

Decouple Power Apps with Azure Integration



Dataverse



Service Bus



Event Hubs



WebHook



Azure Functions



Power Automate



- Praveen

Contents

Why Azure Integration with Power Apps?

Flow diagram of Azure integration with Power Apps

High-level Steps of Azure Integration using Plugin Registration Tool

Azure Service Bus, Event Hub high level overview & differences

Demo -

Creation of Azure Service bus in Azure Portal

Installation of Plugin Registration Tool via Power Platform CLI

New Registration of Service Endpoint for Service Bus

New Step Registration against Service Bus Service Endpoint

Trigger the action using the model-driven app

Verify the action message in Azure Service bus Queue Explorer

Listen to the Azure Service bus Queue message via Azure Function App

Listen to the Azure Service bus Queue message via Power Automate

Creation of Azure Event Hub in Azure Portal

New Registration of Service Endpoint for Event Hub

New Step Registration against Event Hub Service Endpoint

Trigger the action using the model-driven app

Verify the action message in Event hub via stream analytics

Creation of HTTP Triggered Azure Function in Azure Portal

New Registration of Service Endpoint for Azure Function Webhook

New Step Registration against Webhook Endpoint

Trigger the action using the model-driven app

Verify the action message in Azure Function Webhook via monitor

Decouple Power Apps with Azure Integration

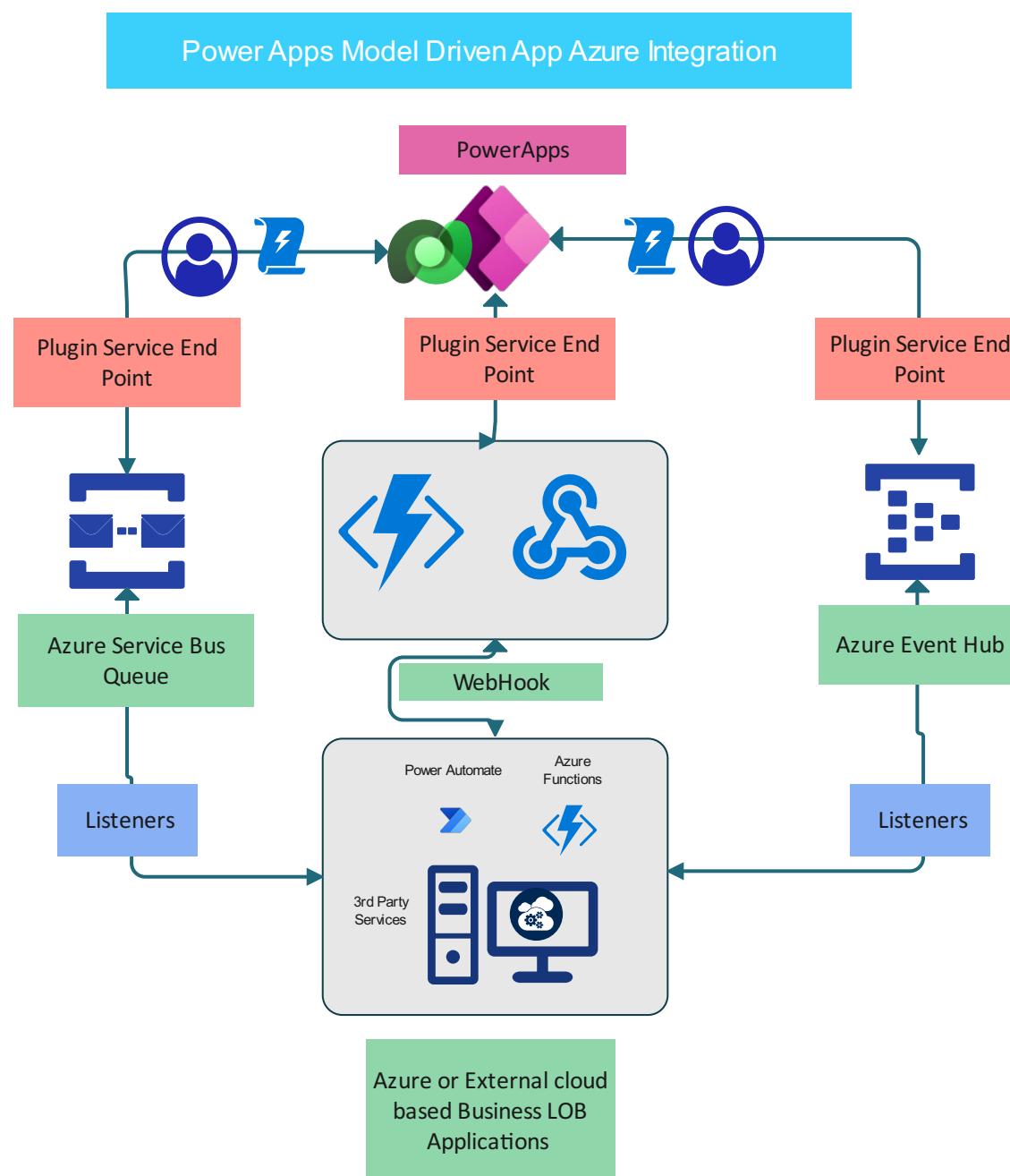


- Praveen



By integrating with Azure Service Bus / Azure Event Hub / Azure Functions.

- We are able to decouple power apps and improve the scalability and dependability of the application and services.
- Line-of-business (LOB) apps from outside the Dataverse can safely and effectively communicate with one other.
- For maintaining corporate data synchronization between numerous Dataverse systems or other Dataverse servers.



High Level Steps of integration

Create Azure Service Bus / Azure Event Hub/ Webhook

You can create an **event hub or service bus** in Azure either through API programming or interactively by using the Azure portal. Either way, after creating your event hub you must obtain a copy of the connection string and provide that string when registering the Azure service endpoint detailed in the next section.

