# OFFICE AMS: Workflow

|  |  |
| --- | --- |
| Summary: | Applies to: |
| This sample shows how to create a workflow that calls a custom web service that updates SharePoint list data. | * Office 365 Multi-Tenant (MT) |
| Solution: | Workflow.CallCustomService, version 1.0 |
| Author: | Todd Baginski (Canviz LLC)  Tyler Lu (Canviz LLC)  Romy Ji (Canviz LLC) |
| //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // THIS CODE IS PROVIDED \*AS IS\* WITHOUT WARRANTY OF  // ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING ANY  // IMPLIED WARRANTIES OF FITNESS FOR A PARTICULAR  // PURPOSE, MERCHANTABILITY, OR NON-INFRINGEMENT.  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | |

# Scenario: Call custom web service

This provider-hosted sample application for SharePoint demonstrates how to create a workflow that calls a custom web service that updates SharePoint list data.

## Web API Service

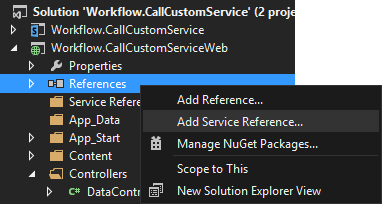
In this code sample, we use *DataController* controller. It contains a method called *Post* which is used to handle the workflow’s http post request.

*Post* method calls the [Northwind OData service](http://services.odata.org/V3/Northwind/Northwind.svc) to get the supplier names of the specified country in the list item the workflow is invoked upon. Then, it writes the supplier names back to the *Suppliers* column in the *Part Suppliers* SharePoint list in the app-web.

|  |
| --- |
| **public** **class** DataController : ApiController  {  **public** **void** Post([FromBody]**string** country)  {  //...  }  } |

### Call Northwind OData Service

To call the Northwind OData Service, a Service Reference has been added to the web service. To add a Service Reference to a provider-hosted web project, in the Solution Explorer, right click the References node in the provider-hosted web project then click Add Service Reference.



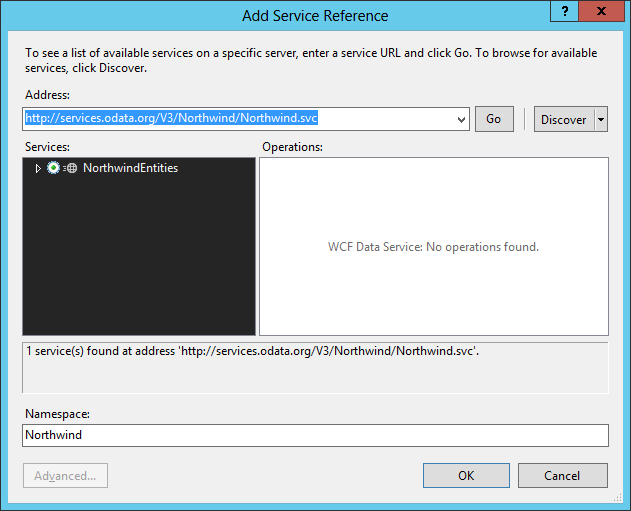
Then, in the Add Service Reference dialog, type the address of the service you wish to reference. In this example, the URL is:

<http://services.odata.org/V3/Northwind/Northwind.svc>.

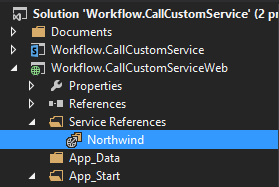
In this sample, the Namespace is Northwind.

After the URL and Namespace are added, click Go. At this point, Visual Studio reads the OData metadata document to discover the entities in the service.

Finally, click OK to add the proxy class to your project.



Here you can see the Northwind OData Service after it has been added to the provider-hosted web project.



After the service reference has been added, we can use LINQ to get the supplier names for a specified country. This is shown in the code snippet below.

|  |
| --- |
| **using** Workflow.CallCustomServiceWeb.Northwind;  // DataController  **private** **string**[] GetSupplierNames(**string** country)  {  Uri uri = **new** Uri("http://services.odata.org/V3/Northwind/Northwind.svc");  var entities = **new** NorthwindEntities(uri);  var names = entities.Suppliers  .Where(s => s.Country == country)  .AsEnumerable()  .Select(s => s.CompanyName)  .ToArray();  **return** names;  } |

### Update SharePoint List Item

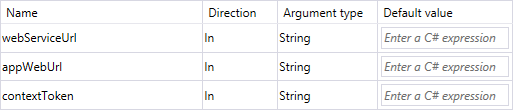
In order to connect to SharePoint and update a list item, the Web API needs a *context token*. The *context token* and *app web URL* will be sent to the web API via http header by the workflow.

|  |
| --- |
| // DataController  **private** **void** UpdateSuppliers(**string** country, **string**[] supplierNames)  {  var request = HttpContext.Current.Request;  var authority = request.Url.Authority;  var spAppWebUrl = request.Headers["SPAppWebUrl"];  var contextToken = request.Headers["SPContextToken"];  **using** (var clientContext = TokenHelper.GetClientContextWithContextToken(  spAppWebUrl, contextToken, authority))  {  var service = **new** PartSuppliersService(clientContext);  service.UpdateSuppliers(country, supplierNames);  }  } |

## Workflow

### Workflow Arguments

The workflow needs to send the *context token* *and app web URL* to the web API. To send these values to the Web API these two values are passed to the workflow when it starts. The *web API’s URL* is also passedto the workflow upon startup. Three arguments are created in the workflow to receive these values.



