How To... Enable E2E Tracing (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](#_Toc396314354)

[2. Background Information 1](#_Toc396314355)

[3. Prerequisites 1](#_Toc396314356)

[4. Step-by-Step Procedure 2](#_Toc396314357)

[4.1 Windows Project 2](#_Toc396314358)

[4.2 End to End Tracing 3](#_Toc396314359)

[4.2.1 Steps involved in logging messages locally on device 3](#_Toc396314360)

[4.2.2 Steps involved in uploading log messages to SMP Server 3](#_Toc396314361)

[4.2.3 Steps involved in uploading trace messages to SMP Server 4](#_Toc396314362)

[4.2.4 End to End Tracing 5](#_Toc396314363)

[4.3 Running the application 7](#_Toc396314364)

[5. Appendix 10](#_Toc396314365)

[5.1 NuGet Package Manager 10](#_Toc396314366)

[5.1.1 Adding Windows SMP SDK package in Visual Studio 10](#_Toc396314367)

[5.1.2 Adding Windows SMP SDK references to the project 11](#_Toc396314368)

[5.1.3 Enabling devices to upload logs to SMP Server 13](#_Toc396314369)

[5.1.4 Enabling SMP Server for End to End Tracing 15](#_Toc396314370)

# 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 viewing flight information and booking flights, the application also needs to be able to log information on the device that can subsequently be uploaded to the server. Furthermore, trace logs containing request information also needs to be uploaded to the server. This would immensely help the help desk personnel troubleshoot any issues with the application.

# 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.

# Prerequisites

This exercise targets the Windows Desktop platform and has the following prerequisites:

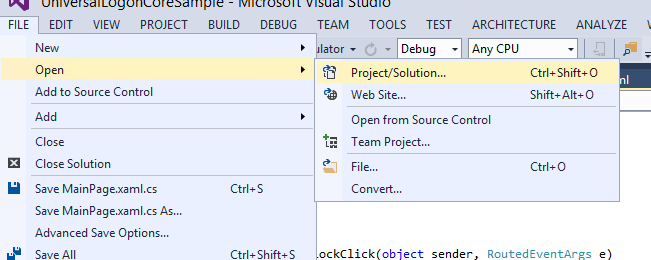
* Visual Studio 2013 with Update 2
* Any Windows machine running .NET 4.5
* To get the most out of this exercise, experience with Windows programming is recommended.

# Step-by-Step Procedure

In the previous exercises, we have successfully on-boarded a device, retrieved data from an OData endpoint and retrieved data from a non-OData JSON endpoint and also booked a flight. In this exercise, we will look into how we can log messages on the device, upload the logged messages and trace logs to the SMP Server.

## Windows Project

1. ...
   1. Open Visual Studio 2013 with Update 2 and open the solution RKT\_WPF\_E2ETracing.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. By default, devices will not be able to upload their logs and traces to the SMP Server. In order for devices to be able to upload their logs and traces, the administrator needs to configure the SMP Server. See appendix on how to configure the SMP Server to allow devices to upload their logs and traces.

## End to End Tracing

### Steps involved in logging messages locally on device

The SMP SDK allows the developer to log messages on the device file system and also the Output window of Visual Studio. Additionally, the developer can also read the log messages on the device.

The following steps are involved in logging messages locally on the device file system.

* Initializing the logManager
* Configuring basic settings (log level, where you want to log etc.)
* Logging the message
* Additionally, you can also read the log messages and display to user

The following code snippet logs messages to the file system and console and also reads the log messages.

|  |
| --- |
| // initializing logManager  var logManager = SAP.Supportability.SupportabilityFacade.Instance.ClientLogManager;    // basic settings  logManager.SetLogLevel(ClientLogLevel.Info);  logManager.SetLogDestination(ClientLogDestinations.Console | ClientLogDestinations.FileSystem);    // logging a message  var logger = logManager.GetLogger("testLogger");  logger.LogError("WPF - sample error message");    // read messages on device  await logManager.GetLogEntriesForLoggerAsync("testLogger", ClientLogLevel.All); |

### Steps involved in uploading log messages to SMP Server

In addition to logging messages on the device, the developer can also optionally upload the log messages to the SMP Server. Note that in order to upload the log messages from the device to the SMP Server, the SMP administrator must have already enabled the device to upload log messages to the SMP Server.

The following steps are involved in uploading log messages to the SMP Server.