With Microsoft Dataverse, you can send data about events that occur on the server to a web application using **webhooks**. Webhooks is a lightweight HTTP pattern for connecting Web APIs and services with a publish/subscribe model. Webhook senders notify receivers about events by making requests to receiver endpoints with some information about the events.

Register a new Service Endpoint / new Webhook

Use the Plug-in Registration tool (PRT) to register the service endpoint. When filling out the PRT registration form specify a contract type of Event Hub or service bus. For the message body format, you can choose XML or JSON. You must provide the connection string obtained when you created the event hub or service bus.

In the Plug-in Registration tool there is a new Register New WebHook option to select. When you register a WebHook you must provide three items of information - Name, Endpoint URL & Authentication (HttpHeader, WebhookKey-used for azure functions, HttpQueryString).

Register a Step & Start Listening

Dataverse needs to know the exact operation (table and message combination) that, when processed, would cause the Azure-aware plug-in to execute. Since you are creating an event hub/Service bus/Webhook, this operation would be related to the processing of Azure service bus/eventhub/functions data in particular. You must register a step for the Azure-aware plug-in in the execution pipeline.

Trigger the action

Perform an operation in Dataverse that would cause the plug-in to execute. This is the same operation (table and message combination) that you registered the plug-in step for in the previous section of this topic. You can perform the intended operation by using the power apps model driven app or web application or through application code accessing the Azure web services.

Verification

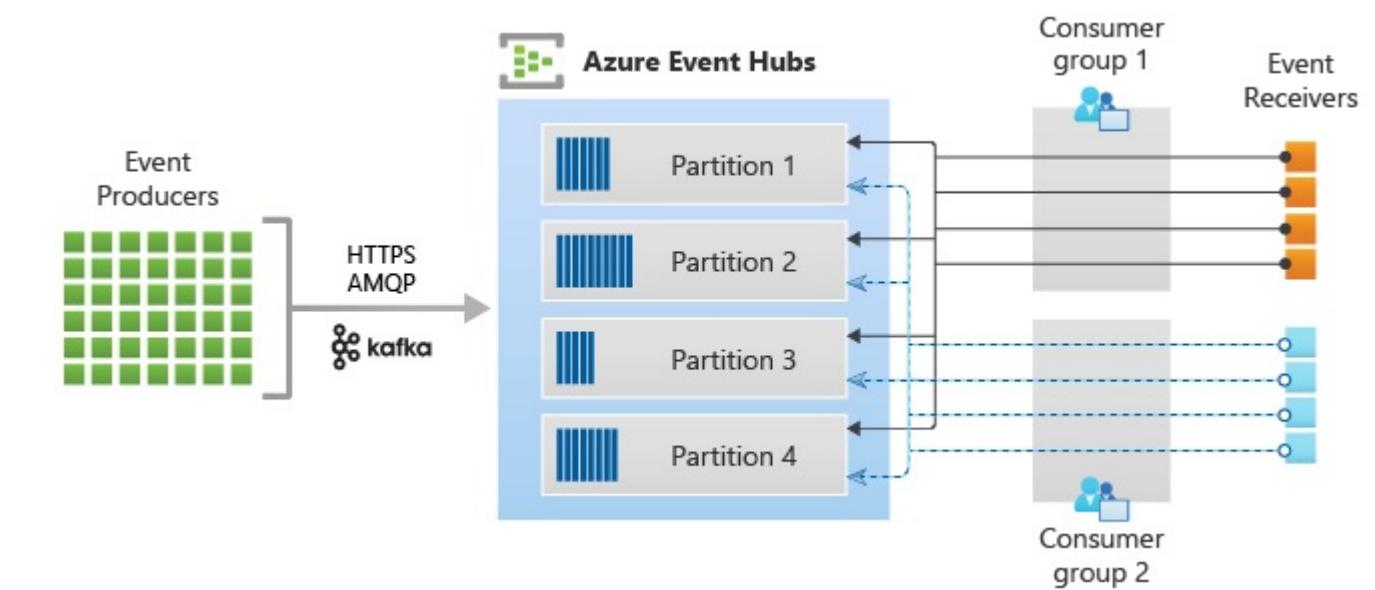
You can check the related system job in the Dataverse environment and look for a status of Succeeded. If you find a status of Failed, use the status information to identify the possible cause of the failure. You can then recheck the configurations of both systems