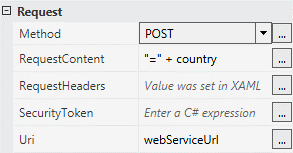
### Call Web API Service

To call the Web API., the *HttpSend* activity is used.

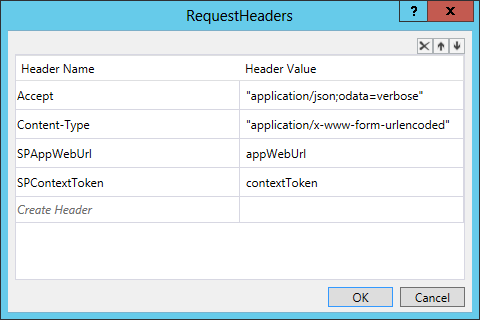
The *HttpSend* activity’s Uri is set to the webServiceUrl variable which is passed to the workflow on startup.



The Method is set to POST; the ReqestContent is set to “=” + country. The country is a variable obtained from the current list item the workflow is interacting with.



The RequestHeaders are set to pass the appWebUrl and contextToken. These are the variables that were initially passed into the workflow startup method. Other header values are set to facilitate the request.



### Start the Workflow

In *PartSuppliersController*, the *app web URL*, *web service URL* and *context token* variables are packaged into another variable named payload.

|  |
| --- |
| // PartSuppliersController  [HttpPost]  [SharePointContextFilter]  **public** ActionResult StartWorkflow(**int** id, Guid workflowSubscriptionId, **string** spHostUrl)  {  var spContext = SharePointContextProvider.Current  .GetSharePointContext(HttpContext) **as** SharePointAcsContext;  var webServiceUrl = Url.RouteUrl(  "DefaultApi",  **new** { httproute = "", controller = "Data" },  Request.Url.Scheme);  var payload = **new** Dictionary<**string**, **object**>  {  { "appWebUrl", spContext.SPAppWebUrl.ToString() },  { "webServiceUrl", webServiceUrl },  { "contextToken", spContext.ContextToken }  };  **using** (var clientContext = spContext.CreateUserClientContextForSPAppWeb())  {  var service = **new** PartSuppliersService(clientContext);  service.StartWorkflow(workflowSubscriptionId, id, payload);  }  //…  } |

The payload variable is passed to the *PartSuppliersService.StartWorkflow* method.

In the *PartSuppliersService*, the workflow is started with the payload variable. The 3 values in the payload variable are passed to the StartWorkflowOnListItem method.

|  |
| --- |
| // PartSuppliersService  **public** **void** StartWorkflow(  Guid subscriptionId, **int** itemId, Dictionary<**string**, **object**> payload)  {  var workflowServicesManager =  **new** WorkflowServicesManager(clientContext, clientContext.Web);  var subscriptionService =  workflowServicesManager.GetWorkflowSubscriptionService();  var subscription = subscriptionService.GetSubscription(subscriptionId);  var instanceService = workflowServicesManager.GetWorkflowInstanceService();  instanceService.StartWorkflowOnListItem(subscription, itemId, payload);  clientContext.ExecuteQuery();  } |