* Call the UploadClientLogsAsync method of the ClientLogManager class.

|  |
| --- |
| try  {  await SAP.Supportability.SupportabilityFacade.Instance.ClientLogManager.UploadClientLogsAsync(new SupportabilityUploader(httpClient, false));  }  catch (Exception ex)  {  var supportabilityException = ex as SAP.Supportability.ISupportabilityException;  message = ex.Message + ((supportabilityException != null) ? ("(" + supportabilityException.UploadResult.ResponseStatusCode + ")") : "");  } |

The UploadClientLogsAsync method takes an interface IUploader as an argument. The developer is responsible for creating a class that derives from the IUploader interface. An instance of this class is then passed as an argument for the UploadClientLogsAsync.

### Steps involved in uploading trace messages to SMP Server

In addition to uploading log messages to the SMP Server, the developer can also optionally upload the trace messages to the SMP Server. Note that in order to upload the trace messages from the device to the SMP Server, the SMP administrator must have already enabled the SMP Server for End to End Tracing.

The following steps are involved in uploading log messages to the SMP Server.

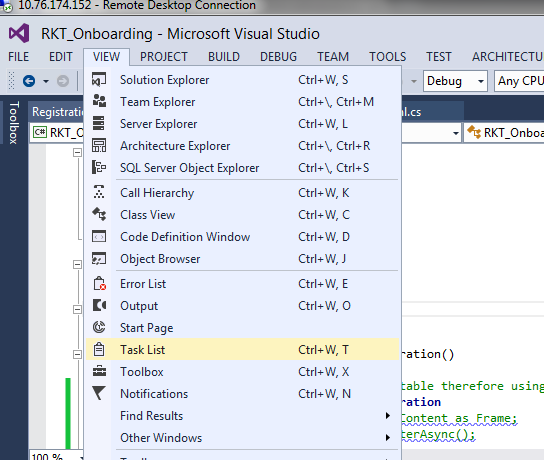
* Call the UploadBtxAsync method of the ClientLogManager class.

|  |
| --- |
| try  {  await SAP.Supportability.SupportabilityFacade.Instance.ClientLogManager.UploadBtxAsync(new SupportabilityUploader(httpClient));  }  catch (Exception ex)  {  var supportabilityException = ex as SAP.Supportability.ISupportabilityException;  message = ex.Message + ((supportabilityException != null) ? ("(" + supportabilityException.UploadResult.ResponseStatusCode + ")") : "");  } |

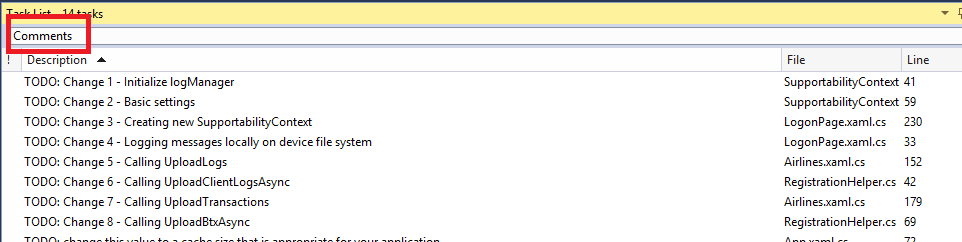
The UploadBtxAsync method takes an interface IUploader as an argument. The developer is responsible for creating a class that derives from the IUploader interface. An instance of this class is then passed as an argument for the UploadBtxAsync.

### End to End Tracing

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 8 tasks for this exercise.



* 1. Open SupportabilityContext.cs in the sample application.
  2. Go to the LogManager property. In between the BEGIN and END //TODO Change1: markers enter the following code:

|  |
| --- |
| if (logManager == null)  {  logManager = SAP.Supportability.SupportabilityFacade.Instance.ClientLogManager;  }  return logManager; |

The LogManager property now returns an instance of the ClientLogManager class.

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

|  |
| --- |
| if (logger == null)  {  LogManager.SetLogLevel(SAP.Supportability.Logging.ClientLogLevel.Info);  LogManager.SetLogDestination(SAP.Supportability.Logging.ClientLogDestinations.FileSystem | SAP.Supportability.Logging.ClientLogDestinations.Console);  logger = LogManager.GetLogger("testLogger");  }  return logger; |

The log level and log destination values are now set on the LogManager property. The GetLogger method is called on the LogManager to return the Logger property.

