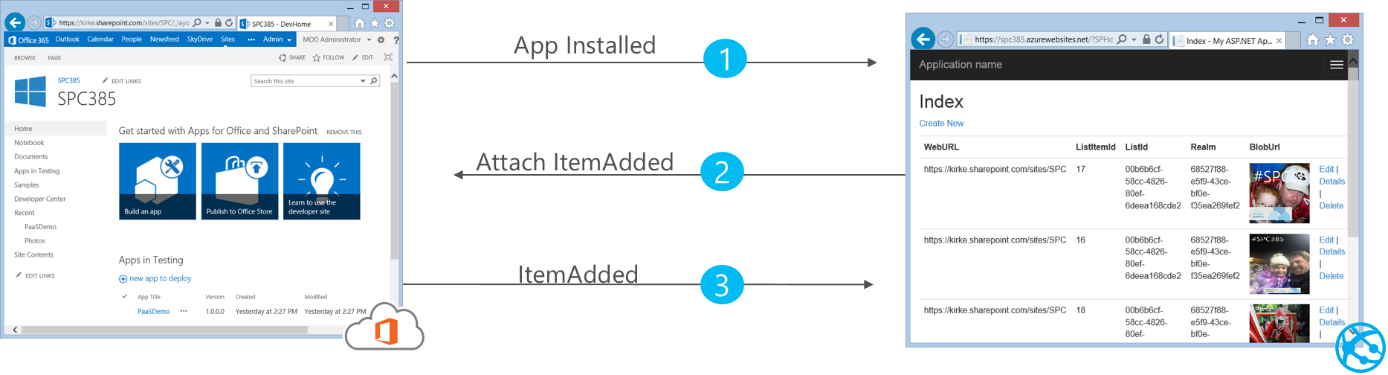
# OfFICE AMS: Remote Event Receivers

|  |  |
| --- | --- |
| Summary: | Applies to: |
| This sample shows how to attach remote event receivers to a list in the host web. | * Office 365 Multi Tenant (MT) * Office 365 Dedicated (D) * SharePoint 2013 on-premises |
| Solution: | Core.EventReceivers, version 1.0 |
| Author: | Kirk Evans (Microsoft) |
| //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // 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.  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | |

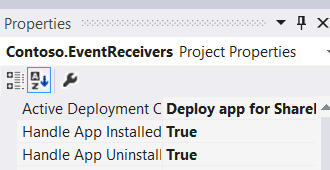
# Scenario: Using AppInstalled events to attach events in the host web

This scenario shows how an app can use the App Installed event to perform additional work in the host web, such as attaching event receivers to lists in the host web.

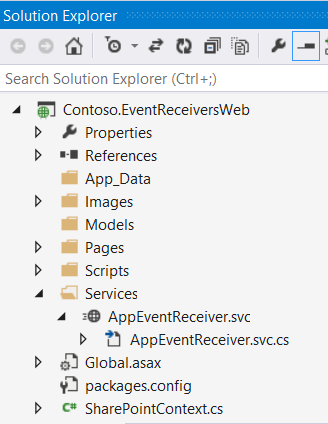


For more information on this scenario, see the blog post: <http://blogs.msdn.com/b/kaevans/archive/2014/02/26/attaching-remote-event-receivers-to-lists-in-the-host-web.aspx>.

The solution is a provider-hosted app (remote event receivers are not supported with SharePoint-hosted apps). After creating the app, go to the app project in Visual Studio and change the **Handle App Installed** and **Handle App Uninstalling** properties to True.



Visual Studio will then add a new WCF service to your web application project named AppEventReceiver.cs.



Visual Studio will also provide sample code for a remote event receiver. We can replace that code and check the type of event that was received:

private const string RECEIVER\_NAME = "ItemAddedEvent";

private const string LIST\_TITLE = "Remote Event Receiver Jobs";

public SPRemoteEventResult ProcessEvent(SPRemoteEventProperties properties)

{

SPRemoteEventResult result = new SPRemoteEventResult();

switch (properties.EventType)

{

case SPRemoteEventType.AppInstalled:

HandleAppInstalled(properties);

break;

case SPRemoteEventType.AppUninstalling:

HandleAppUninstalling(properties);

break;

case SPRemoteEventType.ItemAdded:

HandleItemAdded(properties);

break;

}

return result;

}

When an event is received indicating the app was installed, our code checks to see if a list named “Remote Event Receiver Jobs” exists in the host web and, if it does not, it creates it. The code then checks to see if that list has a remote event receiver attached to it for the ItemAdded event and, if it does not, it attaches one.

EventReceiverDefinitionCreationInformation receiver =

new EventReceiverDefinitionCreationInformation();

receiver.EventType = EventReceiverType.ItemAdded;

//Get WCF URL where this message was handled

OperationContext op = OperationContext.Current;

Message msg = op.RequestContext.RequestMessage;

receiver.ReceiverUrl = msg.Headers.To.ToString();

receiver.ReceiverName = RECEIVER\_NAME;

receiver.Synchronization = EventReceiverSynchronization.Synchronous;

myList.EventReceivers.Add(receiver);

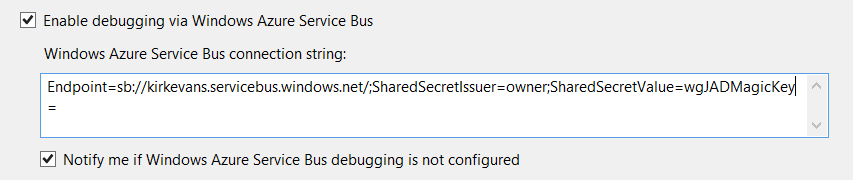
clientContext.ExecuteQuery();

Once this code executes and a call to clientContext.ExecuteQuery() is made, the list will now have a remote event receiver attached to it.

