Getting Started with the Ed-Fi ODS and Ed-Fi ODS API

Ed-Fi ODS and Ed-Fi ODS API Version 2.0 - Technical Preview

October 2014

For the latest information about the Ed-Fi Alliance visit our website at www.ed-fi.org.

Contents

Introduction

This document outlines the steps necessary to download, configure, and deploy the Ed-Fi ODS (Operational Data Store) and Ed-Fi ODS API (Application Programming Interface). The Ed-Fi ODS API enables applications to read and write education data stored in an Ed-Fi ODS through a secure REST interface. The Ed-Fi ODS API supports both transactional and bulk modes of operation.

The Ed-Fi ODS and Ed-Fi ODS API are technical components of the Ed-Fi Implementation Suite.

Audience

This document is for technical professionals who work with educational data—including business analysts, database administrators, and software developers.[1]

Prerequisites

Familiarity with the following technologies is required for installing and configuring the software components:

- Microsoft Windows (installation and configuration)
- PowerShell
- Visual Studio 2013 (Professional Edition or higher)
    - ASP.NET MVC 4/ C#
    - NuGet (Package Manager)

[1]You need to be a licensee of Ed-Fi technology, with access to the Ed-Fi repository on GitHub, in order to download and work with the Ed-Fi ODS 2.0 and the Ed-Fi ODS API 2.0. Licensing information is available at http://www.ed-fi.org/ed-fi-solution-works/license. If you are an Ed-Fi licensee, and need to get access to the Ed-Fi repository, please send an email that includes your GitHub username to info@ed-fi.org.

- Microsoft SQL Server 2012
    - SQL Server Management Studio (SSMS)

Tested Configurations

The Ed-Fi ODS API configuration described in this document has been tested with the following software configurations:

- Windows 7 Professional (64-bit) and Ultimate (64-bit), Windows Server 2012, or the Microsoft Azure development fabric
- Microsoft SQL Server 2012 (Developer Edition or Standard Edition)[2]
- Visual Studio 2013 with Update 3 (Professional Edition or higher)

Overview: The Ed-Fi ODS API Setup

The steps required to prepare and load the Ed-Fi ODS and Ed-Fi ODS API can be summarized as follows:

1. Install and configure Windows components
2. Install and configure required software
3. Download the Ed-Fi ODS source code
4. Prepare the development environment
5. Build the Visual Studio solution
6. Set the startup projects
7. Run the Ed-Fi ODS API

[2] Express Edition is NOT supported.

Step 1: Install and Configure Windows Components

Verify that the following Windows components are installed and configured:

- **.NET 4.5 Framework** (full install, included with Visual Studio 2013) http://msdn.microsoft.com/en-us/library/5a4x27ek(v=vs.110).aspx
- **IIS Express** (included with Visual Studio 2013)
- **PowerShell 3.0** (included with the Windows Management Framework 3.0) http://www.microsoft.com/en-us/download/details.aspx?id=34595
- **Microsoft Message Queue (MSMQ)** (Windows feature included in all versions of Windows 7/8 and Windows Server 2008+)

Step 2: Install and Configure Required Software

Install the three following software components:

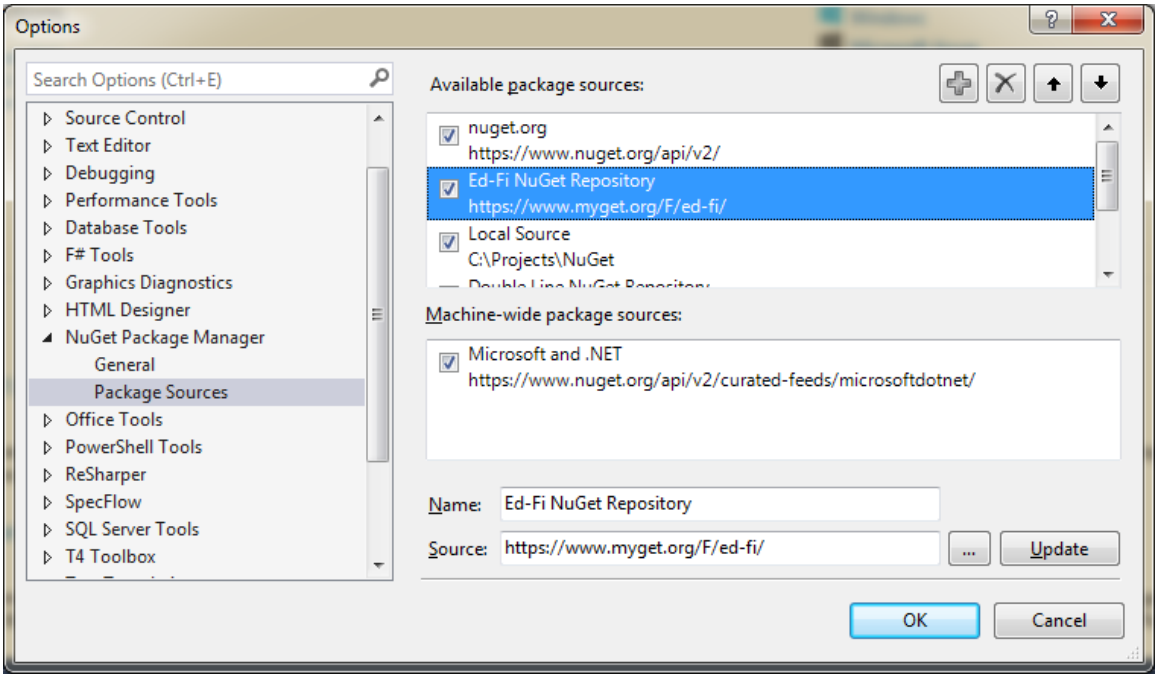
- **Visual Studio 2013** (Professional or Premium Edition)
- **Microsoft SQL Server 2012 – with Service Pack 2** (Developer Edition, Standard Edition, or Enterprise Edition)
- **Microsoft Message Queue** (MSMQ) **Server Core**

Installation details and configuration procedures for these software components are described below.

Visual Studio 2013

In addition to a base installation of Visual Studio 2013 (Professional or Premium Edition), the following components must be installed and configured:

1. Install Visual Studio 2013 SDK (http://www.microsoft.com/en-us/download/details.aspx?id=40758).
2. Install Modeling SDK for Microsoft Visual Studio 2013 (http://www.microsoft.com/en-us/download/details.aspx?id=40754).
3. Install Microsoft SQL Server 2012 Data-Tier Application Framework (http://www.microsoft.com/en-us/download/details.aspx?id=39976).
4. Check the Visual Studio start page for Visual Studio 2013 updates (currently at Update 3).
5. Check Windows Update for updates to VS 2013.
6. Add the following Nuget feed as an available package source (Visual Studio->Tools-Nuget Package Manager->Package Manager Settings->Package Sources): https://www.myget.org/F/ed-fi/.