* 1. Open LogonPage.xaml.cs in the sample application.
  2. Go to the Window\_Loaded event handler. In between the BEGIN and END //TODO Change3: markers enter the following code:

|  |
| --- |
| if (LoggingContext.Context == null)  {  LoggingContext.Context = new SupportabilityContext();  } |

A new instance of the SupportabilityContext class is created and assigned to LoggingContext.Context object.

* 1. Go to the GoToState method. In between the BEGIN and END //TODO Change4: markers enter the following code:

|  |
| --- |
| LoggingContext.Context.Logger.LogInfo("Entering state: " + state); |

Log messages are stored on the device file system. Log messages at various critical points in your code.

* 1. Open Airlines.xaml.cs in the sample application.
  2. Go to the MenuItemUploadLogsClick event handler. In between the BEGIN and END //TODO Change5: markers enter the following code:

|  |
| --- |
| try  {  await RegistrationHelper.UploadLogs();  }  catch (Exception ex)  {  var supportabilityException = ex as SAP.Supportability.ISupportabilityException;  message = ex.Message + ((supportabilityException != null) ? ("(" + supportabilityException.UploadResult.ResponseStatusCode + ")") : "");  } |

The UploadLogs helper method is called within a try-catch block.

* 1. Open RegistrationHelper.cs in the sample application.
  2. Go to the UploadLogs method. In between the BEGIN and END //TODO Change6: markers enter the following code:

|  |
| --- |
| await SAP.Supportability.SupportabilityFacade.Instance.ClientLogManager.UploadClientLogsAsync(new SupportabilityUploader(LoggingContext.Context.HttpClient, false)); |

The UploadClientLogsAsync method is called on the ClientLogManager class. The UploadClientLogsAsync method takes an interface IUploader as an argument.

* 1. Open Airlines.xaml.cs in the sample application.
  2. Go to the MenuItemUploadTraceClick event handler. In between the BEGIN and END //TODO Change7: markers enter the following code:

|  |
| --- |
| try  {  await RegistrationHelper.UploadTransactions();  }  catch (Exception ex)  {  var supportabilityException = ex as SAP.Supportability.ISupportabilityException;  message = ex.Message + ((supportabilityException != null) ? ("(" + supportabilityException.UploadResult.ResponseStatusCode + ")") : "");  } |

The UploadTransactions helper method is called within a try-catch block.

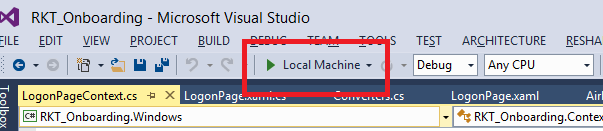
* 1. Open RegistrationHelper.cs in the Shared project.
  2. Go to the UploadTransactions method. In between the BEGIN and END //TODO Change8: markers enter the following code:

|  |
| --- |
| await SAP.Supportability.SupportabilityFacade.Instance.E2ETraceManager.UploadBtxAsync(new SupportabilityUploader(LoggingContext.Context.HttpClient)); |

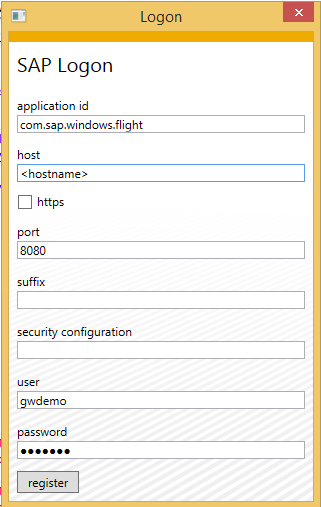
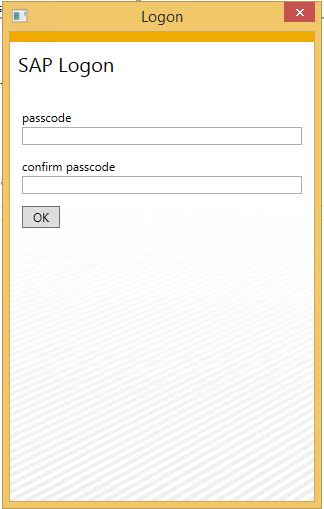
The UploadBtxAsync method is called on the ClientLogManager class. The UploadBtxAsync method takes an interface IUploader as an argument.