External cloud LOB Application Consumer listening the Actions for performing business cases

You can check the related system job in the Dataverse environment and look for a status of Succeeded. If you find a status of Failed, use the status information to identify the possible cause of the failure. You can then recheck the configurations of both systems

Azure EventHub

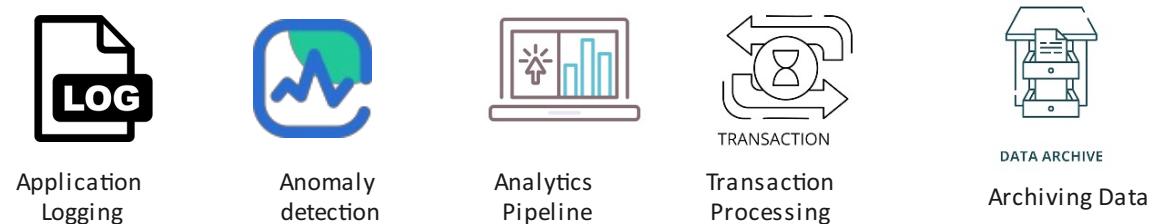
Key features



Integrations

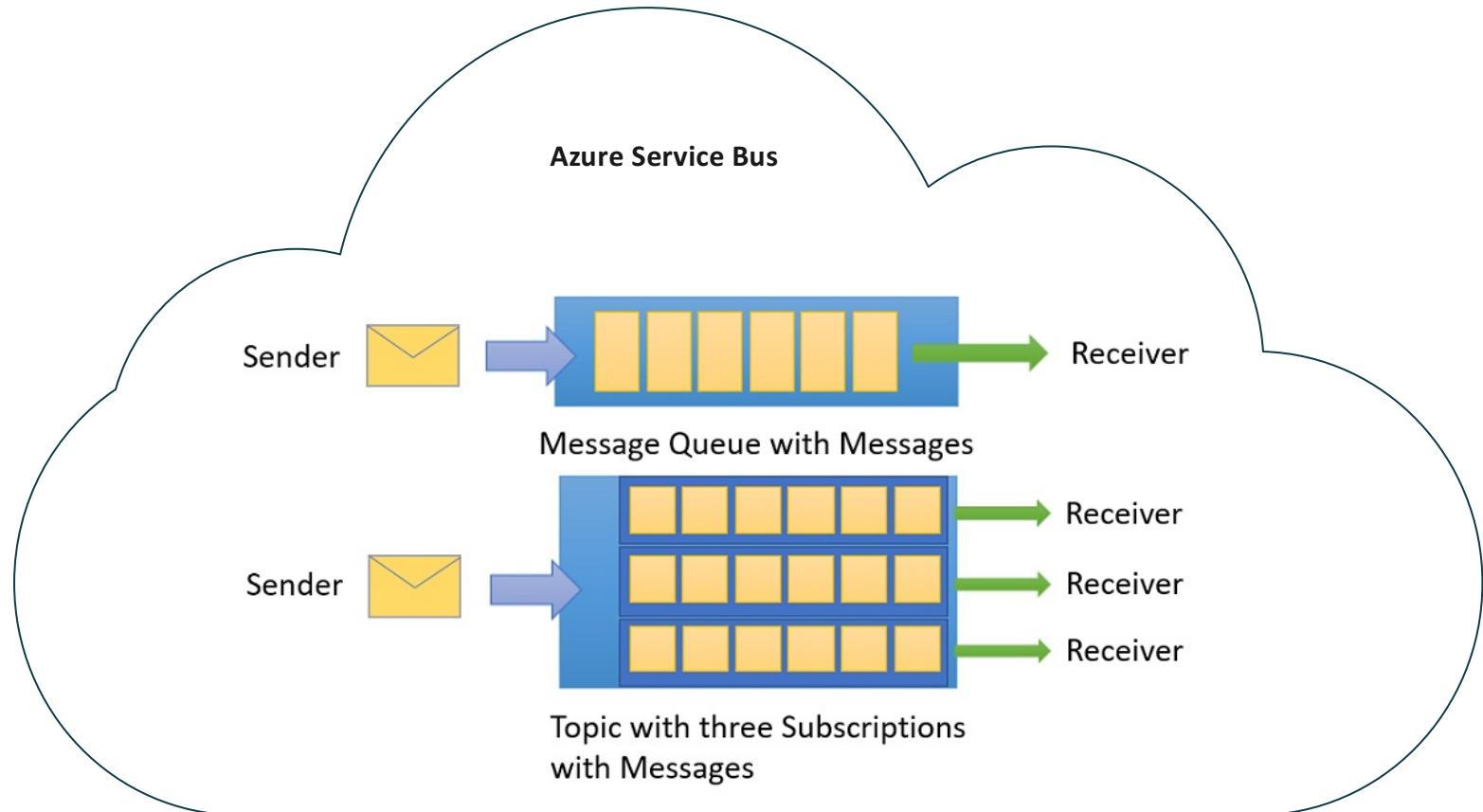


Scenarios



Azure ServiceBus

Key features



Integrations



Scenarios



When deciding between the WebHook model and the Azure Service Bus integration, here are some items to keep in mind:

- Azure Service Bus works for high scale processing, and provides a full queueing mechanism if Dataverse is pushing many events.
- Webhooks can only scale to the point at which your hosted web service can handle the messages.
- Webhooks enables synchronous and asynchronous steps. Azure Service Bus only allows for asynchronous steps.
- Webhooks send POST requests with JSON payload and can be consumed by any programming language or web application hosted anywhere.
- Both Webhooks and Azure Service Bus can be invoked from a plug-in or custom workflow activity.

	Azure Service Bus	Event Hubs
Definition	Azure Service Bus is a fully managed enterprise message broker with message queues and publish-subscribe topics	Azure Event Hubs is a big data streaming platform and event ingestion service. It can receive and process millions of events per second. Data sent to an event hub can be transformed and stored by using any real-time analytics provider or batching/storage adapters.
