# 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 via a web proxy. | * Office 365 Multi-Tenant (MT) |
| Solution: | Workflow.CallServiceUpdateSPViaProxy, 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 via a web proxy.

## Web API Service

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

The *Post* method updates the *Suppliers* column in the *Part Suppliers* SharePoint list in the app-web.

|  |
| --- |
| **public** **class** DataController : ApiController  {  **public** **void** Post(UpdatePartSupplerModel model)  {  var request = HttpContext.Current.Request;  var authority = request.Url.Authority;  var spAppWebUrl = request.Headers["SPAppWebUrl"];  var accessToken = request.Headers["X-SP-AccessToken"];  **using** (var clientContext = TokenHelper  .GetClientContextWithContextToken(spAppWebUrl, accessToken, authority))  {  var service = **new** PartSuppliersService(clientContext);  service.UpdateSuppliers(  model.Id,  model.Suppliers.Select(s => s.CompanyName));  }  }  } |

In the above method, 2 values are required:

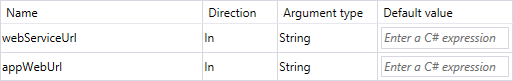
1. App web URL.
2. Access token.

The workflow calls this Web API method via a Web Proxy. The Web Proxy adds the *access token* to the http headers. So we only have to send *app web URL* to the Web API form workflow.

## Workflow

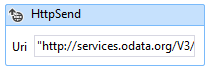
### Workflow Arguments

The workflow needs to send the *app web URL* to the web API. When the workflow starts the *app web URL* is passed to it. The *Web API’s URL* is also passed to the workflow. Two arguments are created in the workflow to receive these values.



### Call Northwind OData Service

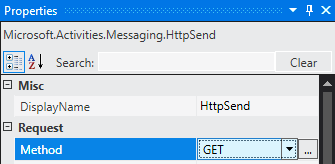
The [Northwind OData Service](http://services.odata.org/V3/Northwind/Northwind.svc/Suppliers) supports anonymous access, therefore it may be called without authenticating.



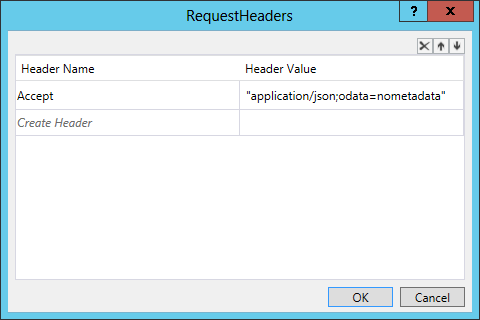
To call the Northwind OData Service, the *HttpSend* activity is used. The Uri is:

|  |
| --- |
| "http://services.odata.org/V3/Northwind/Northwind.svc/Suppliers/?$filter=Country eq '" + country.Replace("'", "''") + "'&$select=CompanyName" |

The *Get* Method is used to return data from the Northwind Odata Service.



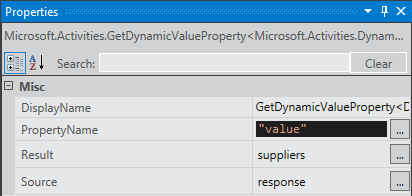
To handle the response, an Accept header is added to the RequestHeaders:



After the call to the Northwind OData Service, the response looks like this:

|  |
| --- |
| {  value: [  {  CompanyName: "Ma Maison"  },  {  CompanyName: "Forêts d'érables"  }  ]  } |

A GetDynamicValueProperty<DynamicValue> Activity is added to the workflow to handle the response. It’s PropertyName to ‘value’.



The suppliers variable value looks like this:

|  |
| --- |
| [  {  CompanyName: "Ma Maison"  },  {  CompanyName: "Forêts d'érables"  }  ] |

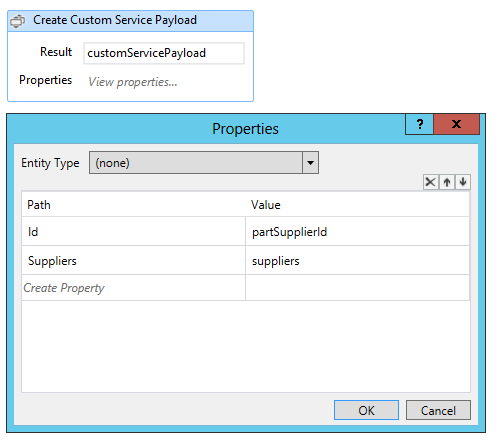
### Call Web API Service via Web Proxy

The Web Proxy’s URL is:

|  |
| --- |
| appWebUrl + "/\_api/SP.WebProxy.invoke" |

#### Create Custom Service Payload

To call the Web Proxy, the workflow uses a BuildDynamicValue Activity to build the payload.

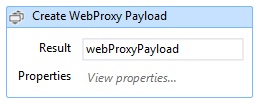


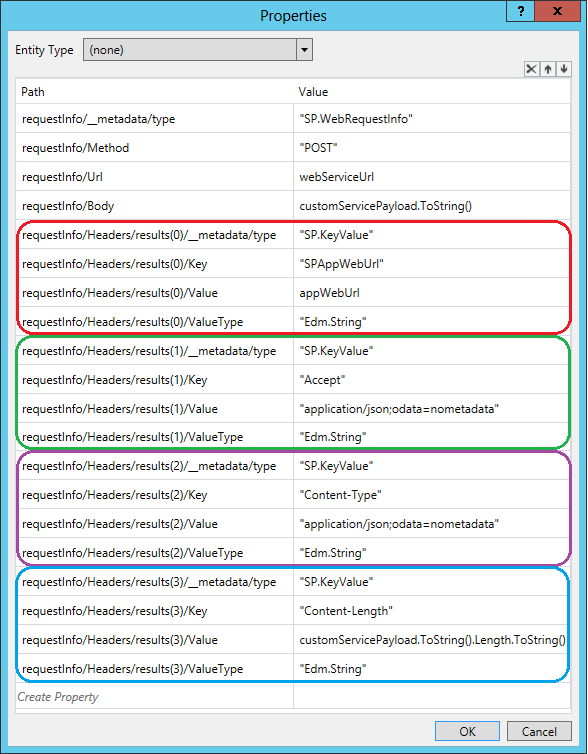
The variable customServicePayload’s value looks like this:

|  |
| --- |
| {  Id: 1  Suppliers: [  {  CompanyName: "Ma Maison"  },  {  CompanyName: "Forêts d'érables"  }  ]  } |

#### Create Web Proxy Payload

A BuildDynamicValue Activity builds the payload.