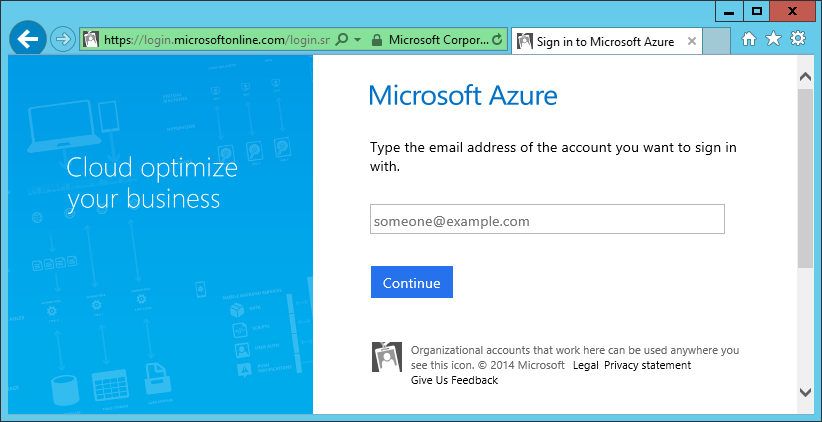
# Deployment Guide

## Deploy the provider hosted web site

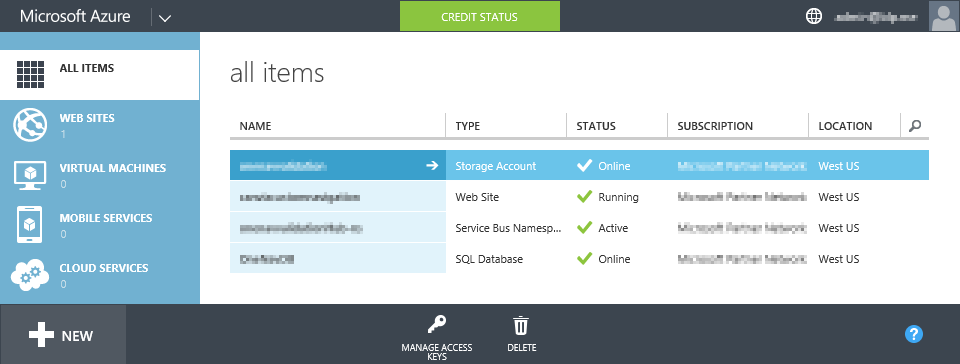
### Create a web site in windows azure

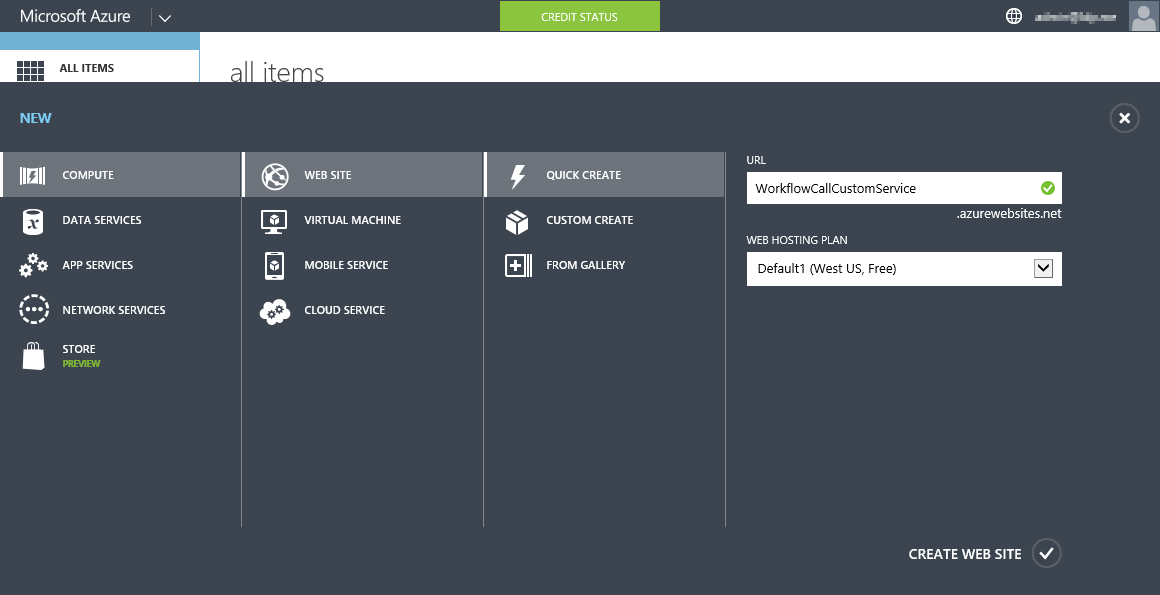
Open <https://manage.windowsazure.com>.

Login to you windows azure account.



Click +New at the bottom left.





Click Computer, click WEB SITE, click QUICK CREATE, and input a URL.

Here, we input WorkflowCallCustomService, and the web site will be created at the following URL:

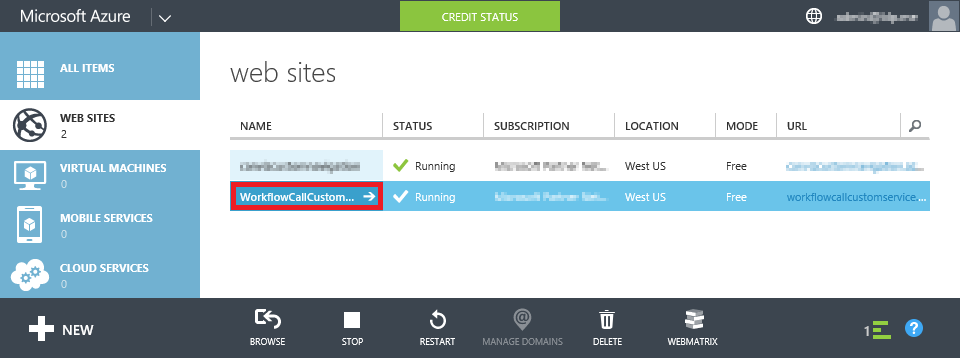
WorkflowCallCustomService.azurewebsites.net

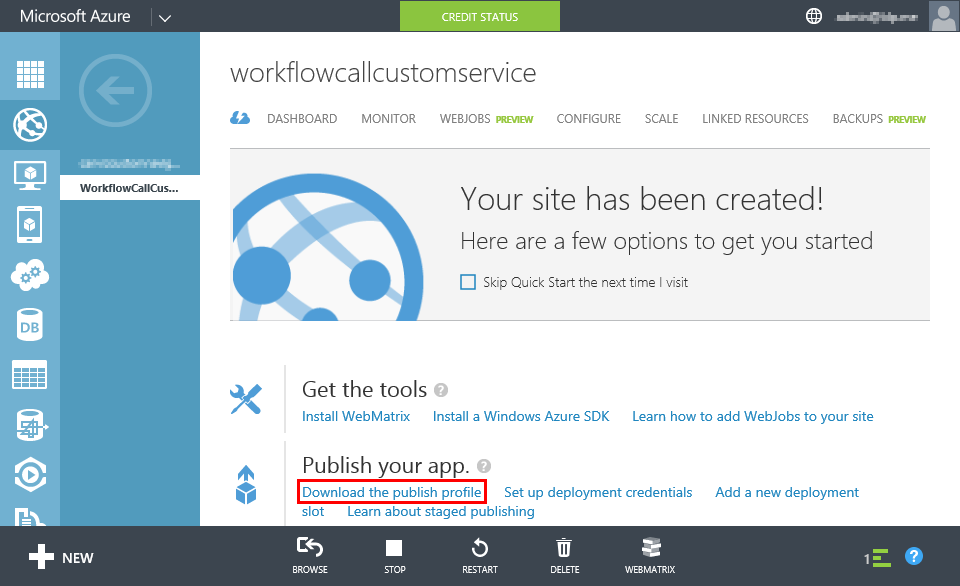
You will need to input a different url. Please remember the url. You are going to use it later.

Click CREATE WEB SITE at the bottom right.