Microsoft SQL Server 2012

In addition to a base installation of Microsoft SQL Server 2012 (Developer Edition, Standard Edition, or Enterprise Edition), the following components must be installed and configured. In SQL Server 2012 setup:

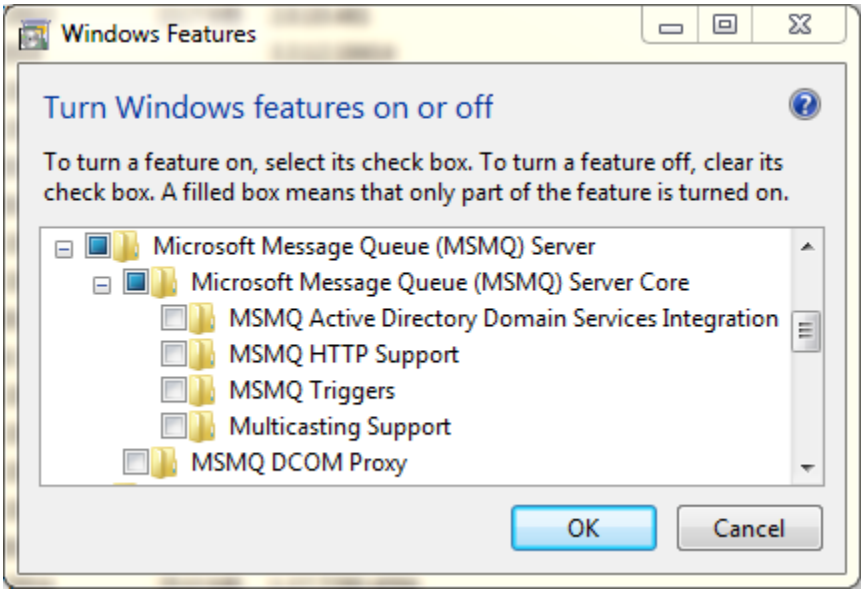
1. When prompted, select the following features:
    - "Database Engine Services" (Replication/text search/data quality services are NOT required.)
    - "SQL Server Data Tools"
    - "Management Tools – Complete"
2. Use the default instance ("MSSQLSERVER").
3. Select "SQL Server" and "Windows Authentication Mode."
4. In the "Specify SQL Server administrator," click on "Add Current User."

Microsoft Message Queue Server Core

The MSMQ Server Core must be installed and configured.

1. From the Control Panel | Programs | Programs and Features control panel select the "Turn Windows features on or off" menu.

2. Select the “Microsoft message Queue (MSMQ) Server Core” option as shown below.



3. Press “OK” to confirm installation of MSMQ.

Step 3: Download the Ed-Fi ODS Source Code

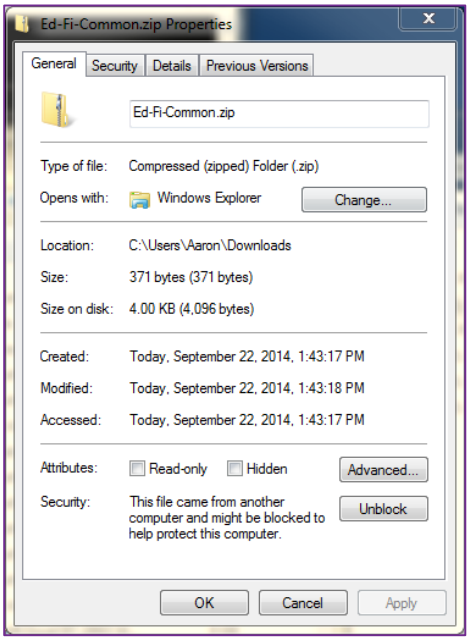
The Ed-Fi ODS source code is contained in three Ed-Fi repositories located at github.com (https://github.com/Ed-Fi-Alliance). The repositories are:

- Ed-Fi-Common
- Ed-Fi-ODS
- Ed-Fi-ODS-Implementation

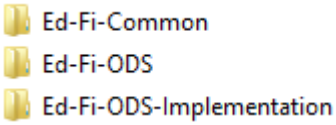
Follow these steps to download each of the repository archives and then extract them:

1. Navigate to https://github.com/Ed-Fi-Alliance/Ed-Fi-Common, https://github.com/Ed-Fi-Alliance/Ed-Fi-ODS, and https://github.com/Ed-Fi-Alliance/Ed-Fi-ODS-Implementation, downloading each of the repository archives to your local drive.

2. In Windows Explorer, right-click each of the downloaded ZIP files and select "Properties." On the General tab, press the "Unblock" button to allow the contents of the contained scripts to execute properly.



3. In Windows Explorer, right-click on each of the downloaded ZIP files and select the "Extract All…" menu item. Enter "C:\" for the target folder. The ZIP files contain an embedded folder ending in "-master" or "-development". For example, the "Ed-Fi ODS ZIP" archive contents will be extracted into "C:\Ed-Fi-ODS-development."
4. After the extractions are complete, rename the folders to remove the "-master" or "-development" from the folder names. For example, change "C:\Ed-Fi-ODS-master" to C:\Ed-Fi-ODS."[3] When the extraction and renaming is complete, there should be three folders for the dashboard source code as shown below.



[3] If you decide to use a different path be sure to substitute your chosen path where appropriate in the setup instructions throughout this document.

Step 4: Prepare the Development Environment

Preparing the development environment involves the following procedures (which are discussed below):

- Configure MSBuild
- Restore any missing NuGet packages
- Create SQL logins
- Initialize PowerShell scripts for development
- Install a development certificate
- Initialize the development environment

Configure MSBuild

Because packages compiled from some of the projects are used during the code generation of later projects, there is a potential for earlier packages to be locked if the solution is recompiled in rapid succession. This is due to the way that MSBuild caches build processes to minimize compile time by default. Build processes are normally held for reuse for approximately 10 minutes.