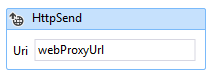




Its value looks like this:

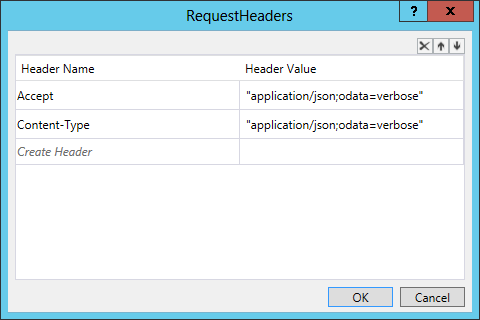
|  |
| --- |
| {  requestInfo: {  \_\_metadata: {  type: "SP.WebRequestInfo"  },  Url: /\* url \*/,  Method: "GET",  Headers: {  results: [  {  \_\_metadata: {  type: "SP.KeyValue"  },  Key: "SPAppWebUrl",  Value: /\* app web url \*/,  ValueType: "Edm.String"  },  {  \_\_metadata: {  type: "SP.KeyValue"  },  Key: "Accept",  Value: "application/json;odata=nometadata",  ValueType: "Edm.String"  },  {  \_\_metadata: {  type: "SP.KeyValue"  },  Key: "Content-Type",  Value: "application/json;odata=nometadata",  ValueType: "Edm.String"  },  {  \_\_metadata: {  type: "SP.KeyValue"  },  Key: "Content-Length",  Value: /\* content length \*/  ValueType: "Edm.String"  }  ]  }  }  } |

#### Call Web API via Web Proxy



Set Method to *POST*. Set ReqestContent to *webProxyPayload.* Set Uri to *webProxyUrl.*

Then set RequestHeaders as below.



### Start the Workflow

In the *PartSuppliersController*, the *app web URL* and *web service URL* are set to a variable named payload.

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

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

In the *PartSuppliersService*, the workflow is started and the payload is passed to it. The 2 values in the payload are passed to the workflow via startup arguments.

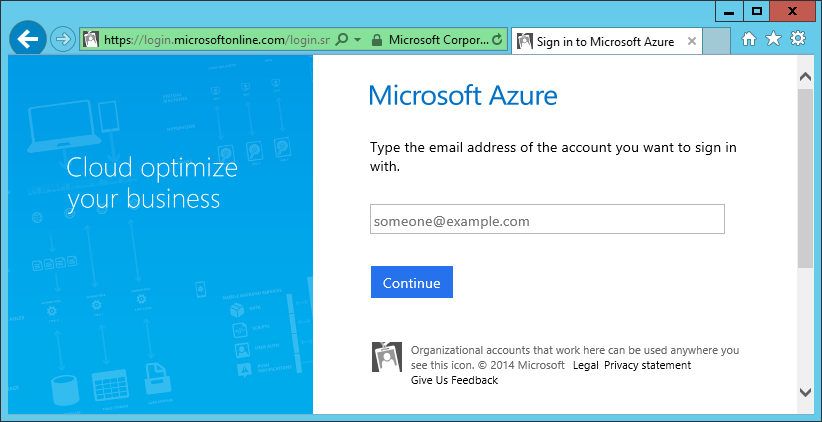
|  |
| --- |
| // PartSuppliersService  **public** **void** StartWorkflow(  Guid subscriptionId, **int** itemId, Dictionary<**string**, **object**> payload)  {  var workflowServicesManager =  **new** WorkflowServicesManager(clientContext, clientContext.Web);  var subscriptionService =  workflowServicesManager.GetWorkflowSubscriptionService();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