Wait for a while, the new web site will be created.

Click the name of the web site.





Click Download the publish profile under the Publish your app.



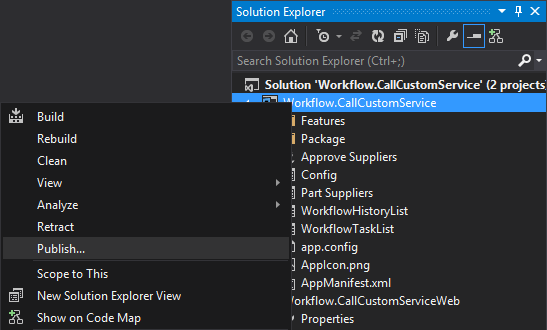
Save the file.

### Publish app web site

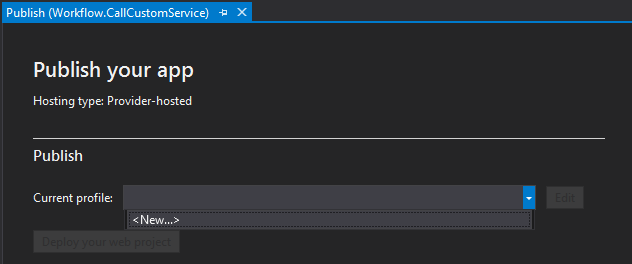
Open the *Workflow.CallCustomService*.sln file with Visual Studio 2013.

In Solution Explorer, right click the *Workflow.CallCustomService* project.

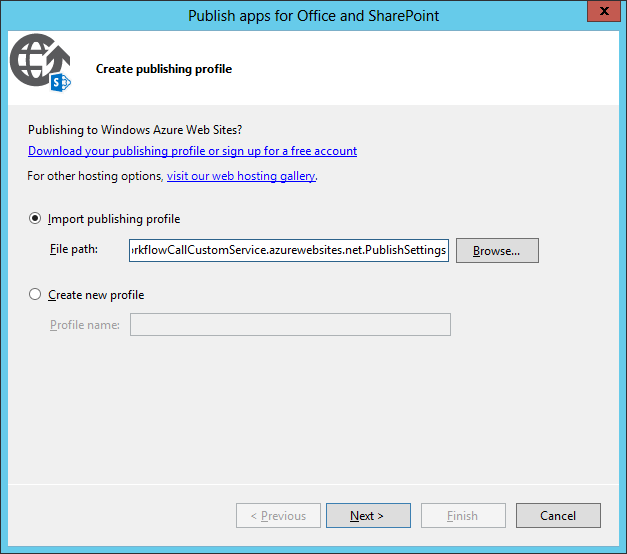
Click Publish…



Click the drop down button, then click <New…>



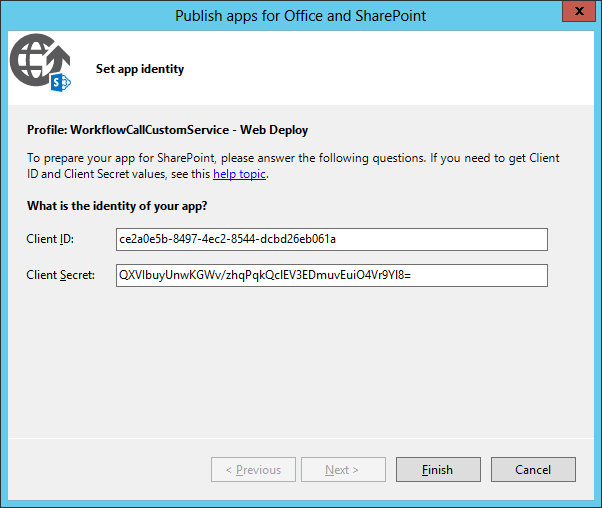
Select Import publishing profile, then click Browse.... Choose the publish settings file you previously downloaded. Click Next.



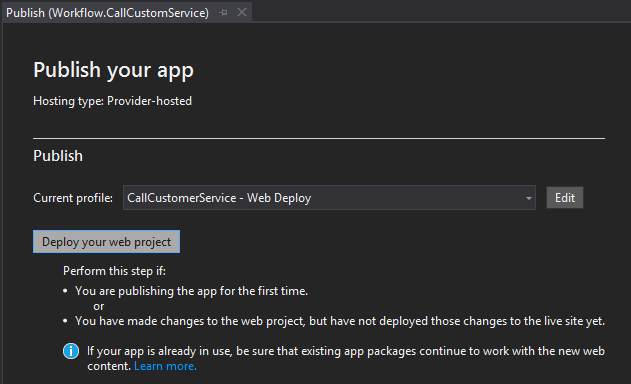
Input the Client ID and Client Secret shown below:

* Client Id: ce2a0e5b-8497-4ec2-8544-dcbd26eb061a
* Client Secret: QXVIbuyUnwKGWv/zhqPqkQclEV3EDmuvEuiO4Vr9Yl8=

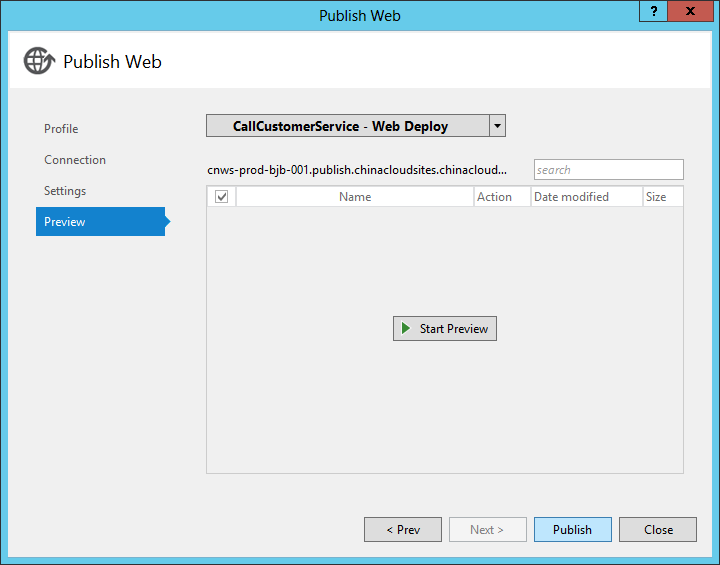
Click Finish.

