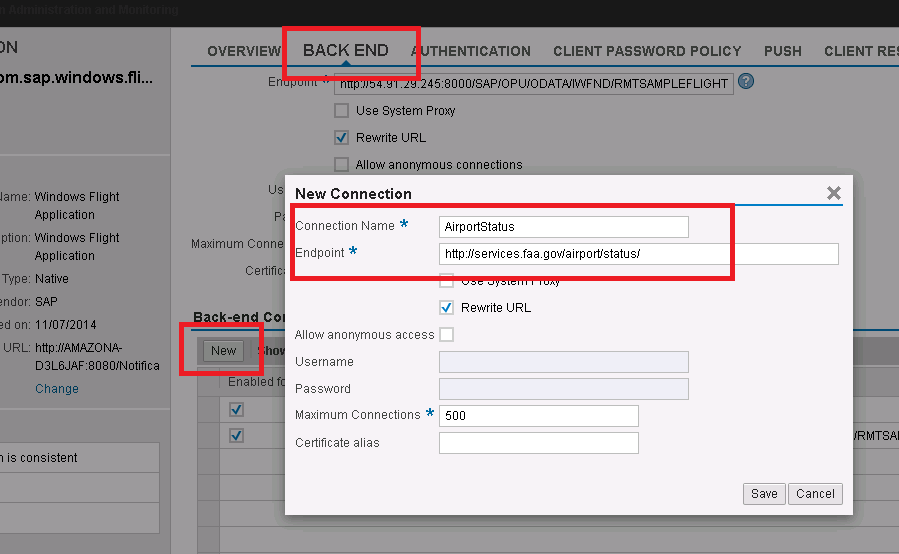
1. In addition to the SAP NuGet packages, the developer also needs to add references to the Microsoft OData library packages. This can be done directly by adding the packages from the NuGet gallery. Click on nuget.org on the left pane and search for ODataLib. From the Search Results, install the package ODataLib for OData v1-3 (The version is 5.6.2). This will install all the dependent packages. Click I Accept to follow the prompts to install the packages.

Machine generated alternative text: .org
Each package is licensed to you by its
owner. Microsoft is not responsible
for, nor does it grant any licenses to,
third-party packages.
ODatalib for WIndows Phone
Classes to serialize, deserialize and validate OData payloads.
Enables construction of OData producers and consumers.Tar...
ODataLib
Classes to serialize, deserialize and validate OData JSON
payloads. Supports OData v4 only.
EdmLib for OData vi-3
Classes to represent construct, parse, serialize and validate
entity data models. Targets NET 4.0, Silverlight 4.0, or .NET P...
System.Spatial for OData vi-3
Contains classes and methods that facilitate geography and
geometry spatial operations. Targets .NET 4.0, Silverlight 4.0...
WCF Data Services Client for OData vi-3
LINQ-enabled client API for issuing OData queries and
consuming OData payloads. Supports OData v3. Targets .NE...
i 2 b
Created by: Microsoft Corpoi
Id: Microsoft.Data,OData
Version: 5.6.2
Last Published: 8/1/2014
Downloads: 1656106
View License
Project Information
Report Abuse
Description:
Classes to serialize, deserialize and
validate OData payloads. Enables
construction of OData producers and
consumers. Targets .NET 4.0, Silverlight
4.0 or NET Portable Lib with support
for NET 4.0, SL 5.0, Win Phone 8, Win
Phone 8.1, and Win 8. Localized for CHS,
CHT, DEU, ESN, FRA, ITA, JPN, KOR and
RUS.
Tags: wcf data services odata odatalib
edmlib spatial adonet S entity
framework open protocol wcfds
wcfdataservices dataservices
Dependencies:
System.Spatial (t 5.6.2)
Microsoft.DataEdm (= 5.62)
t...a.. :s_..... ...L..... . . LS..... _..L.
settings
L ‘.JL/ULO Parsers
Windows SMP SDK
Microsoft and .NET
ODatalib for OData vi-3
Classes to serialize,
OData JSON payloads.
Install
P Updates
Q
Q
QMachine generated alternative text: License Acceptance fl
The following package(s) require a click-to-accept license:
System.Spatial (Author Microsoft corporation)
I Vie’. LicEnzE
Mkrosoft.Data.Edm (Author Microsoft Corporation)
View License
MicrosoftData.OData (Author Microsoft Corporation)
View License
By clicking “I Accept” you agree to the license terms for the package
(s) listed above. If you do not agree to the license terms, click “I
Decline.”
I ecIine I Accept

1. Follow steps 3 and 4 to add references to the Windows 8.1 Phone project as well.

## Create whitelisted connections

1. To create a whitelisted connection for an application, click on the BackEnd tab and click New.



1. Enter Connection Name and Endpoint URL.

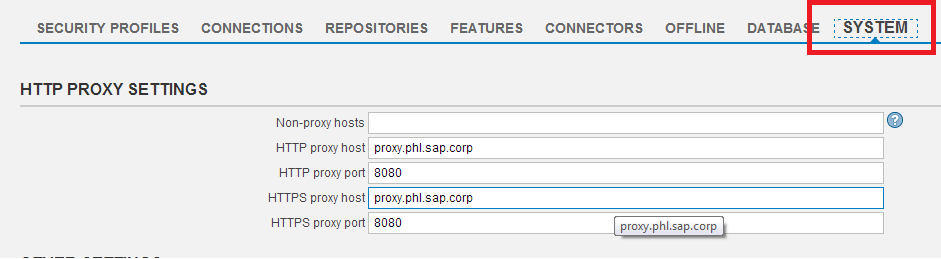
Connection Name: AirportStatus

Endpoint URL: <http://services.faa.gov/airport/status/>

1. Click Save and Save.

## Set up proxy server in SMP Server

1. To set up a proxy server in SMP Server, click on System and enter the host name of the proxy server and the port number.



1. Click Save.
2. Note that the SMP Server service needs to be restarted for the proxy server settings to take effect.