## Running the application

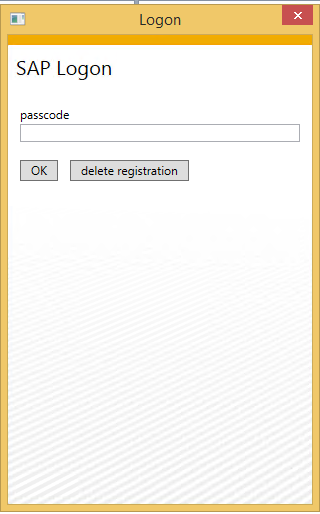
1. Run the application by clicking Run from within Visual Studio 2013 Update 2.



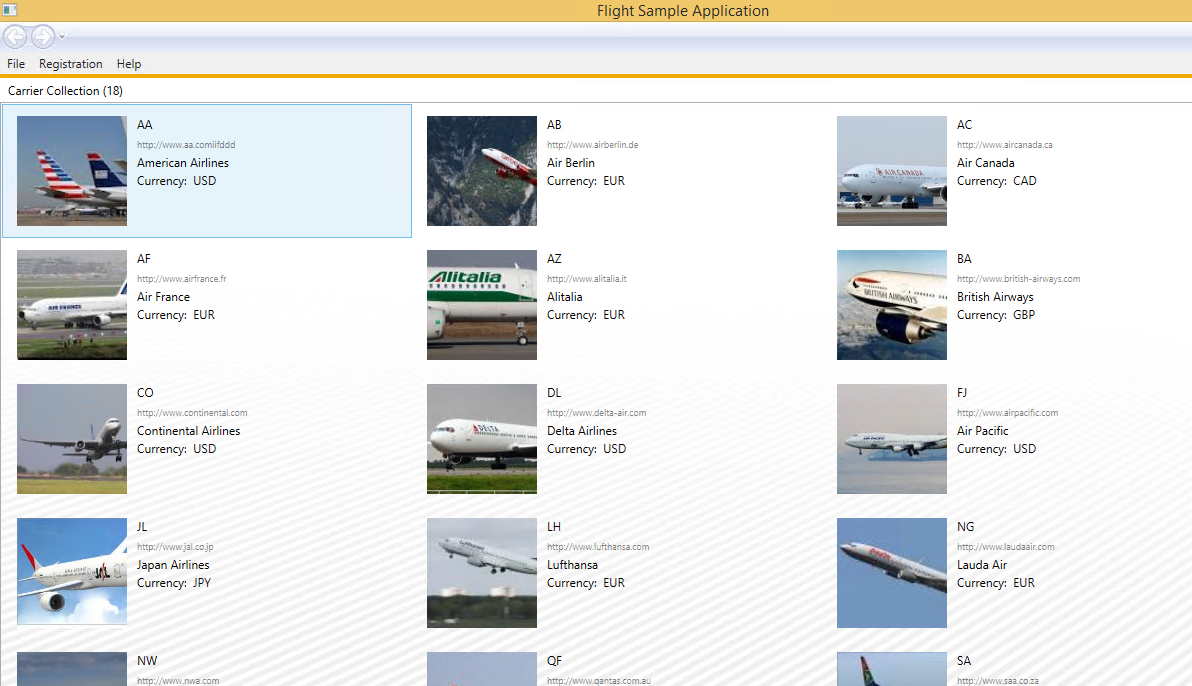
1. If you are not registered, click register. Enter passcode for data vault and click Ok.

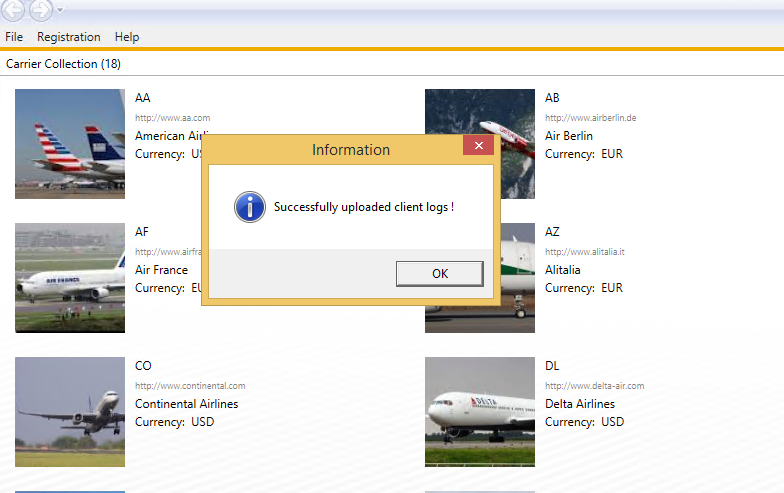
1. If you are already registered, enter the passcode to unlock the data vault.