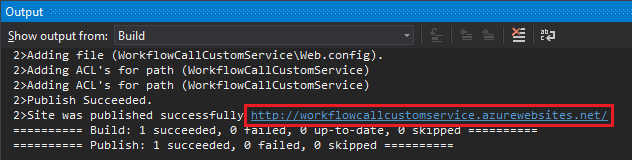


Click Deploy your web project.



Click Publish.



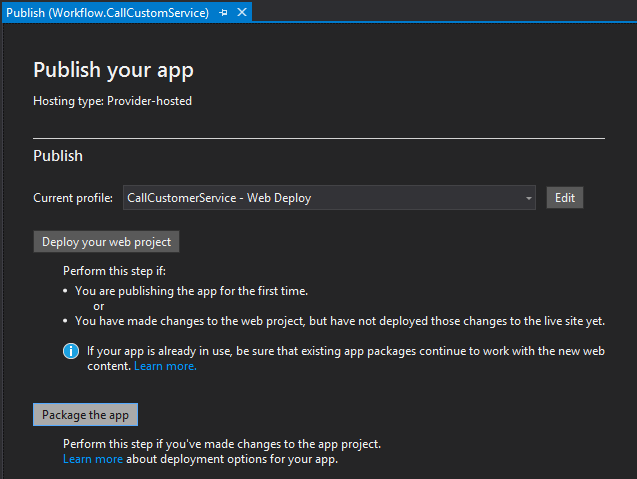


In a few minutes, the site will be published to Windows Azure.

## Deploy the app

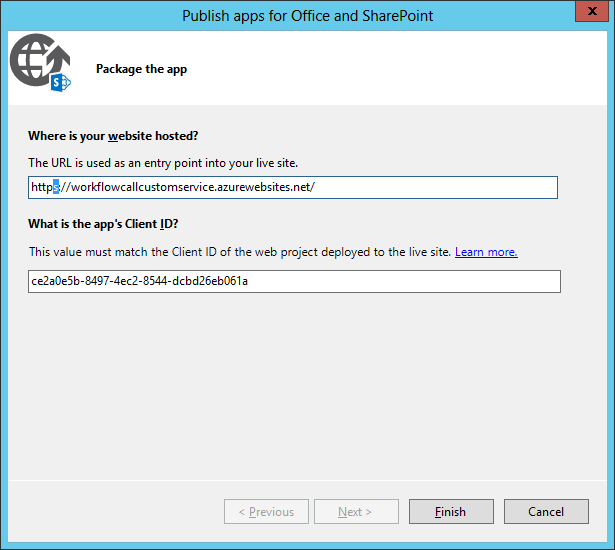
### Package the app

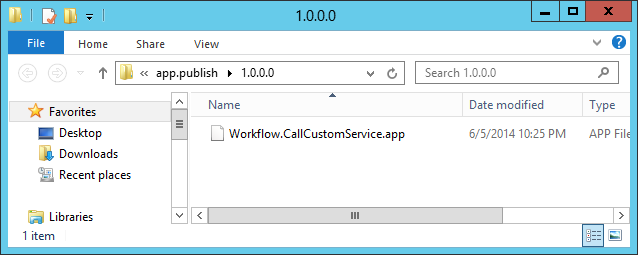
Click Package the app.



Modify the URL, add the letter ‘**s’** after ‘http’.

Click Finish. A Windows Explorer window will pop up and display the .app file you just generated.





### Register the app

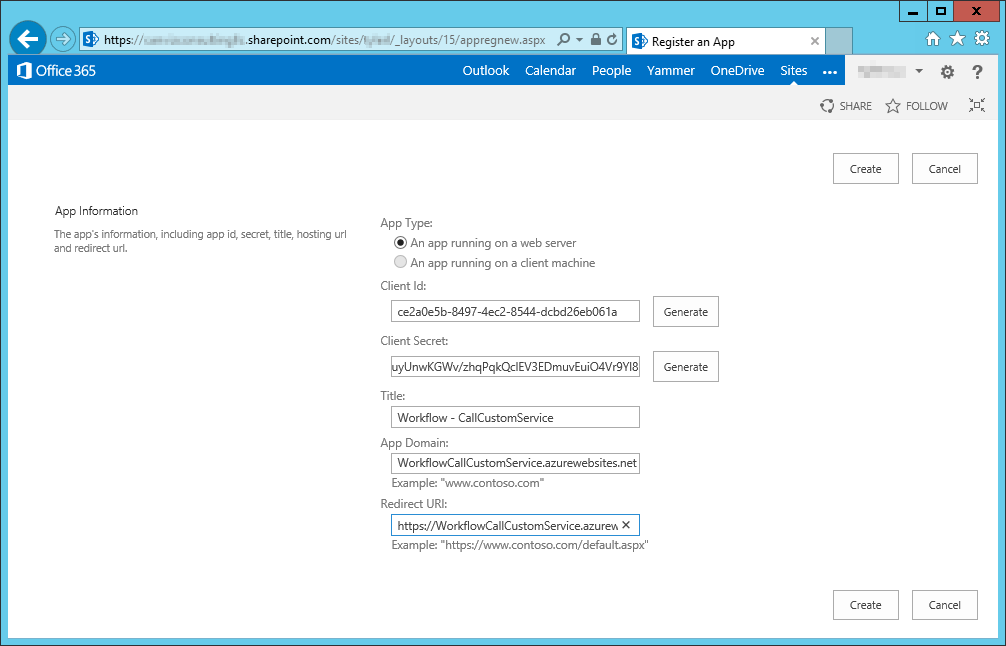
Login to the O365 site where you want to install the app.