Scenario and usages	Messaging Decouple applications Load balancing Topics and subscriptions Transactions Message sessions	Anomaly detection (fraud/outliers) Application logging Analytics pipelines, such as clickstreams Live dashboards Archiving data Transaction processing User telemetry processing Device telemetry streaming
Key features	Queues Topics Messaging Sessions Auto-Forwarding Dead-Lettering Scheduled Delivery Message Deferral Transactions Filtering & Actions Auto Delete on Idle Duplicate Deletion Supports standard Advanced Message Queuing Protocol (AMQP) 1.0 and HTTP/REST protocols.	Fully managed PaaS Support for real-time and batch processing Capture event data - Long term retention (azure blob storage or azure data lake storage) Scalable
Integration	Event Grid Logic Apps Azure Functions Power Platform Dynamics 365 Azure Stream Analytics	Azure SQL DB Azure Storage Stream analytics Power BI Machine learning
Pricing	Basic Standard Premium more info: https://azure.microsoft.com/en-us/pricing/details/service-bus/	Basic Standard Premium more info: https://azure.microsoft.com/en-us/pricing/details/event-hubs/

Installation of Plugin Registration Tool & Power Platform Tools in Visual Studio

We can either use PRT or Power Platform Tools in Visual studio to create service endpoints. New Service End Point creation and New step registration will have have the same interface in both PRT and visual studio power platform tools. We will be using PRT tool in the following demos.