To turn off this default behavior when compiling in Visual Studio, the "MSBuildDisableNodeReuse" variable must be set. There are two options for setting this variable:

- Setting the variable globally
- Setting the variable per command prompt session

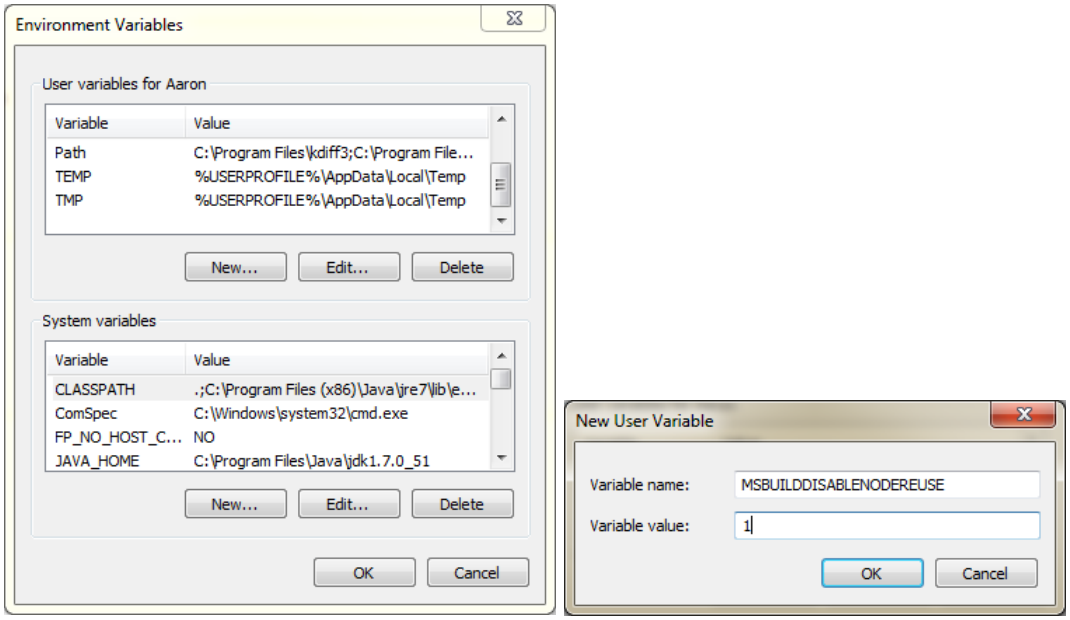
Both options tell the compiler to create a new process for each build job. As a side effect, this action also releases any resources that may be held by inactive compiler processes.

If these settings are not applied, Visual Studio or MSBuild may lock resources during build. Restarting Visual Studio, or the command prompt session, will resolve the problem for the first build after the restart. Waiting for up to 15 minutes between builds will also achieve this.

The options for setting this variable are described below:

Option 1: Setting "MSBuildDisableNodeReuse" Globally

To set the "MSBuildDisableNodeReuse" variable globally use the "Environment Variables" property page, which is accessed by right-clicking Computer, clicking Properties, and clicking the "Environment Variables" button under the "System Properties" dialog window Advanced tab (see below).



To turn off this behavior when compiling the solution using MSBuild, include the following compiler flag:

```
/nr:false
```

Option 2: Setting "MSBuildDisableNodeReuse" per Command Prompt Session

If the MSBuildDisableNodeReuse" variable is set within a command prompt session, the Visual Studio development environment must be launched using the `devenv` command (rather than from an icon):

```
SET MSBUILDDISABLENODEREUSE=1
```

Restore Any Missing NuGet Packages

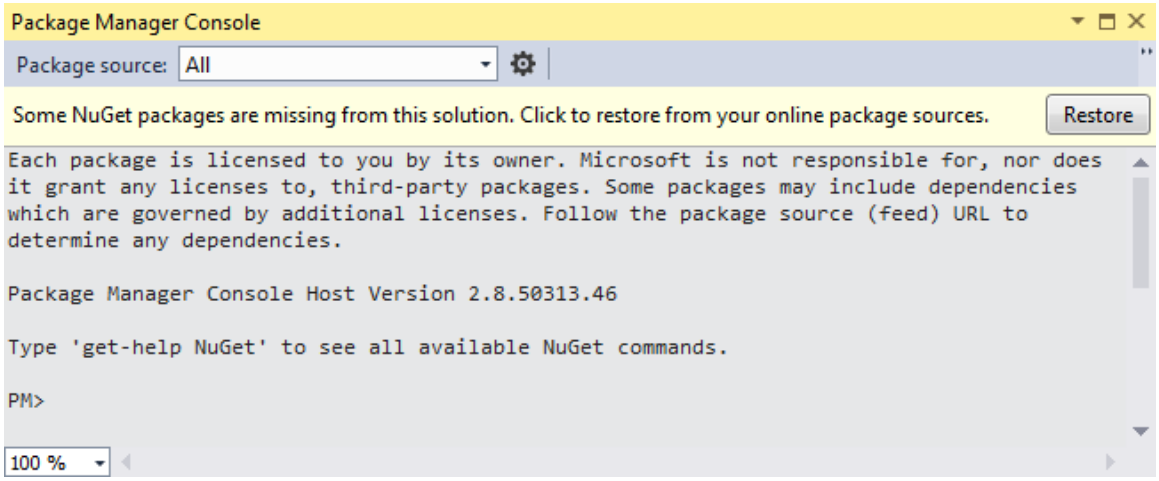
The projects within the Ed-Fi ODS API solution use many NuGet packages, which may need to be restored for a successful build.

To check for missing NuGet packages and restore any that are missing:

1. Start Visual Studio[4] and open C:\Ed-Fi-ODS-Implementation\Application\Ed-Fi-ODS.sln.

2. Open the "Package Manager Console" (from the Tools | NuGet Package Manager | Package Manager Console menu) and press "Restore." (Shown below.) This will download all the packages required to build the solution and configure the environment. Depending upon your internet connection, this may take several minutes.

[4] When starting Visual Studio, it is recommended that you "Run as administrator".



3. Close Visual Studio when all of the missing packages have been downloaded.

Create SQL Logins

Some SQL logins need to be created in SQL Server. These may be added by running the following script from within Microsoft SQL Server Management Studio:

```
C:\Ed-Fi-ODS-
   Implementation\Application\DeveloperSetup\CreateLocalLogins.sql
```

Initialize PowerShell Scripts for Development

There are several databases that must be successfully deployed. PowerShell scripts that initialize all necessary development databases are included in the Visual Studio solution.

These scripts are enabled for use within Visual Studio when the Ed-Fi-ODS solution is opened.[5] They may also be loaded for use within a PowerShell console window[6] by running the initialize PowerShell for development script located at:

`C:\Ed-Fi-ODS-Implementation\Initialize-PowershellForDevelopment.ps1`[7]

In either case, when the scripts are loaded, a series of "sourcing" statements and other information will indicate that the appropriate scripts have been loaded (see the following figure).

[5] To Initialize the Development Environment within Visual Studio, you must close and reopen the solution after restoring the NuGet packages.