Change the Url to:

https://<tenancy>.sharepoint.com/sites/<site>/\_layouts/15/**appregnew.aspx**

Replace the <tenancy placeholder in the URL with your tenancy name.

Replace the <site> placeholder with your site collection name.

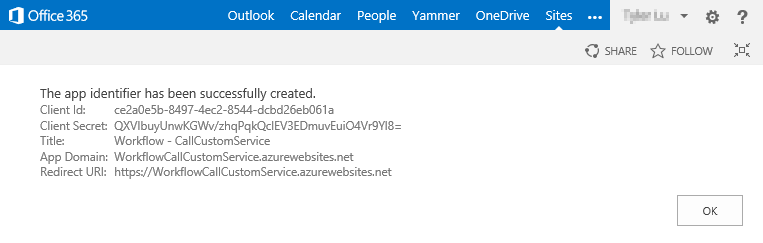


In this step, you should use the domain of you Windows Azure web site, and add prefix “https” as the Redirect URL.

Input these values in the form:

* ClientId: ce2a0e5b-8497-4ec2-8544-dcbd26eb061a
* ClientSecret: QXVIbuyUnwKGWv/zhqPqkQclEV3EDmuvEuiO4Vr9Yl8=
* Title: Workflow.CallCustomService
* AppDomain: ~~WorkflowCallCustomService.azurewebsites.net~~ (Use Your AppDomain)
* Redirect URL: ~~https://WorkflowCallCustomService.azurewebsites.net~~ (Use Your AppDomain)

Then, click Create.

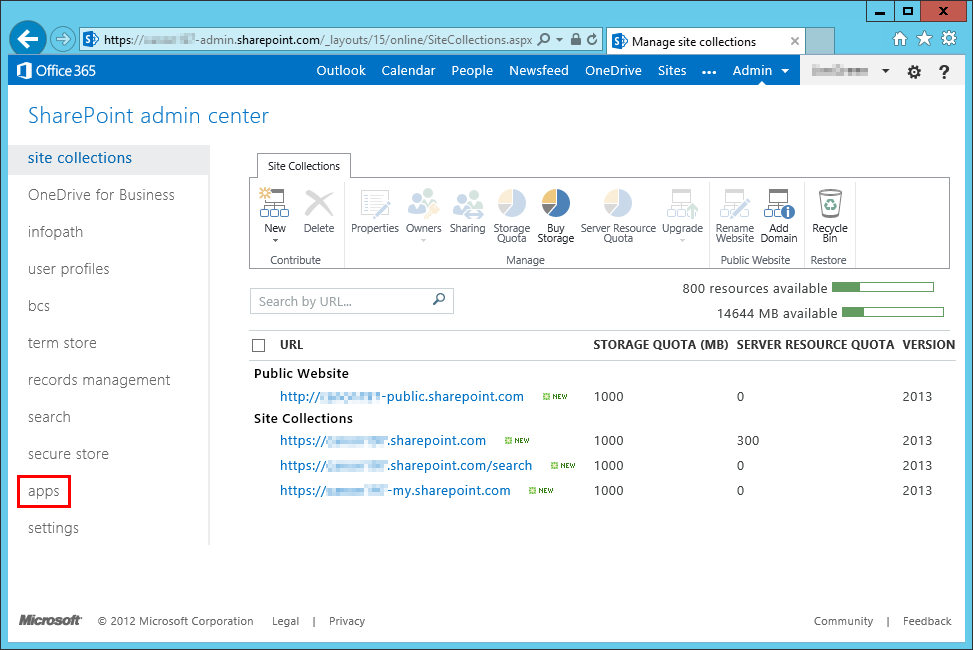


### Create an App Catalog site

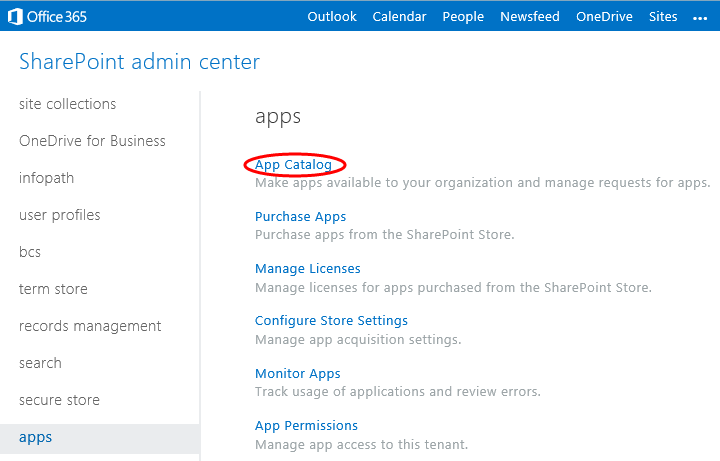
If you don’t have an App Catalog site in your SharePoint Online tenant, you should create one. If there’s already an App Catalog in your tenant, please skip this step.