Install Plugin Registration Tool

Steps to install PRT using Power Platform CLI

Microsoft Power Platform CLI is a simple, one-stop developer CLI that empowers developers and ISVs to perform various operations in Microsoft Power Platform related to environment lifecycle, authentication, and work with Microsoft Dataverse environments, solution packages, portals, code components, and more.

We will install Plugin Registration tool using Power Platform CLI.

If power platform cli is not installed in your system , install Using the below Link

<https://learn.microsoft.com/en-us/power-platform/developer/cli/introduction#install-using-power-platform-tools-for-visual-studio-code>

<https://learn.microsoft.com/en-us/power-platform/developer/cli/introduction#install-power-platform-cli-for-windows>

Once CLI is installed, open the CLI and type the below command to check available/installed tools for the system,
pac tool list

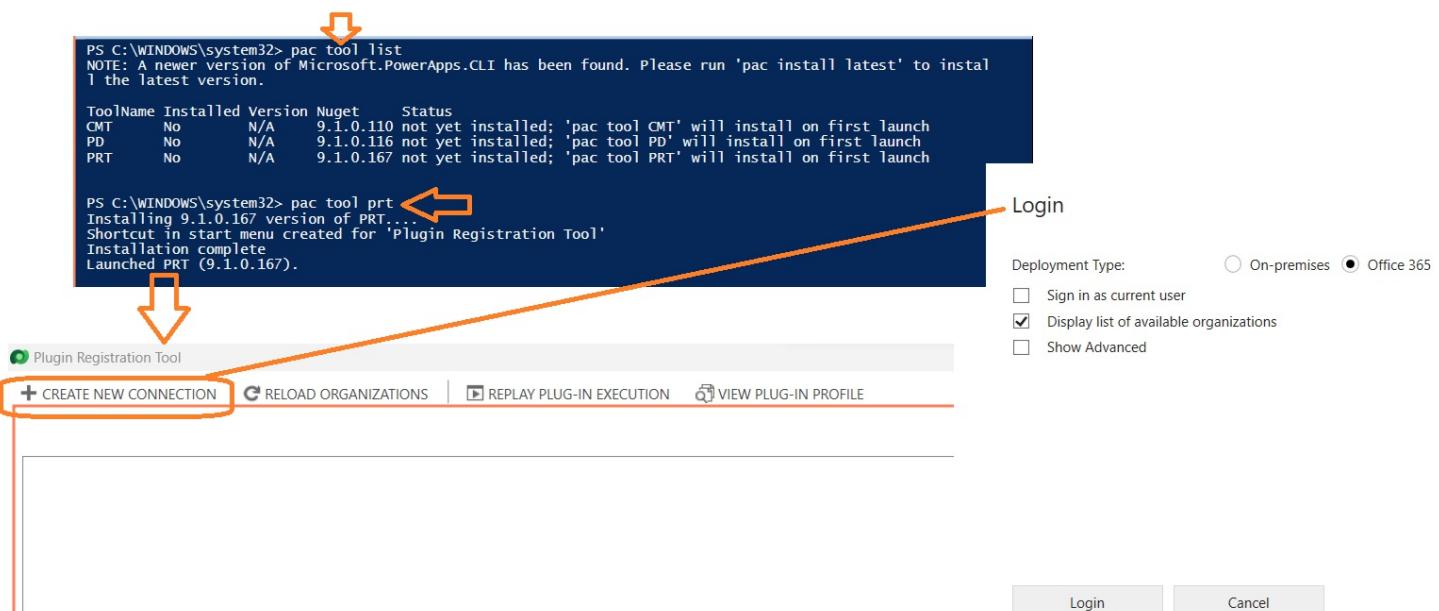
If the PRT tools is not installed on the system, type the below command to install the tool in the system,

pac tool prt

pac tool prt will install the Plugin registration tool in the sytem. Once installed you can open the PRT tool and create new connection.