[6] When starting a PowerShell session, it is recommended that you "Run as administrator".

[7] In order to run the scripts from a PowerShell console, the solution must be loaded at least once from Visual Studio in order to retrieve the required NuGet packages – as described in an earlier step. PowerShell may need to have its execution policy changed to "unrestricted." Within a PowerShell window run as Administrator, execute `Set-ExecutionPolicy Unrestricted` to enable this setting.

```
Sourcing: 01_SolutionPaths.ps1
Using repositories from environment variable: Ed-Fi-Common;Ed-Fi-Ods;Ed-Fi-ODS-Implementation
Sourcing: 02_DeploymentSettings.ps1
Sourcing: 03_ModuleImports.ps1
Sourcing: 04_GlobalFunctions.ps1
Sourcing: Add-MigrationToAdminDatabase.ps1
Sourcing: Add-MigrationToRestApiWorkingDatabase.ps1
Sourcing: Clean-BuildOutput.ps1
Sourcing: Initialize-DevelopmentEnvironment.ps1
Sourcing: Initialize-IIS.ps1
Sourcing: Initialize-LocalTestSettings.ps1
Sourcing: Initialize-MessageQueues.ps1
Sourcing: Migrate-EntityFrameworkDatabase.ps1
Sourcing: Publish-DatabaseToAzure.ps1
Sourcing: PublishLocal.ps1
Sourcing: Rebuild-Solution.ps1
Sourcing: Remove-Sandboxes.ps1
Sourcing: Reset-Databases.ps1
Sourcing: Reset-GeneratedArtifacts.ps1
Sourcing: Set-DeveloperSetting.ps1
```

Install a Development Certificate

A development certificate needs to be installed as a prerequisite to initializing the databases. The certificate is used to encrypt and decrypt connection strings in configuration files as well as database user names and passwords. The

certificate is located in the Ed-Fi-ODS-Implementation/logistics/certificates directory and is named “Development.pfx.”[8]

This certificate is installed by issuing the “Initialize-ProjectCredentials” command from within your PowerShell environment *after* the Initialize-PowershellForDevelopment.ps1 script has been run. This script will install the certificate for use by administrators and the user who runs the script on the local machine. If you have a build agent or other users who use the same machine, they must be added to the certificate by using the “All Tasks | Manage Private Keys…” dialog from the context menu of the “Development_Encryption” certificate in the Certificates MMC snap-in for the local computer.

Alternatively, you can install the certificate manually to the Local Computer Personal Certificates store using the Certificates snap-in to the Microsoft Management Console (mmc.exe). The provided certificate has an empty password. If you are prompted for a password, press “Enter.”

Initialize the Development Environment

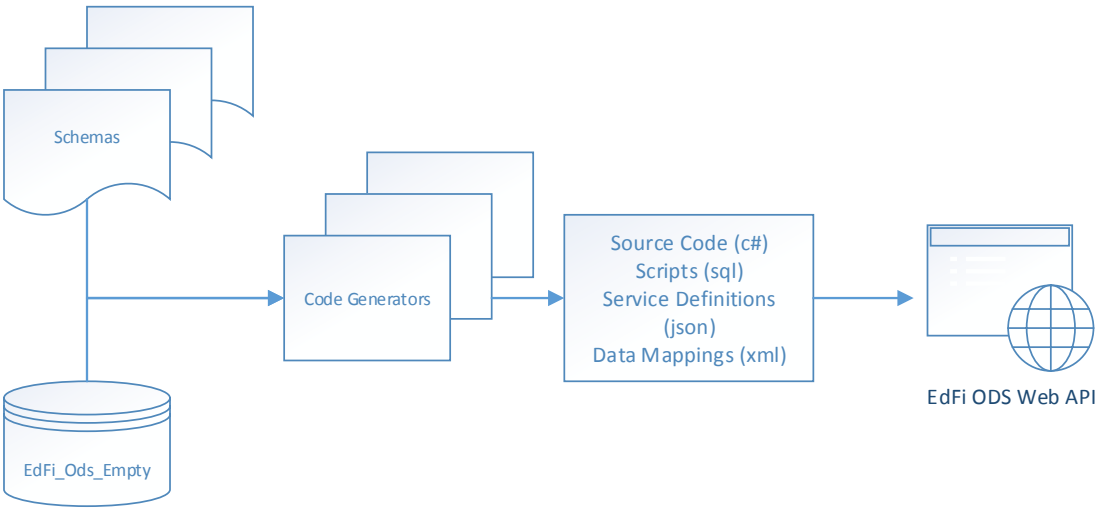
Once the PowerShell development scripts have been loaded and a development certificate has been installed, the development environment may be initialized by typing the `initdev` command in a PowerShell console. This command creates databases, generates code templates, and compiles projects in the solution.

Note: Initializing the development environment will take several minutes to complete.

[8] If you would prefer to generate your own certificate for your development environment, instructions are contained in the Appendix of this document.

Step 5: Build the Visual Studio Solution

The following diagram shows how the XML schema and empty ODS Database are used to create the Ed-Fi ODS API using code generators within the solution. The presence of the “EdFi_Ods_Empty” database is necessary because the code generation uses the database to understand the structure that it uses to generate data access code.



When the “EdFi_Ods_Empty” database has been created, there are two ways to build the solution. Either option may be used.

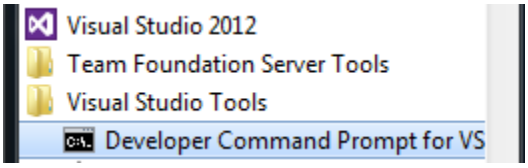
- The solution may be rebuilt using MSBuild from the Visual Studio command prompt
- The solution may be built from within Visual Studio

Option 1: Build from the Developer Command Prompt

The solution may be built from a Developer Command Prompt for Visual Studio using the Windows start menu.

To do a clean build from the command prompt:

1. Open the Developer Command Prompt for Visual Studio (see below).



2. Navigate to your \Ed-Fi-ODS-Implementation\Application menu.

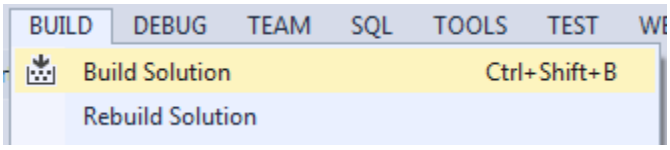
3. Issue a command similar to the following:

```
Msbuild /nr:false /t:clean;build Ed-Fi-Ods.sln
```

Option 2: Build from within Visual Studio

To build the solution from within Visual Studio:

1. Ensure that the MsBuildDisableNodeReuse flag is set (see “Step 4: Configure MSBuild”).
2. Open Visual Studio 2013 either from the Visual Studio 2013 Developer Command Prompt (where you set this value by typing `devenv`) or, if the property has been set globally, just starting Visual Studio normally.
3. Within Visual Studio, open the “Ed-Fi-Ods.sln” solution file from the \Ed-Fi-ODS-Implementation\Application directory.
4. Select “Build Solution” from the “Build” menu (or press Ctrl+Shift+B).