# Deployment Guide

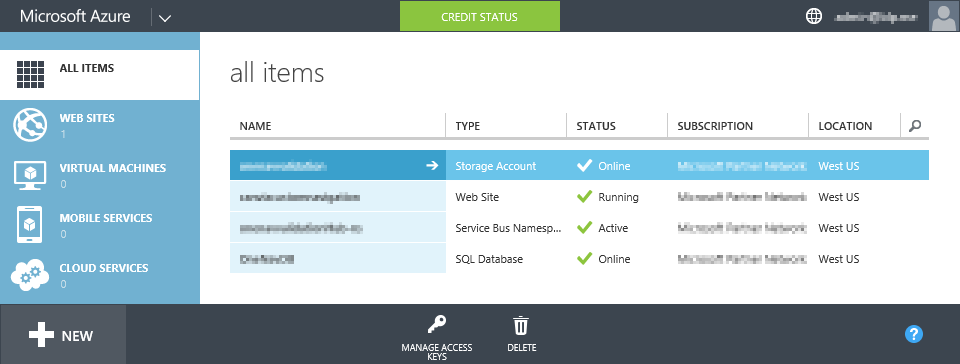
## Deploy the provider hosted web site

### Create a web site in windows azure

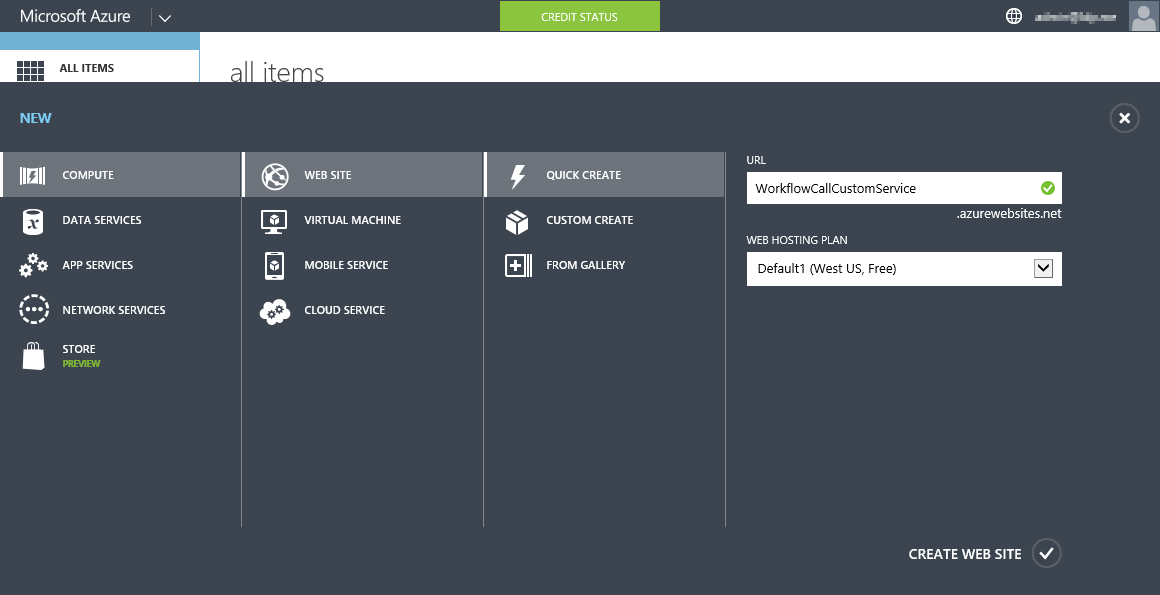
Open <https://manage.windowsazure.com>.



Login in 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:

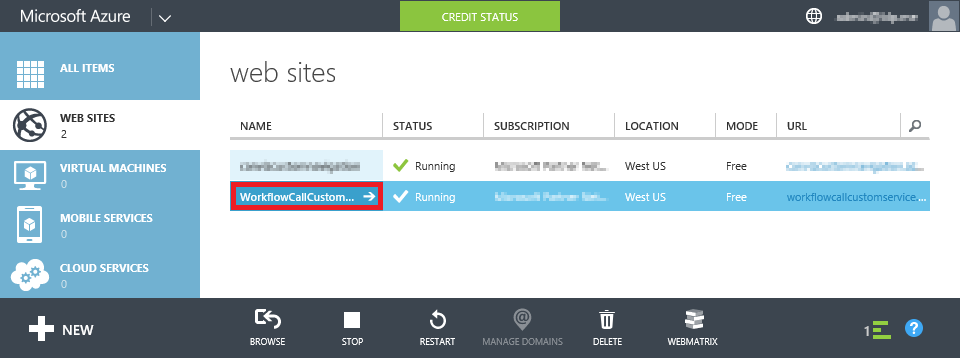
WorkflowCallCustomService.azurewebsites.net

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.

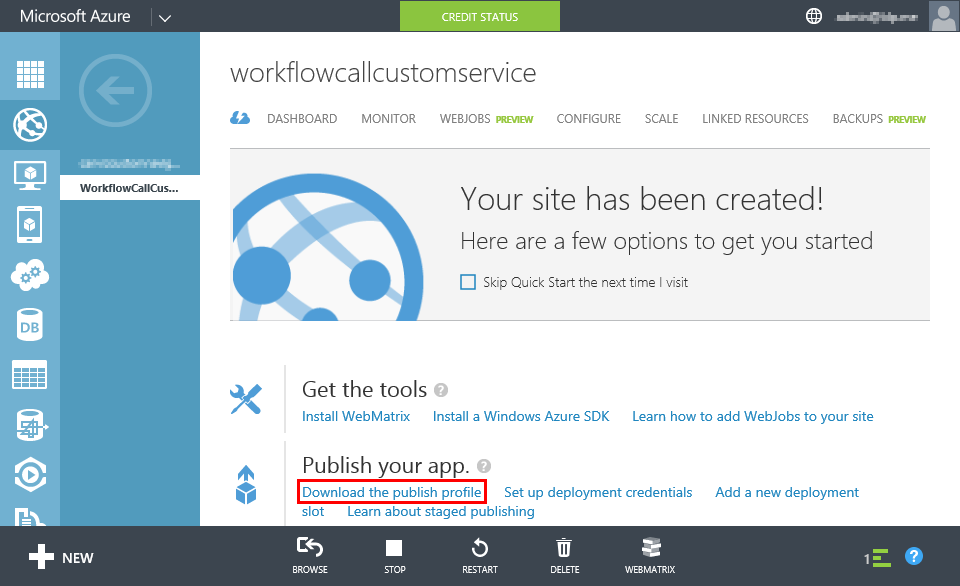
Click the name of the web site.



Click Download the publish profile under the Publish your app.



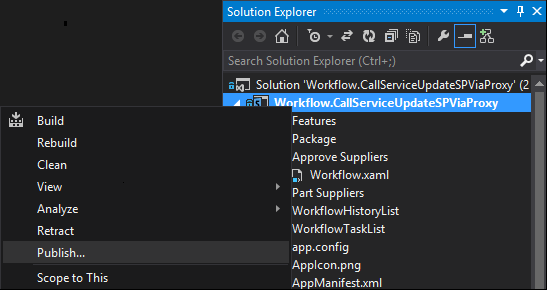
Save the file.