The address for the remote event receiver currently uses the same address that hosts the App Installed remote event receiver by inspecting the WCF “To” header. This is done to facilitate easy debugging using Windows Azure Service Bus. The same code works while debugging locally as well as when the app is deployed to production.

# Executing the Sample App

To execute the sample, first change the SharePoint URL to a valid SharePoint environment configured for apps. Next, go to the Contoso.EventReceivers app project and choose Properties to reveal the SharePoint tab. Scroll down to ensure a Windows Azure Service Bus connection string is configured.

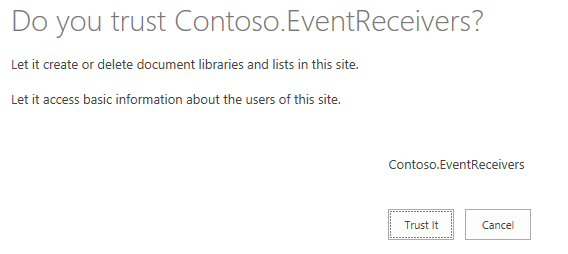


For more information on Remote Event Receiver debugging, see the section “Debugging Remote Events” at <http://msdn.microsoft.com/en-us/library/office/jj220047.aspx#DebugRER>.

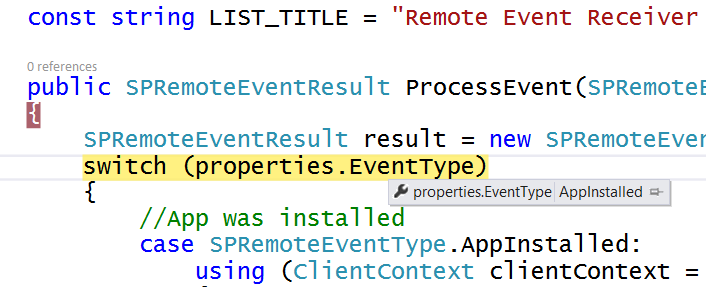
Once the Windows Azure Service Bus connection string is configured, simply press F5 in Visual Studio. The web project will run in IIS Express, while the app is deployed to SharePoint.

Add a breakpoint in the ProcessEvent method to debug the app.

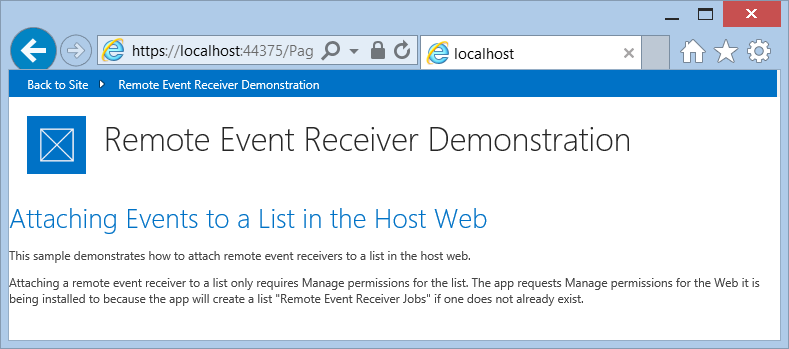
When the app is run, you must click **Trust It** in order to grant the necessary permissions.



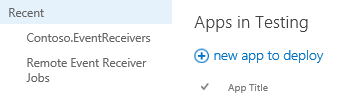
The app begins to install, and the breakpoint will be hit.

****

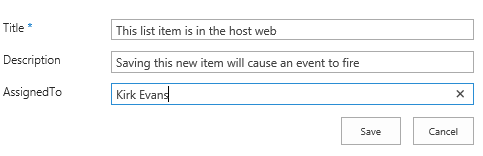
Continue debugging, and the app will finally render the full-page experience.



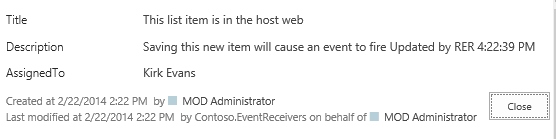
Click the “Back to Site” link in the top left to go back to the SharePoint site. Notice that the app now shows in the Recent navigation heading, as does the newly added list “Remote Event Receiver Jobs”.



Open the Remote Event Receiver Jobs list and add a new item.



Clicking Save will cause the remote event receiver endpoint to be called. The sample code in this solution simply appends text to the Description field.



## Handling App Uninstalling and Debugging

If you attempt to uninstall the app while debugging, you will notice that you receive a permission denied error. This occurs only while side-loading the app, which is what Visual Studio does when you deploy using F5. To see the AppUninstalling event work, you will need to install the app via an App Catalog or the Marketplace.

For more information, see the blog post at <http://blogs.msdn.com/b/kaevans/archive/2014/02/26/attaching-remote-event-receivers-to-lists-in-the-host-web.aspx>.

## Permissions required

Attaching a remote event receiver to an object in the host web only requires Manage permission for that object. If we were simply attaching an event to an existing list, then the app would only require Manage permission for the list. However, this sample also adds a list to the host web and activates a feature in the host web, both of which require Manage permissions for the Web.