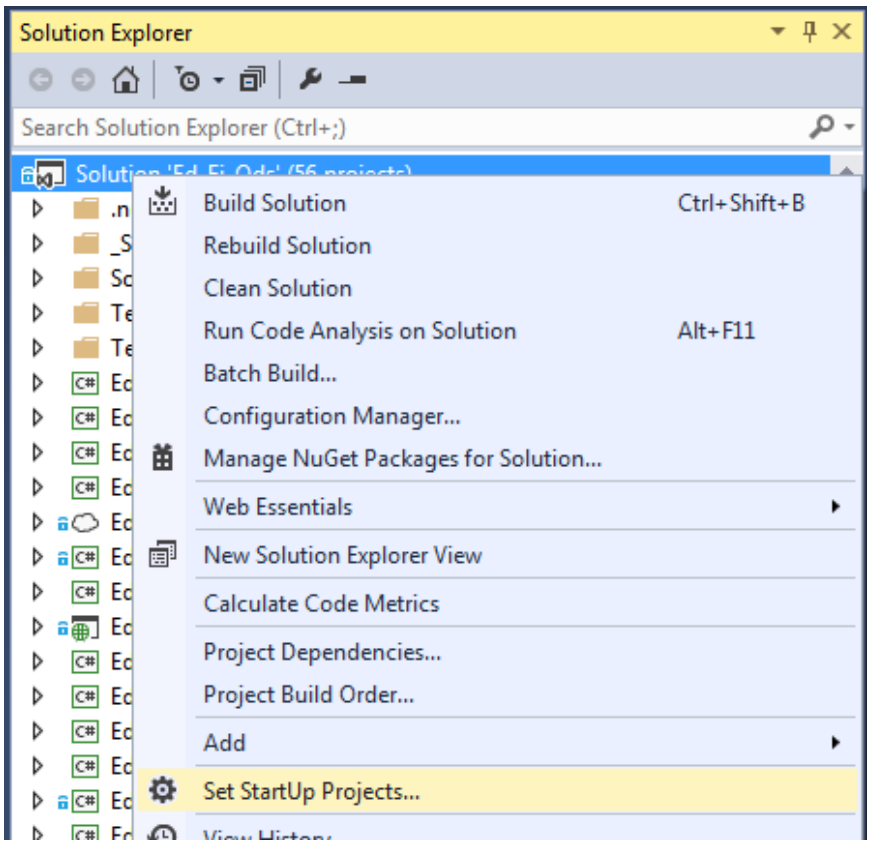


Step 6: Set the Startup Projects

The Ed-Fi ODS API solution consists of several “Startup Projects” that work together. Each of these projects need to be running for the solution to be fully functional.

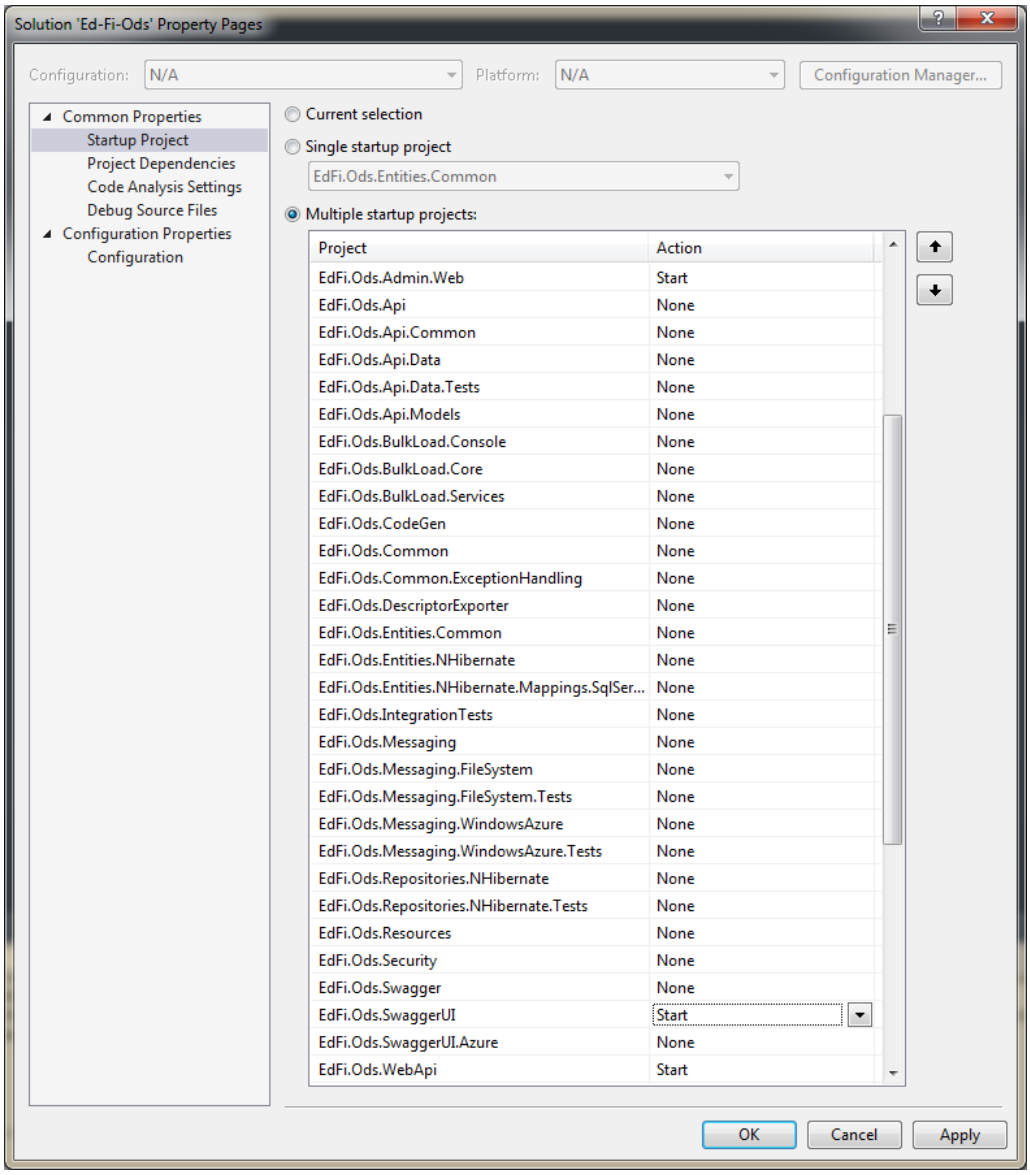
To set the “Startup Projects:

1. Select the “Set Startup Projects…” context menu by right-clicking on the solution file in the Solution Explorer.



2. Within the Startup Project property page, select the “Multiple startup projects” radio button (see screenshot below) and enable the following projects:
   - EdFi.Ods.Admin.Web

- EdFi.Ods.SwaggerUI
- EdFi.Ods.WebApi



3. Click the OK button to accept the changes to your local development settings.

Step 7: Run the Ed-Fi ODS API

The projects in the Ed-Fi-ODS-Implementation repository are configured to run a desktop version of Internet Information Server called “IIS Express.” This server is installed with Visual Studio, and facilitates easy debugging with minimal configuration.

After the startup projects are set, you are ready to run or debug the Ed-Fi ODS API.

Press F5 or press the Start button in the Standard Toolbar (see below) to start the Ed-Fi ODS API.

The solution starts each of the projects that were added to the startup projects list. For each web application, it starts an instance of IIS Express.

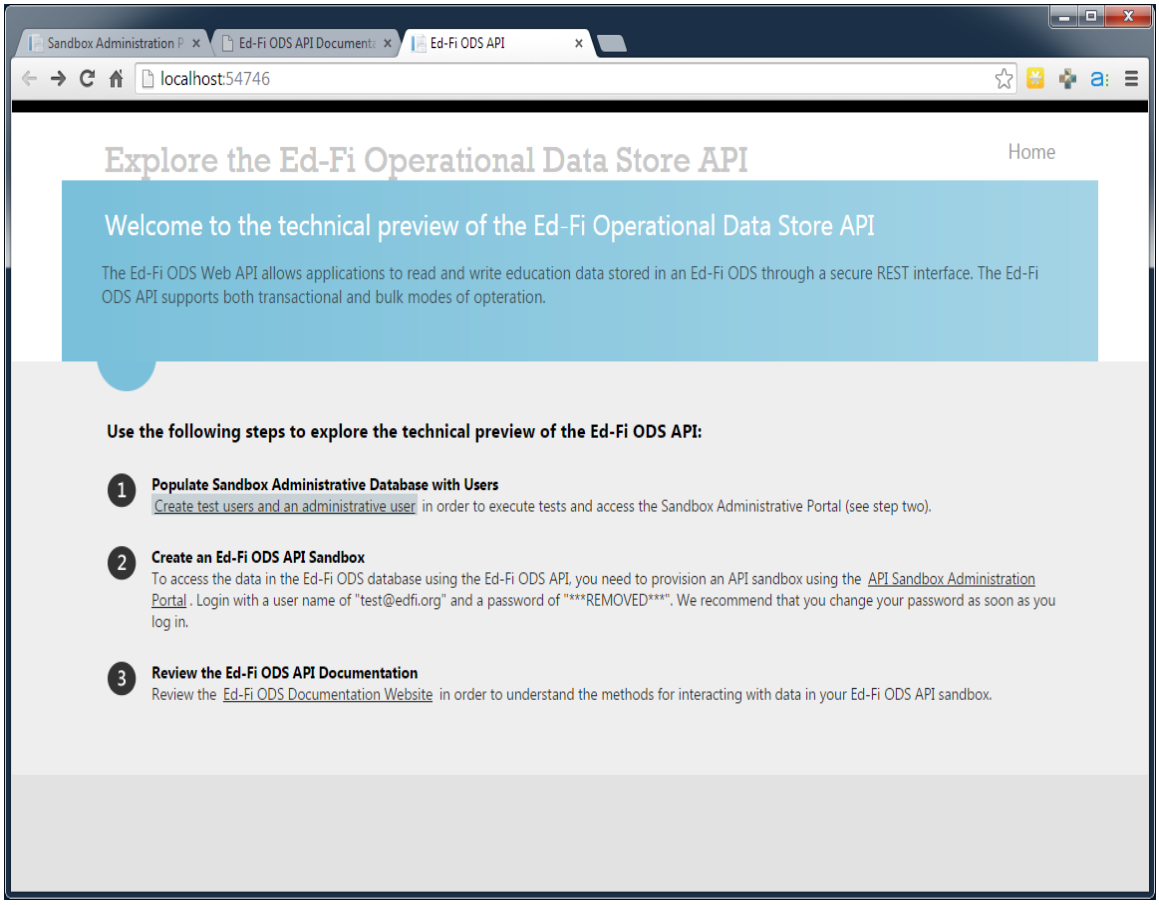
By default, the websites are configured according to the following table:

Website	Project	URL
Ed-Fi ODS API	EdFi.Ods.WebApi	http://localhost:54746/
Sandbox Administration	EdFi.Ods.Admin.Web	http://localhost:38928/
Ed-Fi ODS API Documentation	EdFi.Ods.SwaggerUI	http://localhost:56641/

The Ed-Fi ODS API Home Page

The Ed-Fi ODS API homepage contains the Ed-Fi ODS API. This is not an end-user facing web page and should be deleted from production installations

The following screen welcomes you to the Ed-Fi ODS API:

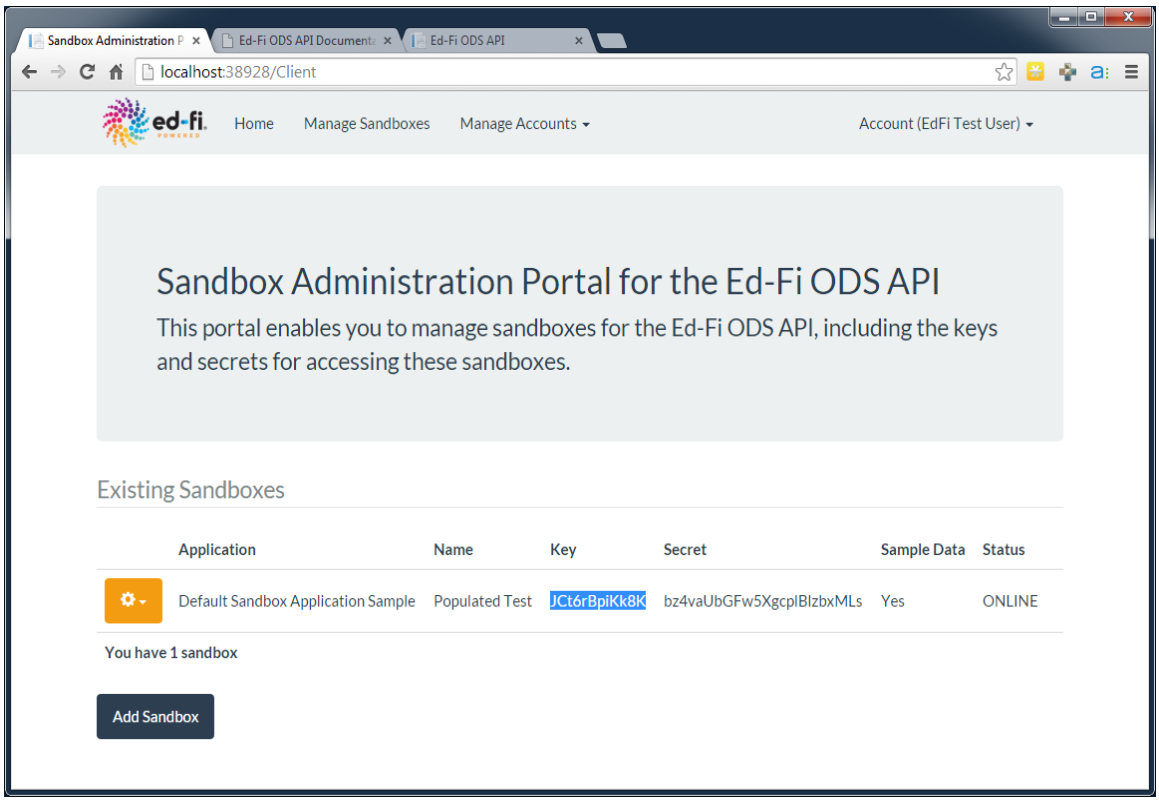


Follow each of the steps on the screen to finish configuring the solution. This involves:

1. Creating an administrative user
2. Creating a sandbox
3. Using the key and secret associated with a sandbox to explore the API

The Sandbox Administration Portal

The Sandbox Administration Portal (see the following screen) is a web application used to create sandbox databases containing data that can be accessed through the Ed-Fi ODS API.

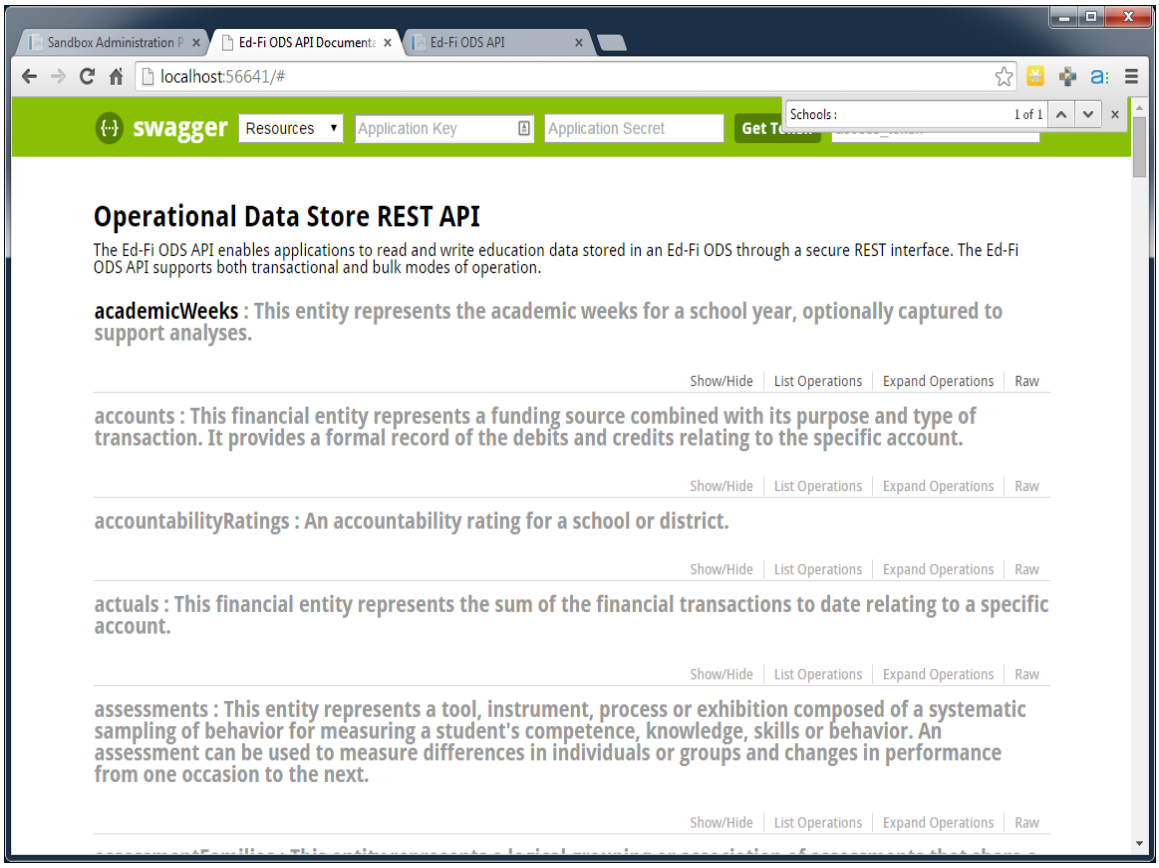


The Ed-Fi ODS API Documentation Web Page

The Ed-Fi ODS API Documentation web page uses a Key and Secret (Sandbox Administration Portal) to access the data that has been placed in the corresponding sandbox. [9]

To view the data in your sandbox, enter the key and secret in the appropriate fields and retrieve a token. This token is used throughout your session to access your sandbox. This is the same process used by other applications to access their data.

[9] The Ed-Fi ODS API Documentation Website uses the Swagger framework (http://swagger.io/).



Allowing Unit Tests

Many of the tests in the solution require the web pages to be running while the tests are running. The default settings of Visual Studio 2013 stop IIS Express when debugging is stopped in Visual Studio. This is a change from previous versions of Visual Studio and prevents successful testing

Allowing the websites to continue running (and, therefore, allow them to be accessed by unit tests), requires turning off the “Enable Edit and Continue” setting.

This can be accomplished globally by selecting Tools | Options | Debugging | Edit and Continue or it can be done for each individual web project by selecting Project Settings | Web.