Click on Office 365 and select Display list of available organizations and click on Login

Once Logged in Select the environment where you want to create service endpoint and proceed to next section



Install Power Platform Tools in Visual Studio

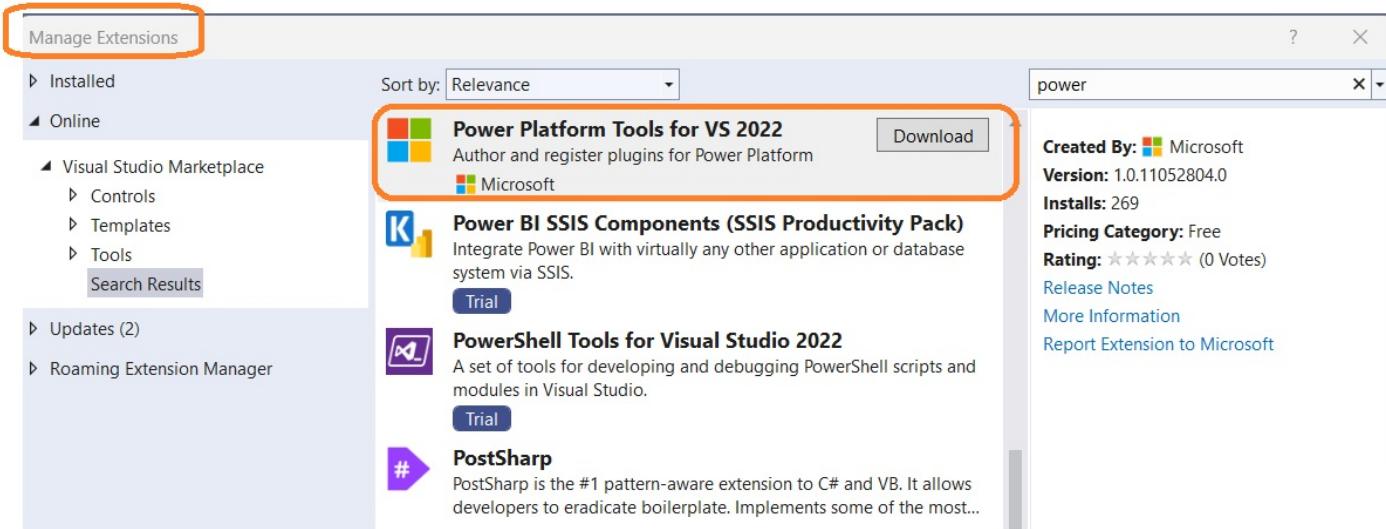
Steps to install PowerPlatformTools in VisualStudio

This is a optional part to cover the installtion steps for Power platform tools in visual studio. We can either user PRT or Power Platform Tools for visual studio.

Open Visual Studio 2019 or 2022

CLick on Extensions --> Select Manage Extensions

Under Manage Extension --> search Power Platform Tools in the visual studio marketplace and click on download

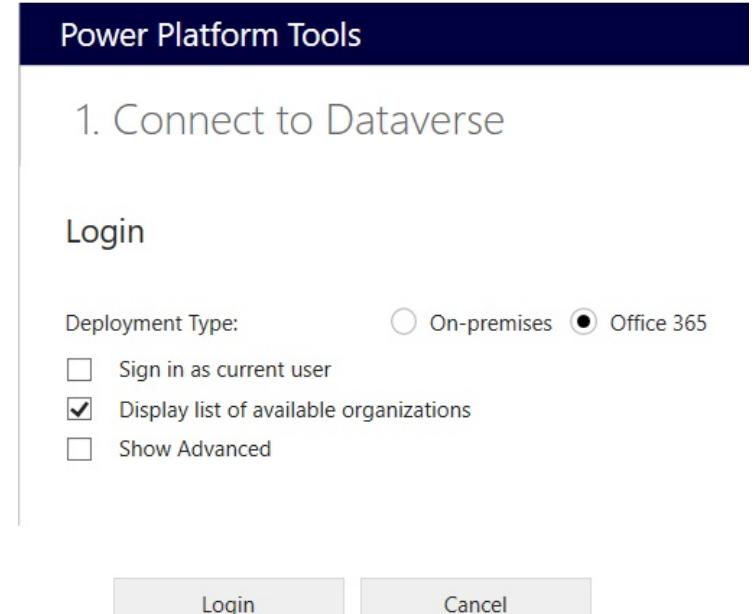
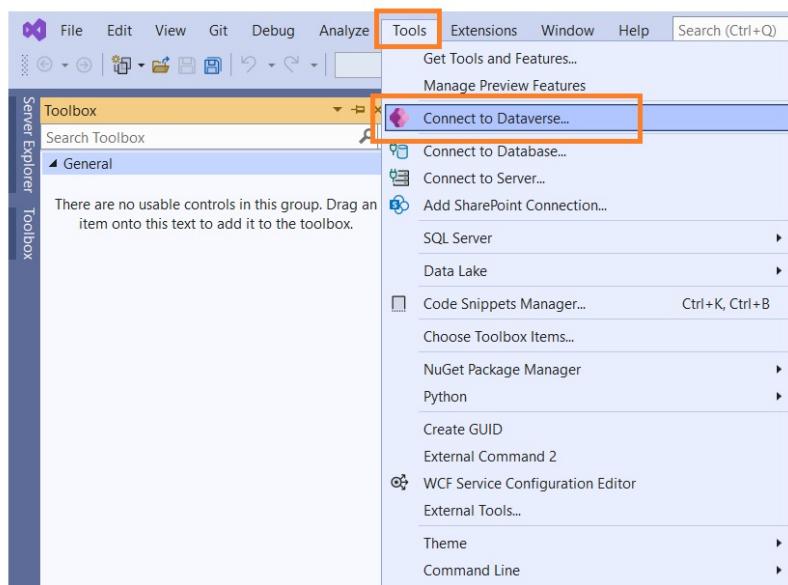