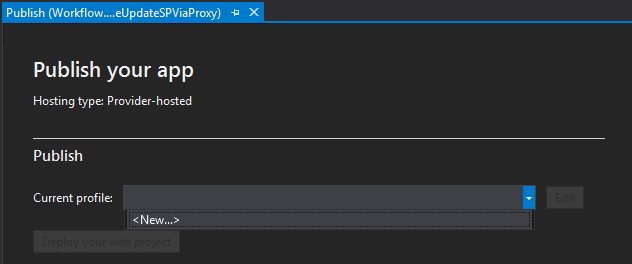
### Publish app web site

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

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

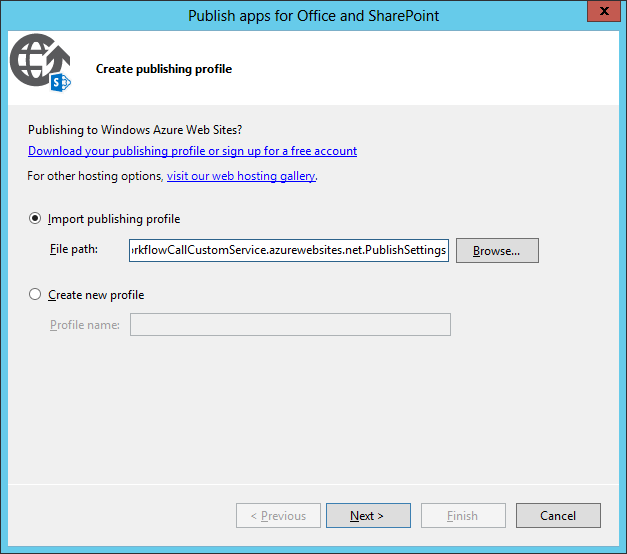


Click Publish…



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

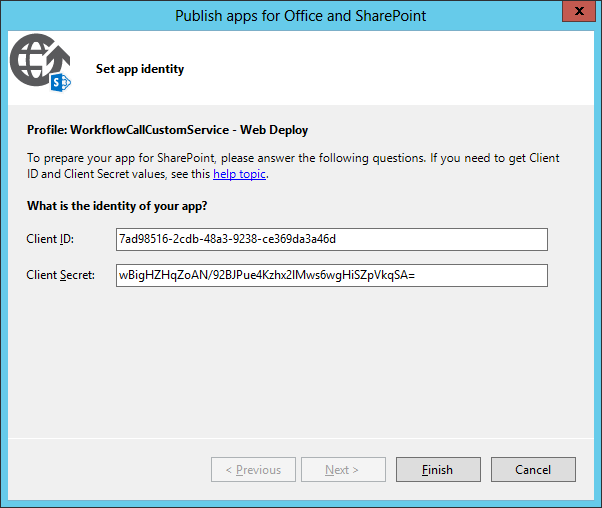
Select Import publishing profile, then click Browse.... Choose the publish settings file you previously downloaded. Click Next.



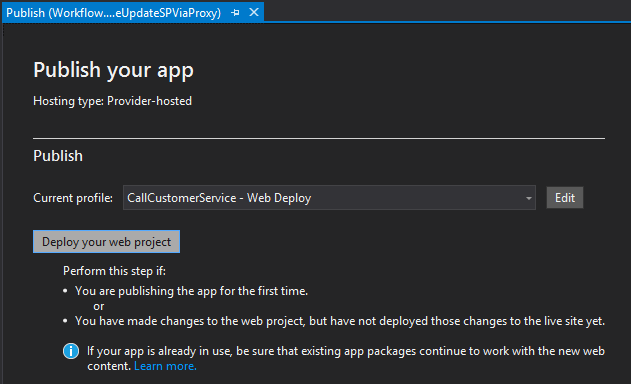
Input the Client ID and Client Secret shown below:

* Client Id: 7ad98516-2cdb-48a3-9238-ce369da3a46d
* Client Secret: wBigHZHqZoAN/92BJPue4Kzhx2lMws6wgHiSZpVkqSA=

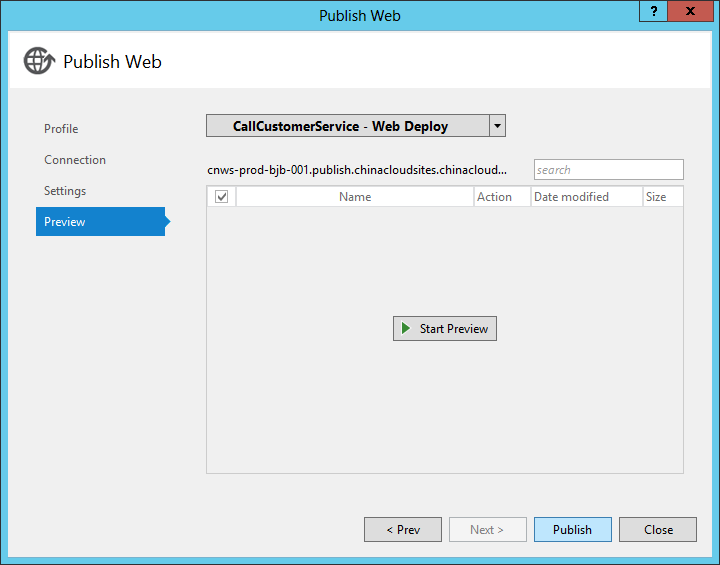
Click Finish.

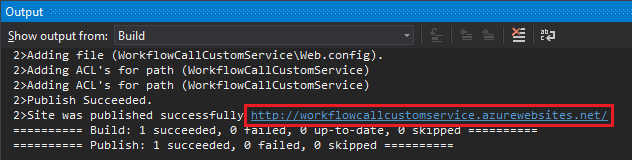


Click Deploy your web project.



Click Publish.