Alternately, the individual websites may be launched in IIS Express from a command prompt using the following commands:

```
c:\Program Files (x86)\IIS Express\iisexpress.exe /site:EdFi.Ods.Admin.Web

c:\Program Files (x86)\IIS Express\iisexpress.exe
   /site:EdFi.Ods.Admin.SwaggerUI

c:\Program Files (x86)\IIS Express\iisexpress.exe /site:EdFi.Ods.Admin.WebApi
```

The websites will run until they are explicitly stopped using the IIS Express system tray.

For more information regarding using IIS Express from the command line, see http://www.iis.net/learn/extensions/using-iis-express/running-iis-express-from-the-command-line.

Providing Feedback

The Technical Preview release of the Ed-Fi ODS and Ed-Fi ODS API is a work in progress, and you shouldn't be surprised to encounter quirks and bugs, as well as areas where functionality is not fully implemented. Your feedback is important so please submit questions, bugs, and feature requests as issues in the Ed-Fi-ODS repository on GitHub (https://github.com/Ed-Fi-Alliance/Ed-Fi-ODS/issues/new).

Appendix A: Ed-Fi ODS API Databases

The Ed-Fi ODS API uses several databases for various aspects of the application and to store data for each sandbox.

Database	Method	Purpose
EdFi_Ods_Empty	Database Scripts	Empty database used for code generation when building the solution
EdFi_Admin	EF Code First	Database containing sandbox administration information
EduId_Db	Database Project	Database containing person lookup information
EdFi_Ods_Minimal_Template	SQL Scripts	A template database used to create empty sandboxes
EdFi_Ods_Populated_Template	SQL Backup	A template database populated with sample data used to create sample data sandboxes
Rest_Api	EF Code First	Stores bulk upload files and tracking information

In addition to these databases, copies of either the minimal or populated template databases are made for each sandbox in the environment.

Appendix B: Creating a Developer Certificate

A default development certificate is provided. Alternatively, you may create your own developer certificate by issuing the following commands using OpenSSL.

```
openssl req -x509 -nodes -days 9999 -subj '/CN=Development_Encryption'
   -newkey rsa:2048 -keyout Development.pem -out Development.pem

openssl pkcs12 -export -out Development.pfx -in Development.pem -name
   'Development_Encryption'
```

After creating and installing a new certificate, you will also need to generate an encrypted database passwords file containing SQL logins. This may be performed by removing the credentials-*.xml files from the C:\Ed-Fi-ODS-Implementation\logistics\scripts\activities\build directory and running the `initdev` command from PowerShell. This will prompt you for passwords that are then encrypted in a new credentials-*.xml file.

To make the file non-machine specific, remove the name of the machine from the file so you have something like “credentials-Development.xml”. If you wish to use integrated authentication, remove the `<Username>edfiAdmin</Username>` and `<Username>edfiLoader</Username>` values from the XML. Do not change any other sections of the XML documents.

Appendix C: Windows Azure

Windows Azure is NOT required for the Technology Preview Release. Azure deployment and infrastructure is a deployment option. If Azure deployment is desired, the Azure SDK for .NET is required to perform local testing and cloud deployment from a developer workstation.

There are a few components that have been designed to work in the Azure environment. These components utilize Azure Queues and Blob Storage. The following table lists the Windows components as well as their Azure counterparts:

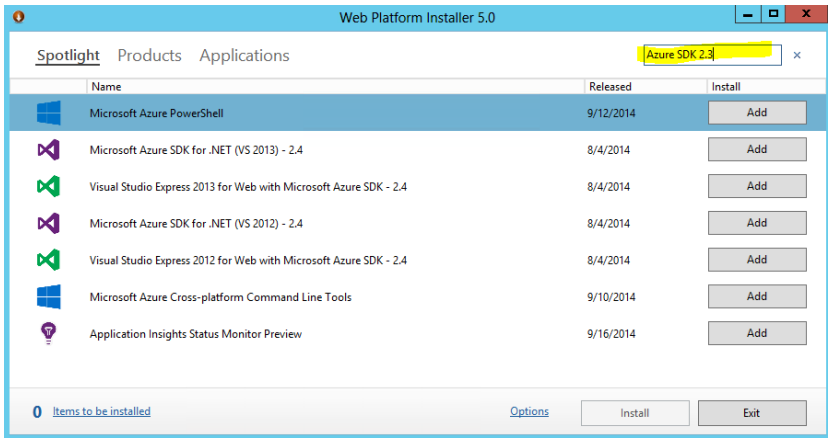
Windows Environment	Azure Environment	Purpose
EdFi.MSMQServices.SendOnly	EdFi.Workers.BulkLoad	Azure Worker Role that performs asynchronous loading of bulk XML.
EdFi.MSMQServices.ListenAndSend	EdFi.Workers.UploadCommit	Azure Worker Role that combines uploaded bulk XML file segments.

Windows Azure SDK Installation

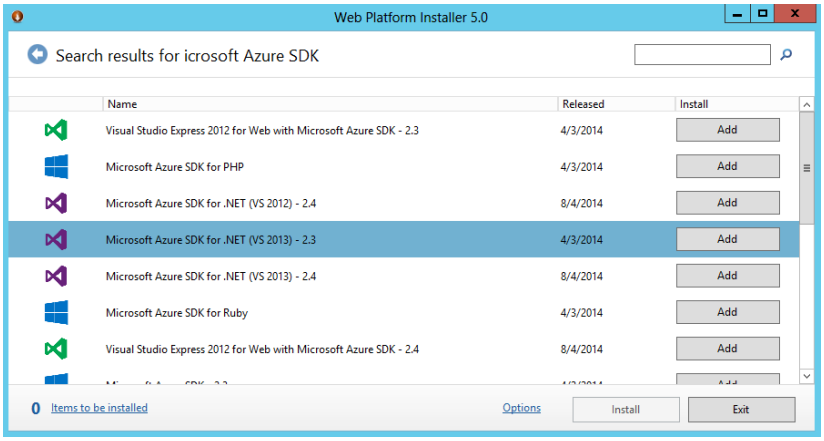
The Windows Azure SDK is used to simulate the Azure environment in a non-production environment (i.e. development machine) and includes local implementations of Azure components as well as tools for deploying to the Azure environment from Visual Studio 2013. The SDK is not required to compile the Technology Preview Release; NuGet packages reference the correct version of Azure libraries for development purposes.

The Windows Azure SDK may be installed using the following steps:

1. Install and run the Microsoft Web Platform Installer v5.0 (http://www.microsoft.com/web/downloads/platform.aspx).
2. Search for "Microsoft Azure SDK" in the search box



3. Select and install "Microsoft Azure SDK for .NET (VS 2013) – 2.3."[10]



[10] Note: Windows Azure SDK for .NET v2.4 is not compatible with the technology preview.