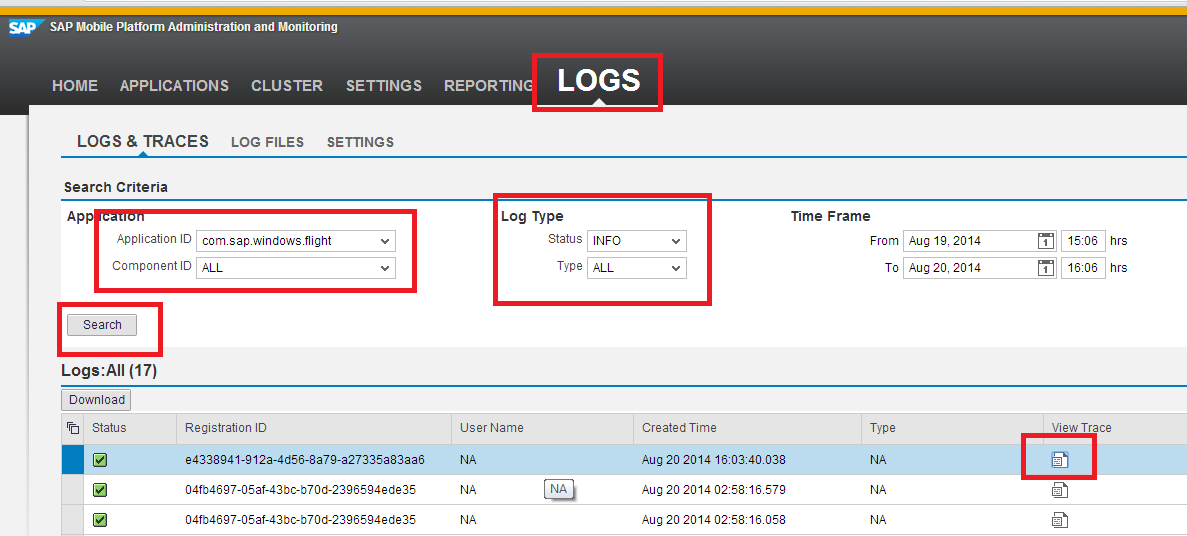
1. The application should now open up. From the File menu, click on Upload Logs.

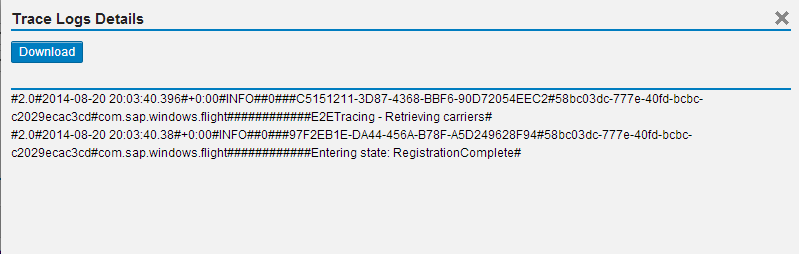


1. Once the logs are uploaded to the SMP Server, you should see a confirmation message on the screen.



1. From within SMP Administrator Cockpit, click on Logs. Select the application id (com.sap.windows.flight). Set Log Status to Info. Click Search. Click on View Trace icon. You should see the logs uploaded to the SMP Server.