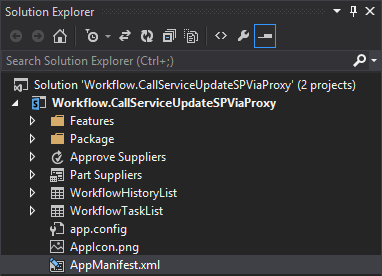




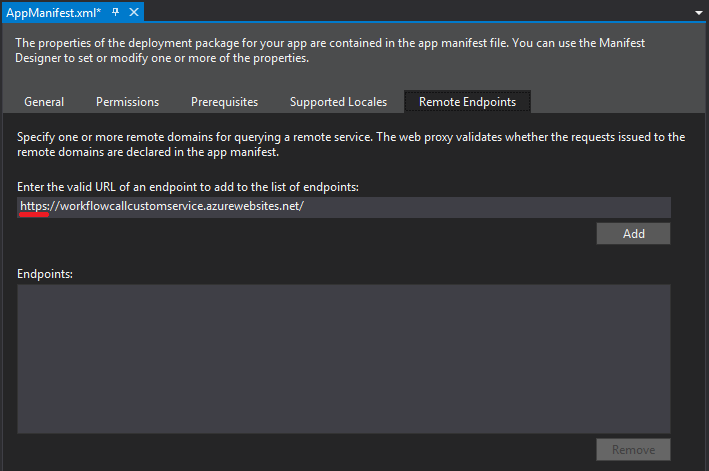
In a few minutes, the site will be published to Windows Azure.

## Deploy the app

### Add Remote Endpoint

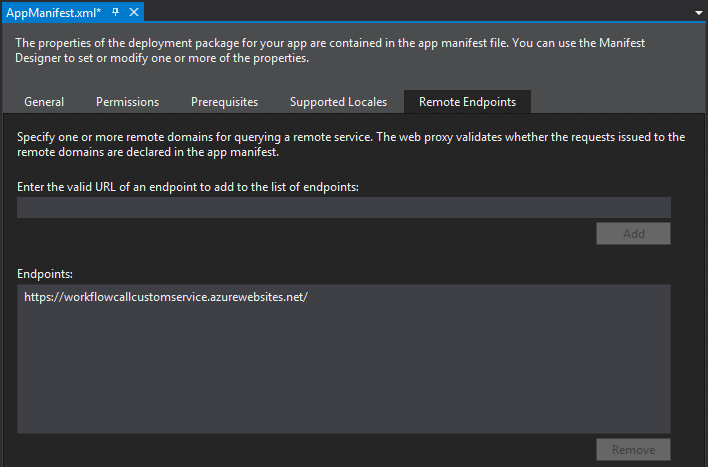


In the Soluction Explorer, click AppManifest.xml.



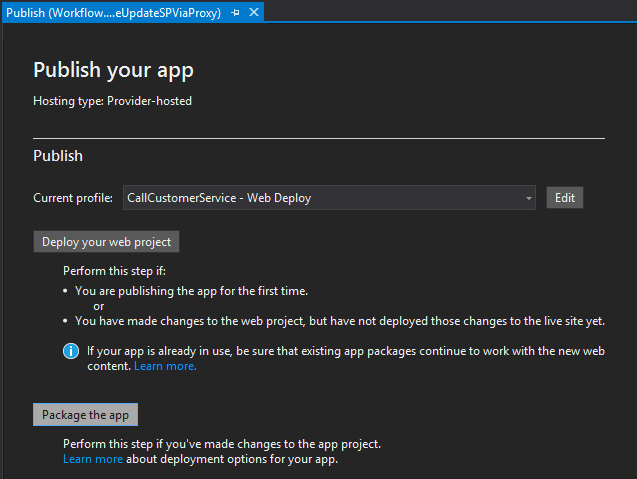
Input the url of your Windows Azure website that you previously created.

Make sure the URL starts with ‘https’. Then click Add.



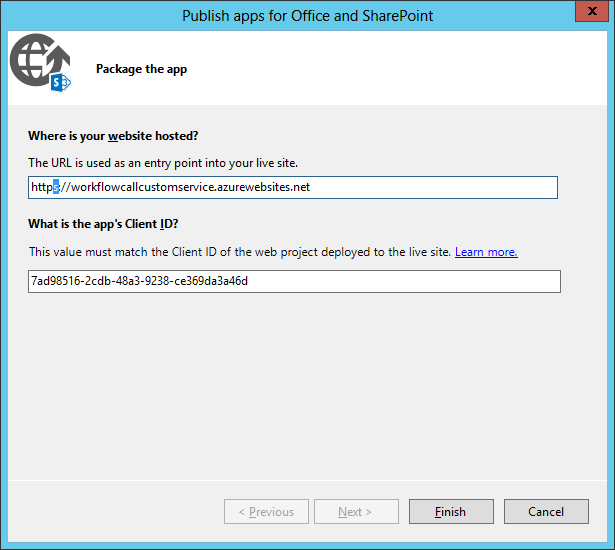
### Package the app

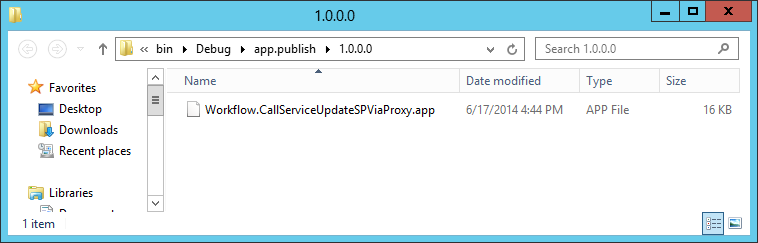
Click Package the app.