Sign in to the Office 365 admin center with your SharePoint Online admin user name and password.

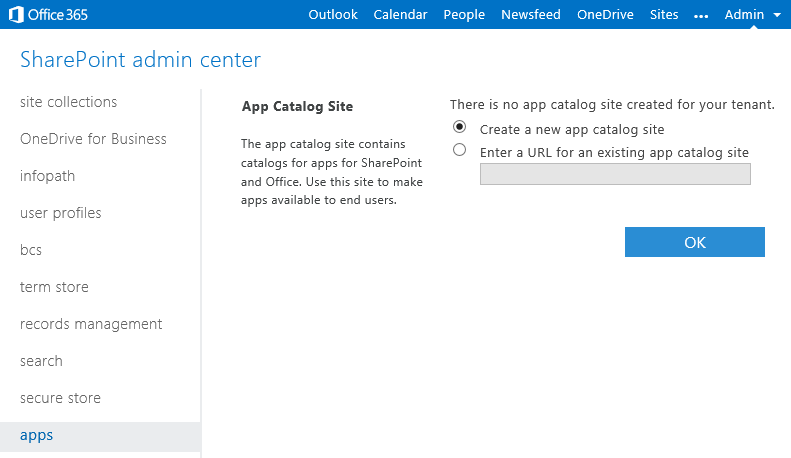
Click apps.



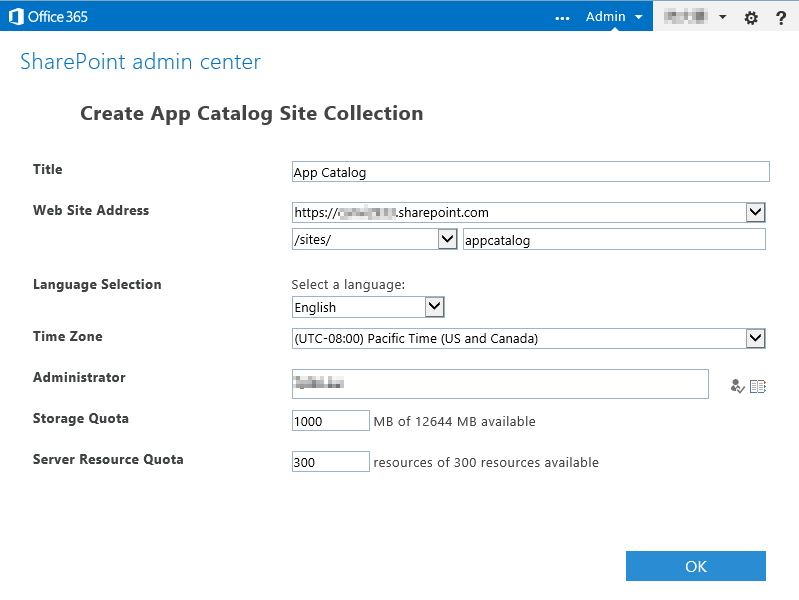
Click App Catalog.



Click OK.



Input the required fields. Click OK.

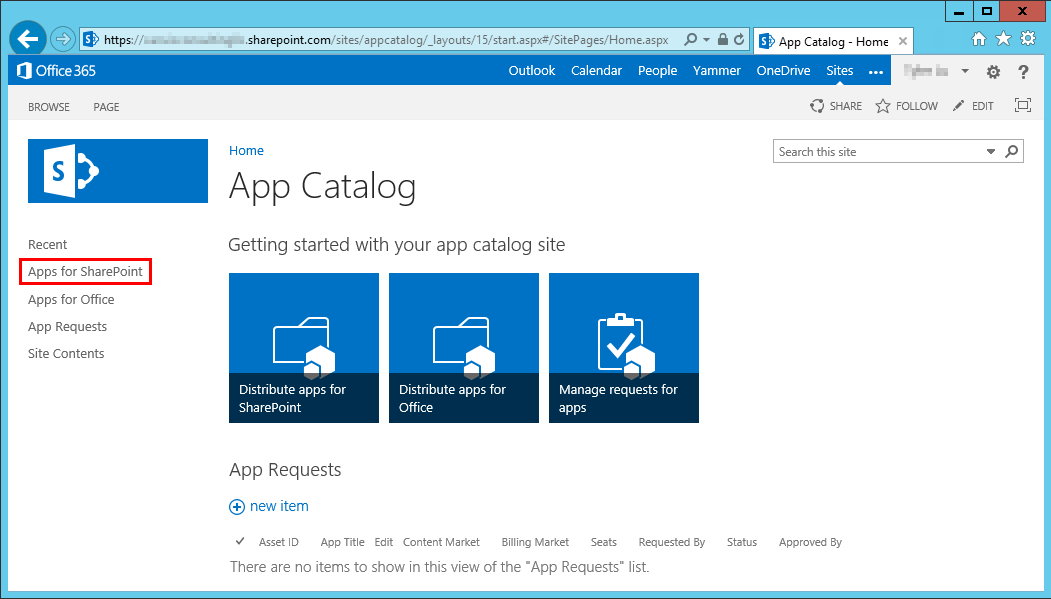


A few minutes later, the App Catalog site will be ready.

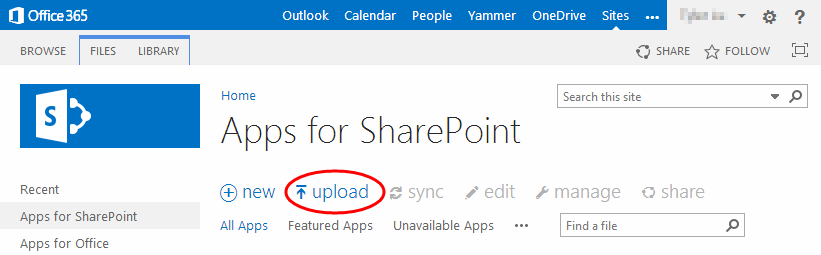
### Upload the app to App Catalog

Login to the App Catalog site.