Once the tool gets downloaded, CLick Tools

Under Tools --> select COnnect to Dataverse --> this will open the Power Platform Tools

Click on Office 365 and select Display list of available organizations and click on Login

Once Logged in Select the environment where you want to create service endpoint and proceed to next section





Integrate with Azure Service Bus

Sections:

- Creation of Azure Service bus in Azure Portal
- Installation of Plugin Registration Tool via Power Platform CLI
- New Registration of Service Endpoint for Service Bus
- New Step Registration against Service Bus Service Endpoint
- Trigger the action using the model-driven app
- Verify the action message in Azure Service bus Queue Explorer
- Listen to the Azure Service bus Queue message via Azure Function App
- Listen to the Azure Service bus Queue message via Power Automate



Service Bus Queue Creation in Azure Portal

Steps for Creating Azure Service Bus NameSpace & Queue

Go to Azure portal and search for Azure Service Bus.
Select Service Bus --> Click on Create NameSpace.
Provide the subscription , resource group, Namespace name, Location, Pricing tier details and click on create.

Home > Service Bus >

Create namespace

Service Bus

Basics Advanced Networking Tags Review + create

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Free Trial

Resource group * Create new

Instance Details

Enter required settings for this namespace.

Namespace name * .servicebus.windows.net

Location * Central US

Pricing tier * Browse the available plans and their features

Review + create < Previous Next: Advanced >

Once Azure Service Bus Name space is created, go to service bus.

Under Settings --> Click on Shared access policies and copy the connection strings and paste it in notepad. These connection string will be used while creating the service end point in Plugin Registration tool.

The screenshot shows the Azure Service Bus Namespace settings page for 'servicebusazureintegrationdemo'. The left sidebar has a red arrow pointing to 'Shared access policies'. The main area shows a policy named 'RootManageSharedAccessKey' with permissions for 'Manage', 'Send', and 'Listen'. It includes fields for Primary Key (vSG1W3a8U4maXRDGxj+73CN0leBWHJA435OF4L7l0=), Secondary Key (p+T6oBQGhFbBxem/n4l6ig+4QQMYnUXkKx5p7pUMxMc=), Primary Connection String (Endpoint=sb://servicebusazureintegrationdemo.servicebus.windows.net/SharedAccess...), Secondary Connection String (Endpoint=sb://servicebusazureintegrationdemo.servicebus.windows.net/SharedAccess...), and SAS Policy ARM ID (/subscriptions/4eb1a9de-3cf2-4609-931b-5c55b8f75f8a/resourcegroups/PowerPlatfor...). A blue arrow points from the 'RootManageSharedAccessKey' policy name to the 'SAS Policy: RootManageSharedAccessKey' modal window.

For the demo, we will be creating the Service Bus Queue in this service bus namespace. So whenever some one creates a new item in the account entity this service bus queue will receive the json queue message about the newly created account item details via service endpoint and step.

Under Entities --> Click on Queue

Provide the name, and leave the other settings to default and create the queue.

The screenshot shows the Azure Service Bus Namespace entities page for 'servicebusazureintegrationdemo'. The left sidebar has a red arrow pointing to 'Queues'. The main area shows a 'Create queue' dialog with the following settings: Name (powerappsazureintegrationqueue), Max queue size (1 GB), Max delivery count (10), Message time to live (14 Days, 0 Hours, 0 Minutes, 0 Seconds), Lock duration (0 Days, 0 Hours, 0 Minutes, 30 Seconds), and two unchecked checkboxes for 'Enable dead lettering on message expiration' and 'Enable partitioning'. A large blue arrow points from the 'Queue' button in the top navigation bar to the 'Create queue' dialog.