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

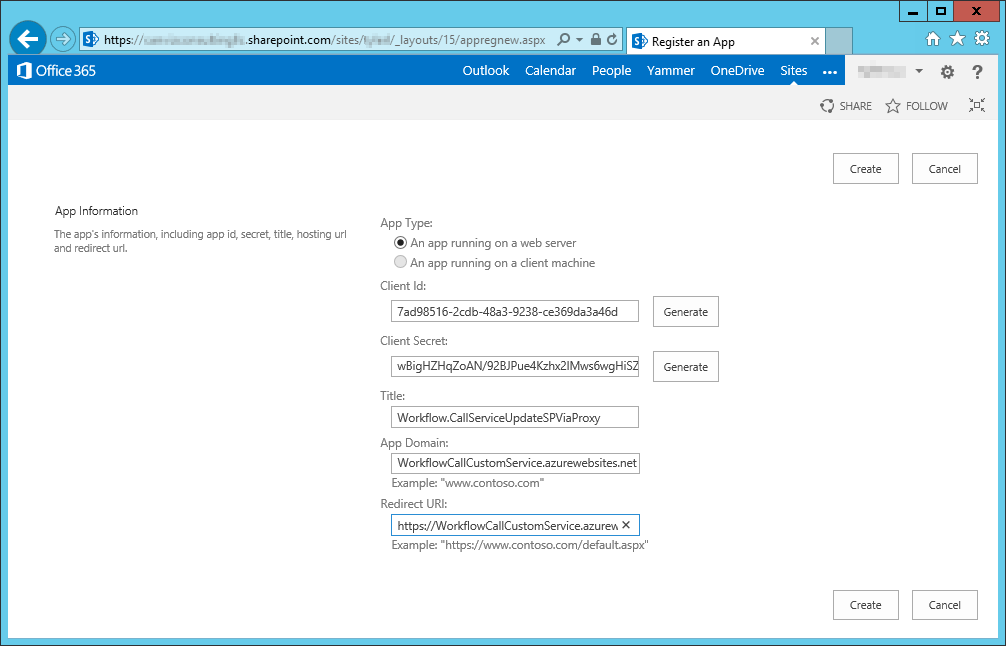
Login to the O365 site where you want to install the app.

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.

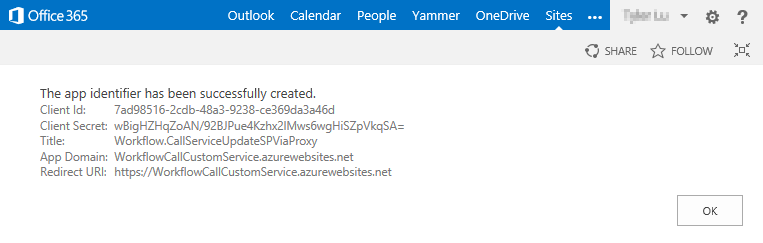


In this step, you should use the domain of your Windows Azure web site, and add the prefix “https” to the Redirect URL.

Input these values in the form:

* ClientId: 7ad98516-2cdb-48a3-9238-ce369da3a46d
* ClientSecret: wBigHZHqZoAN/92BJPue4Kzhx2lMws6wgHiSZpVkqSA=
* Title: Workflow.CallServiceUpdateSPViaProxy
* AppDomain: ~~WorkflowCallCustomService.azurewebsites.net~~ (Use Your AppDomain)
* Redirect URL: ~~https://WorkflowCallCustomService.azurewebsites.net~~ (Use Your AppDomain)

Then, click Create.

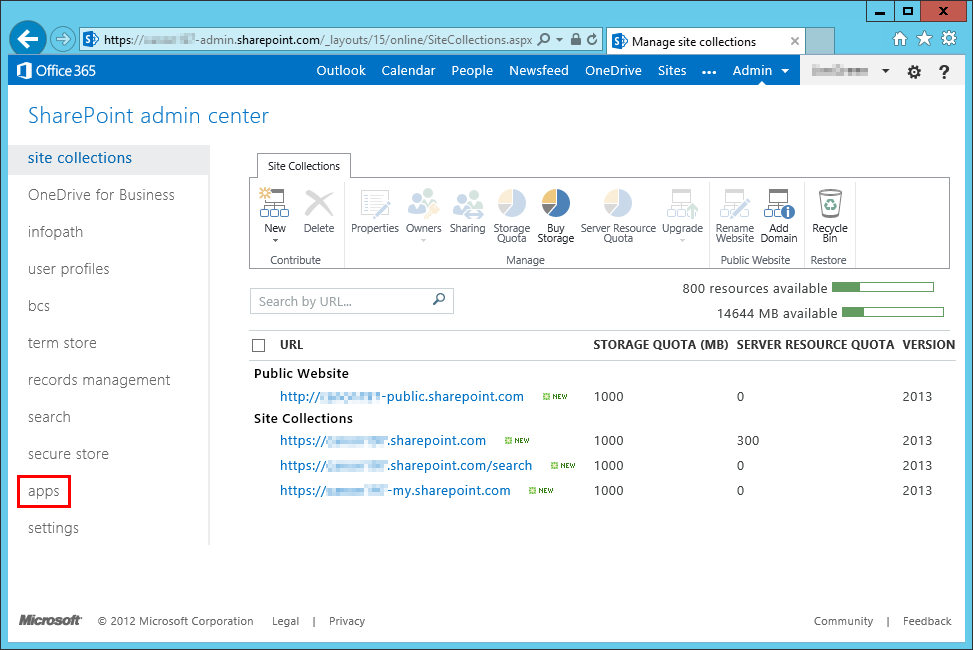


### Create an App Catalog site

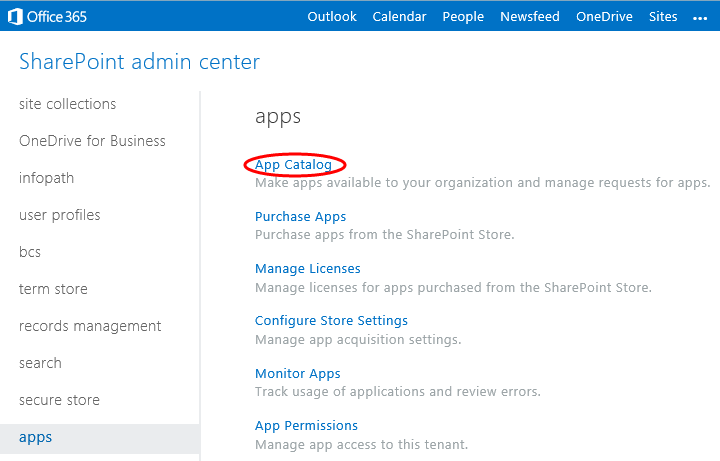
If you don’t have an App Catalog site in your SharePoint Online tenant, you should create one. If there’s already an App Catalog in your tenant, please skip this step.