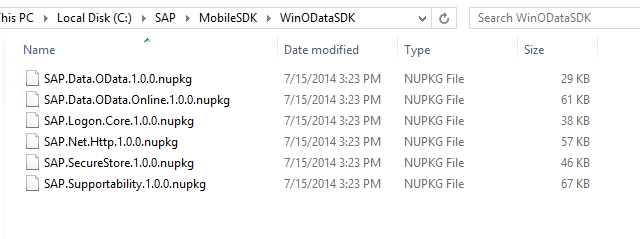
# Appendix

## NuGet Package Manager

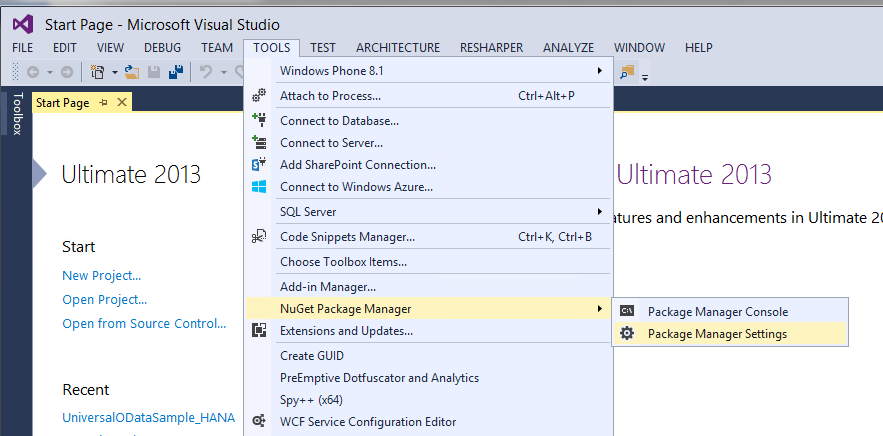
NuGet is the package manager for the Microsoft development platform including .NET. The NuGet client tools provide the ability to produce and consume packages. Starting with Visual Studio 2012, NuGet is included in every edition (except Team Foundation Server) by default. Updates to NuGet can be found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



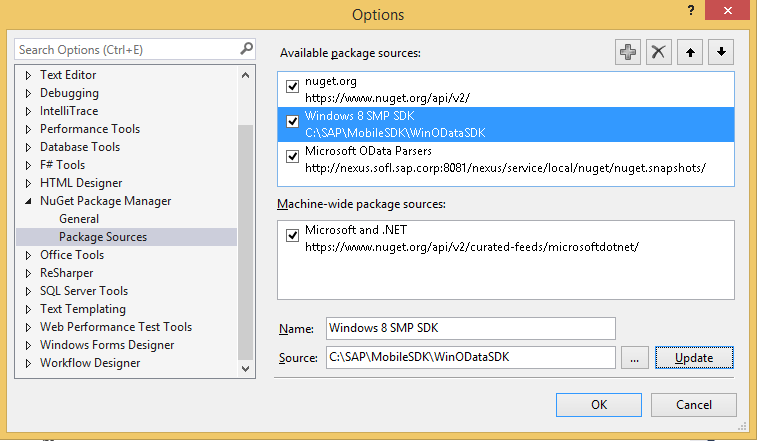
1. Open Visual Studio 2013 with Update 2. Click Tools -> NuGet Package Manager -> Package Manager Settings



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.