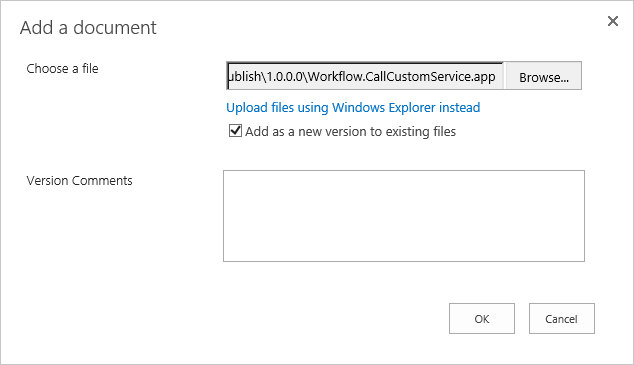
Click Apps for SharePoint.

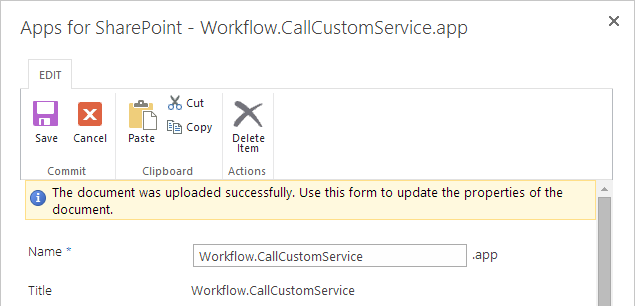


Click upload.



Click Browse…, and choose the .app file you previously created. Then click OK.



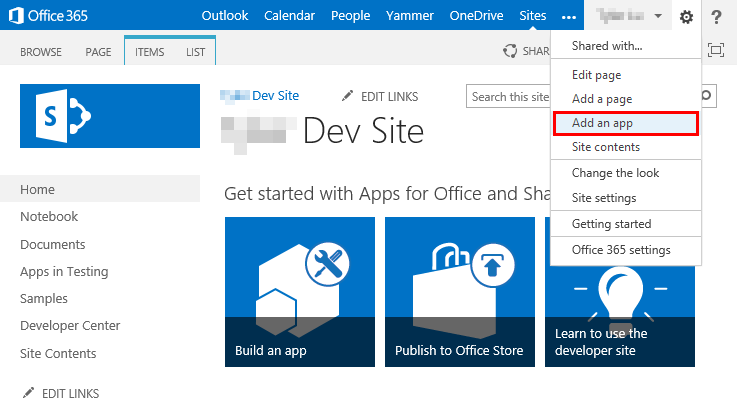


Click Save.

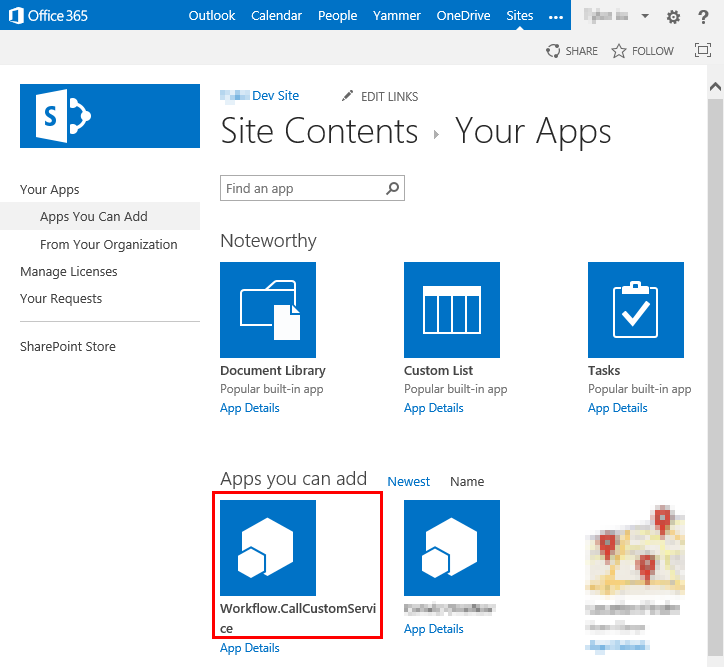
### Install the App

Login to the O365 site where you want to install the app.

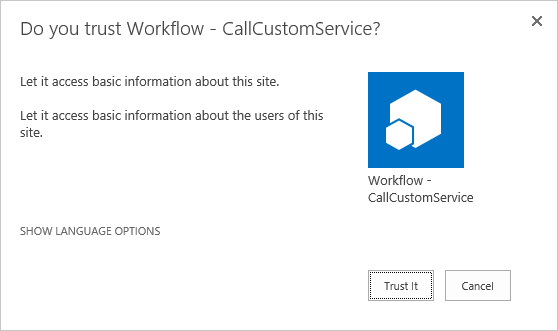
Click  at the top right, then click Add an app.



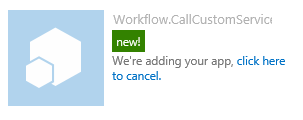
Click Workflow.CallCustomService.



Click Trust It.



The app will be installed in a few minutes.



Once the app is installed, click the app to load it and follow the instructions in the app to run the sample.