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

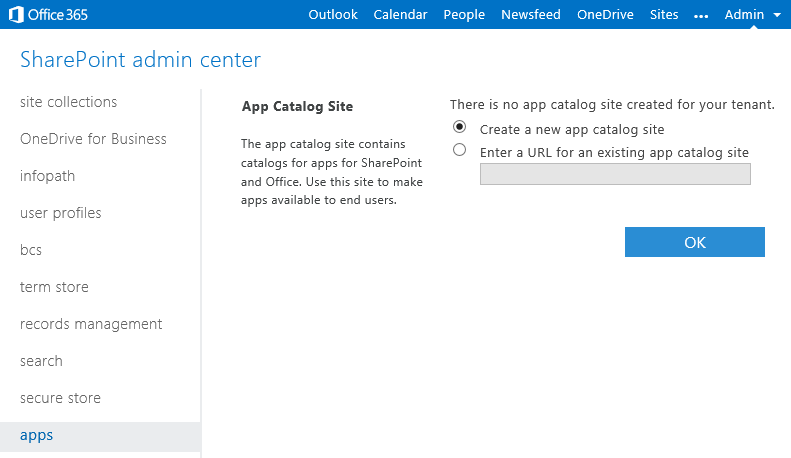
Click apps.



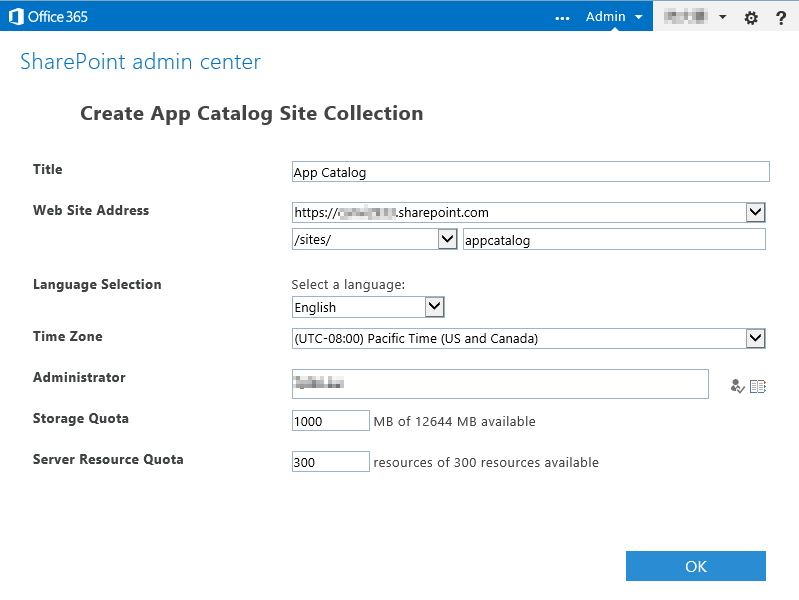
Click App Catalog.



Click OK.



Input the required fields. Click OK.

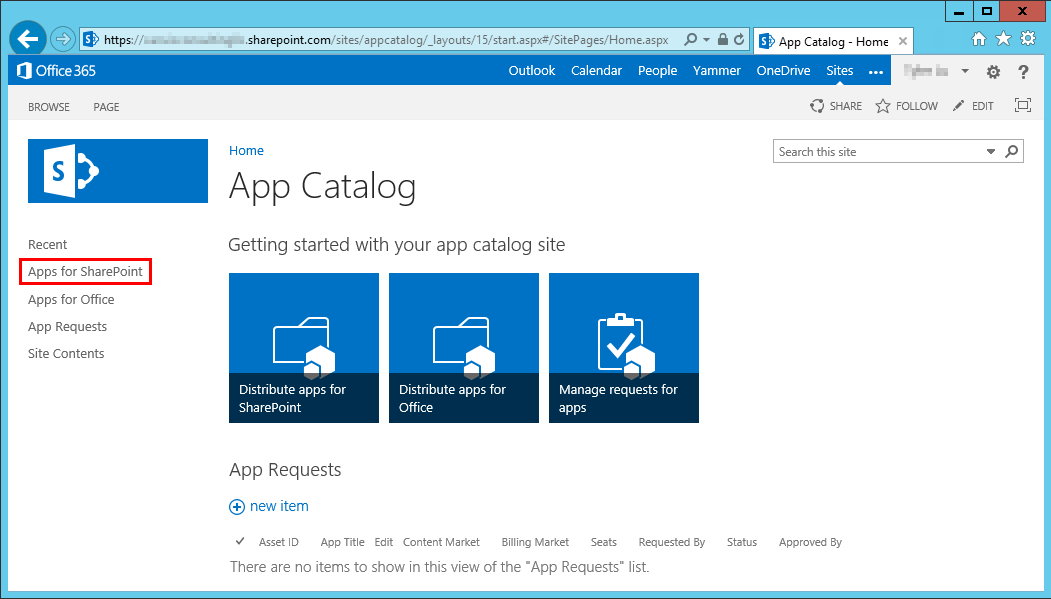


A few minutes later, the App Catalog site will be ready.