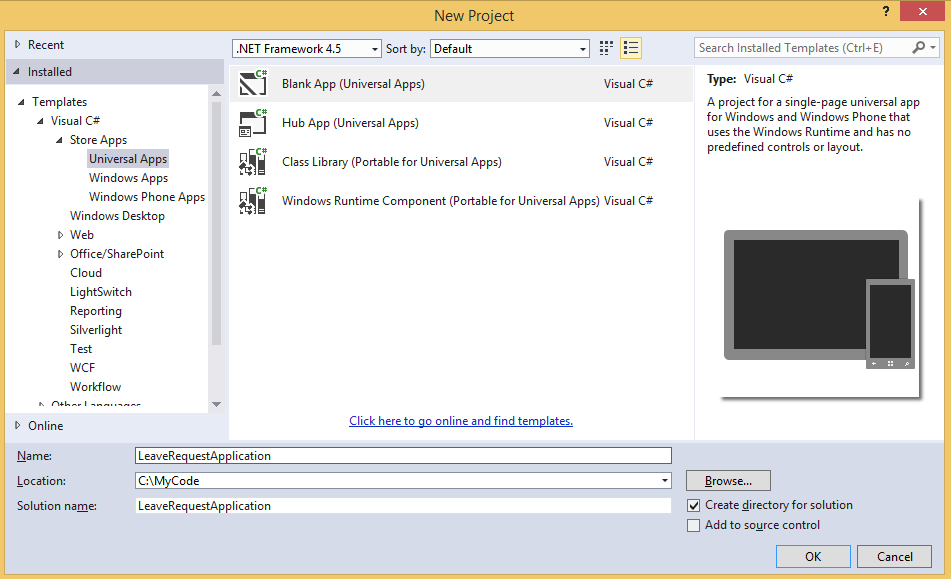
**Name:** Windows SMP SDK

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

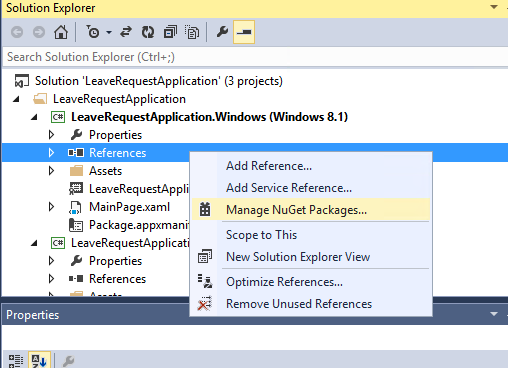


### Adding Windows SMP SDK references to the project

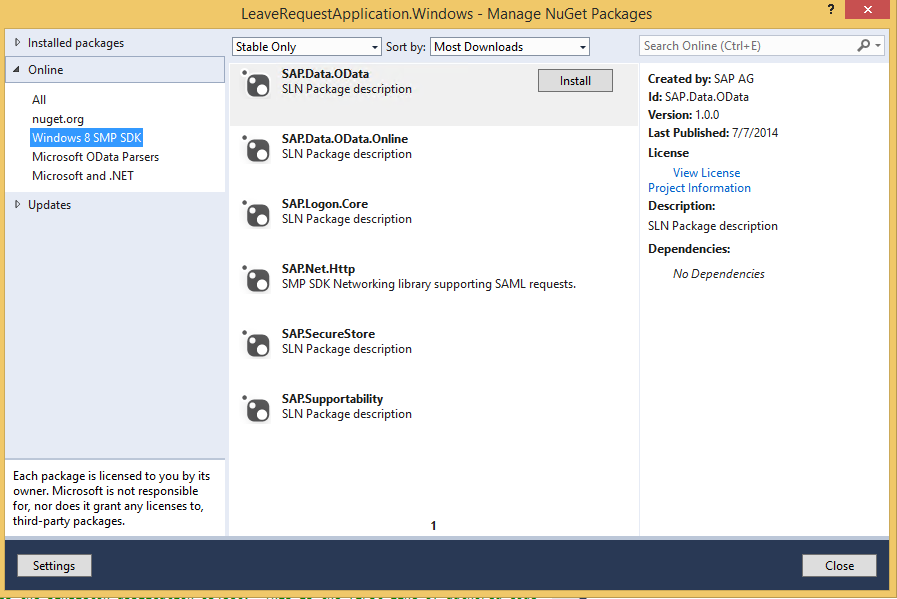
1. Create a new Windows Store Universal project by clicking on the New Project… link. Enter a name for the project.



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.



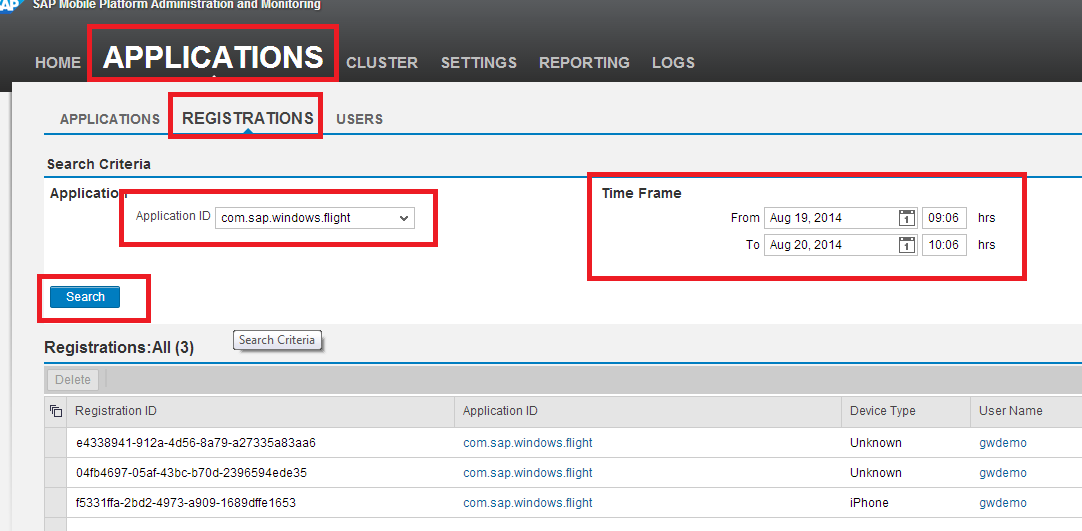
1. In addition to the SAP NuGet packages, the developer also needs to add references to the Microsoft OData library packages. This can be done directly by adding the packages from the NuGet gallery. Click on nuget.org on the left pane and search for ODataLib. From the Search Results, install the package 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

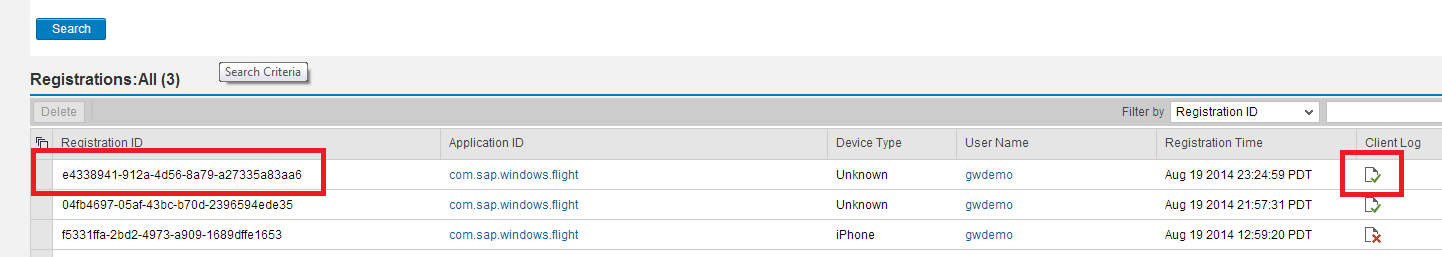
1. Follow steps 3 and 4 to add references to the Windows 8.1 Phone project as well.

### Enabling devices to upload logs to SMP Server

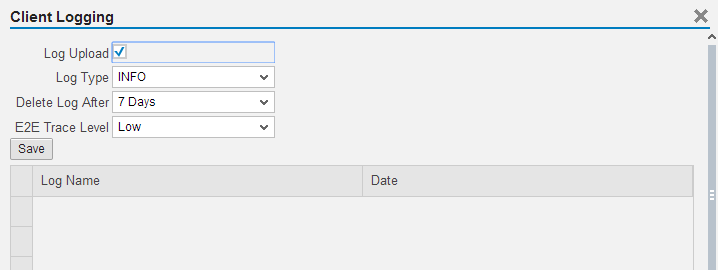
1. From within SMP Administrator Cockpit, click on Applications -> Registrations and select application id (com.sap.windows.flight) from the drop down menu.



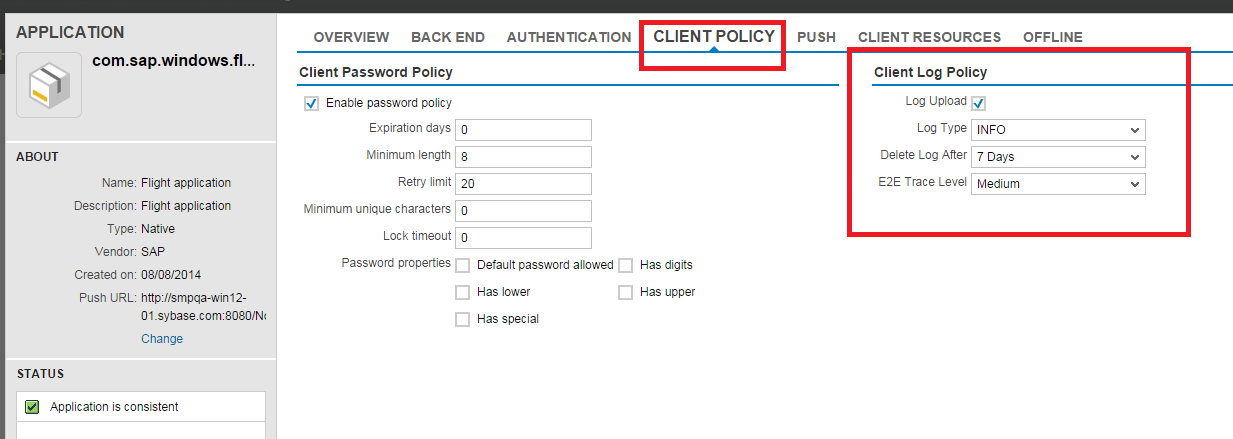
1. Select the device and click on the Client Log icon



1. In the Client Logging dialog box, check the Log Upload and set the various values. Note that this needs to be done for each device that the administrator is interested in retrieving logs and traces.



1. Alternately, you can enable logging and tracing at the application level.  This would enable logging and tracing on all devices that are registered to that specific application.



### Enabling SMP Server for End to End Tracing

1. From within SMP Administrator Cockpit, click on Settings -> System
2. Enter proper values for Solution Manager URL and also check the Enabled for End to End Tracing.