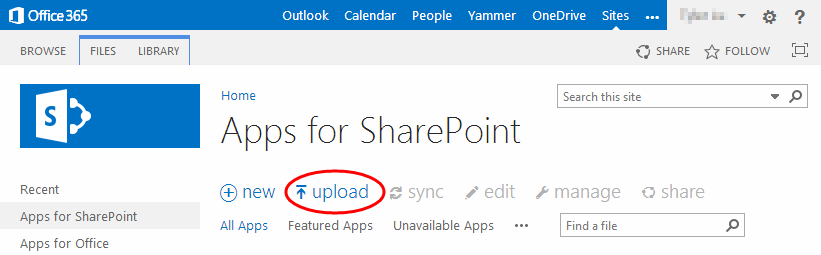
### Upload the app to App Catalog

Login to the App Catalog site.

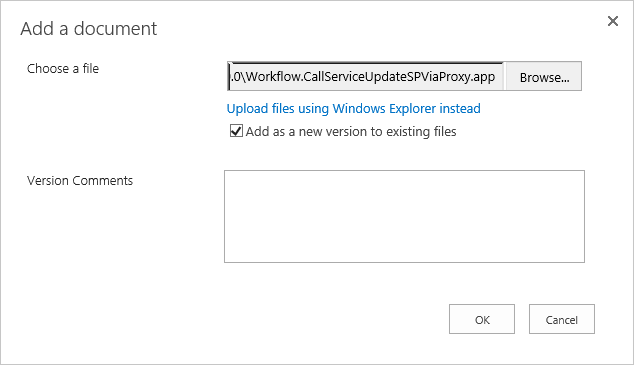
Click Apps for SharePoint.



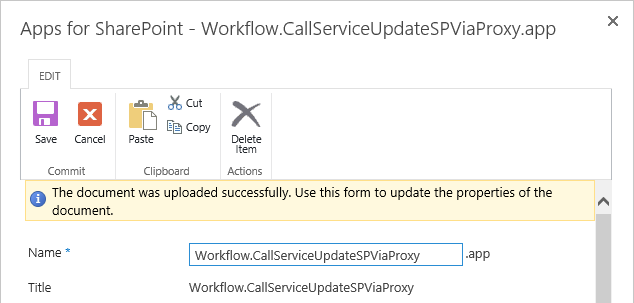
Click upload.



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



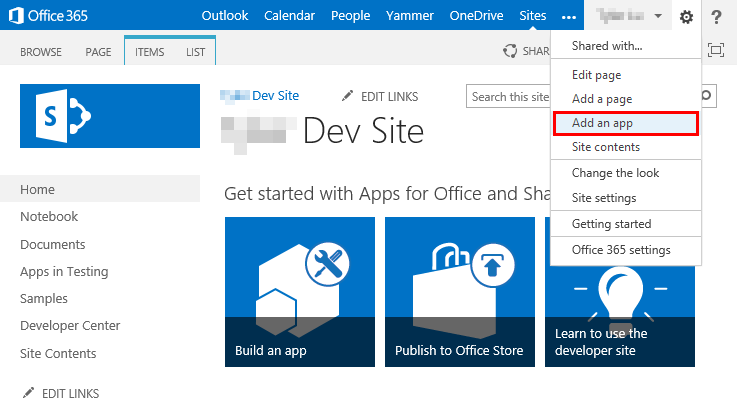
Click Save.



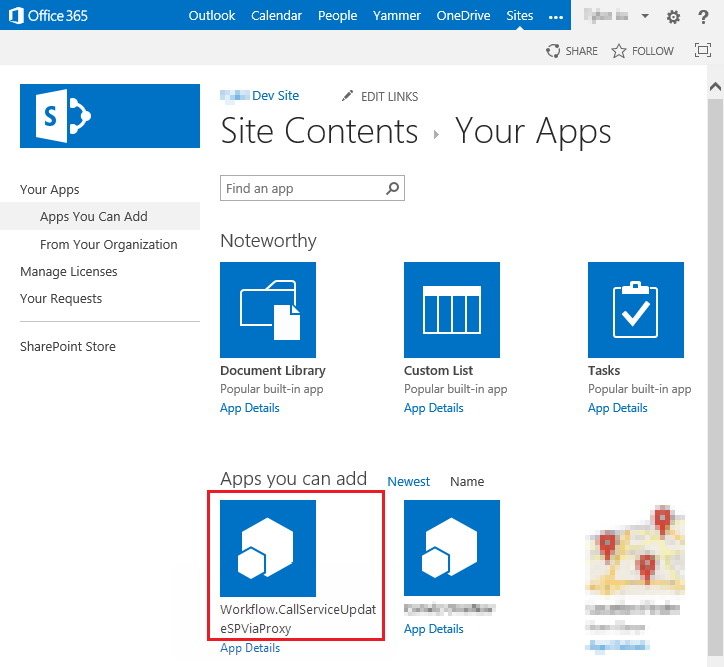
### Install the App

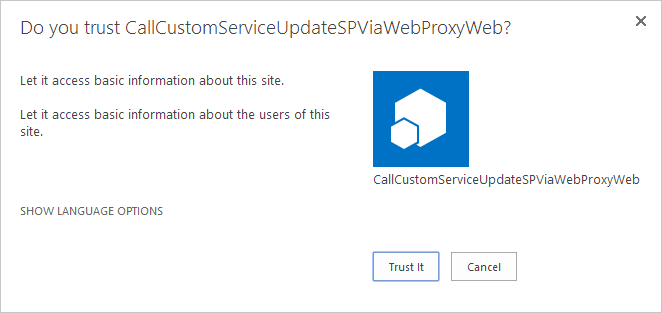
Login to the O365 site where you want to install the app.

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

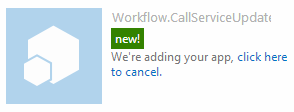


Click Workflow.CallServiceUpdateSPViaProxy.



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.