Once the queue is created, go inside the queue.
We can see Service Bus Explorer is available. with this we can verify the new queue message.

The screenshot shows the Service Bus Explorer interface for a queue named 'powerappsazureintegrationqueue'. The left sidebar has a 'Service Bus Explorer' section highlighted with an orange box. The main area shows 'Queue (0)' and 'Dead-letter (0)'. Below that are buttons for 'Peek from start', 'Peek next messages', 'Peek with options', and 'Re-send selected messages'. A message states 'Showing 0 of 0 messages'. The table header includes columns for Sequence Number, Message ID, Enqueued Time, Delivery Count, Label/Subject, and Message Text. A note says 'Peek or receive messages to view them here.'

Now the service bus queue creation is completed. we will open the PRT tool and register the service end point and step to integrate the dataverse to Azure Service Bus.

Register Service Endpoint for Azure Service Bus

Steps for creating new service endpoint for azure service bus in PRT

Open PRT Tool and click on Create New Connection

Click on Office 365 and select Display list of available organizations and click on Login

Once Logged in Select the environment where you want to create service endpoint and proceed to next section.

(For this demo we will be creating the service end point connection for the default environment. so we have selected default environment)

The screenshot shows the 'Plugin Registration Tool' interface. At the top, there are four buttons: '+ CREATE NEW CONNECTION' (highlighted with an orange box), 'RELOAD ORGANIZATIONS', 'REPLAY PLUG-IN EXECUTION', and 'VIEW PLUG-IN PROFILE'. Below this is a 'Login' dialog box. Inside the dialog, under 'Deployment Type', there are two radio buttons: 'On-premise' (unchecked) and 'Office 365' (checked). There are also three checkboxes: 'Sign in as current user' (unchecked), 'Display list of available organizations' (checked), and 'Show Advanced' (unchecked).

Once the New Connection is registered against the environment.

Select Register --> Click on Register New Service Endpoint

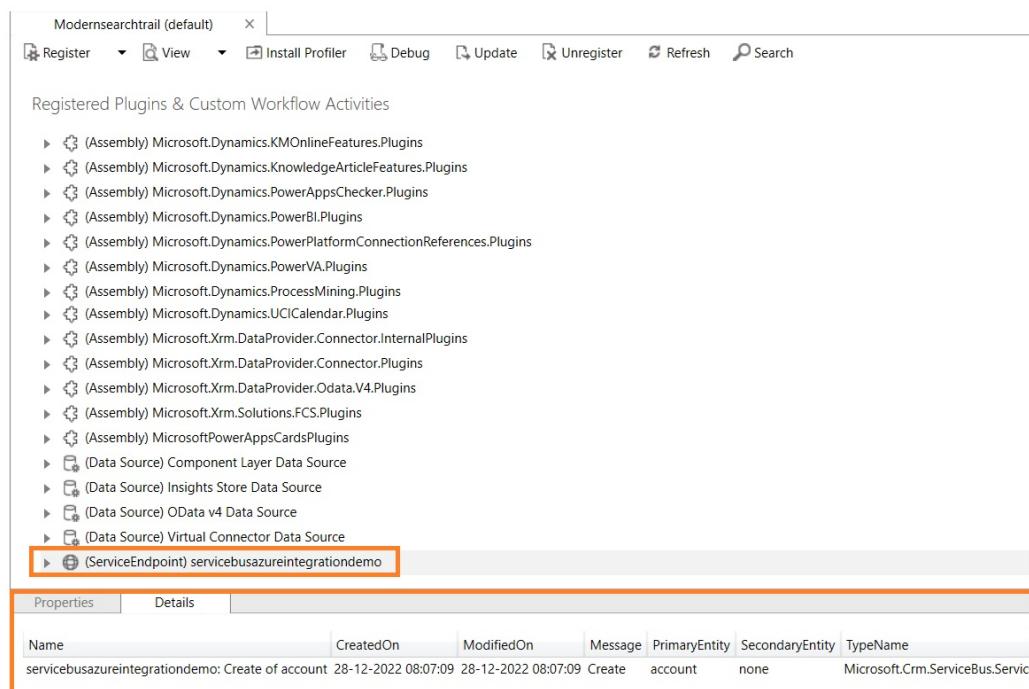
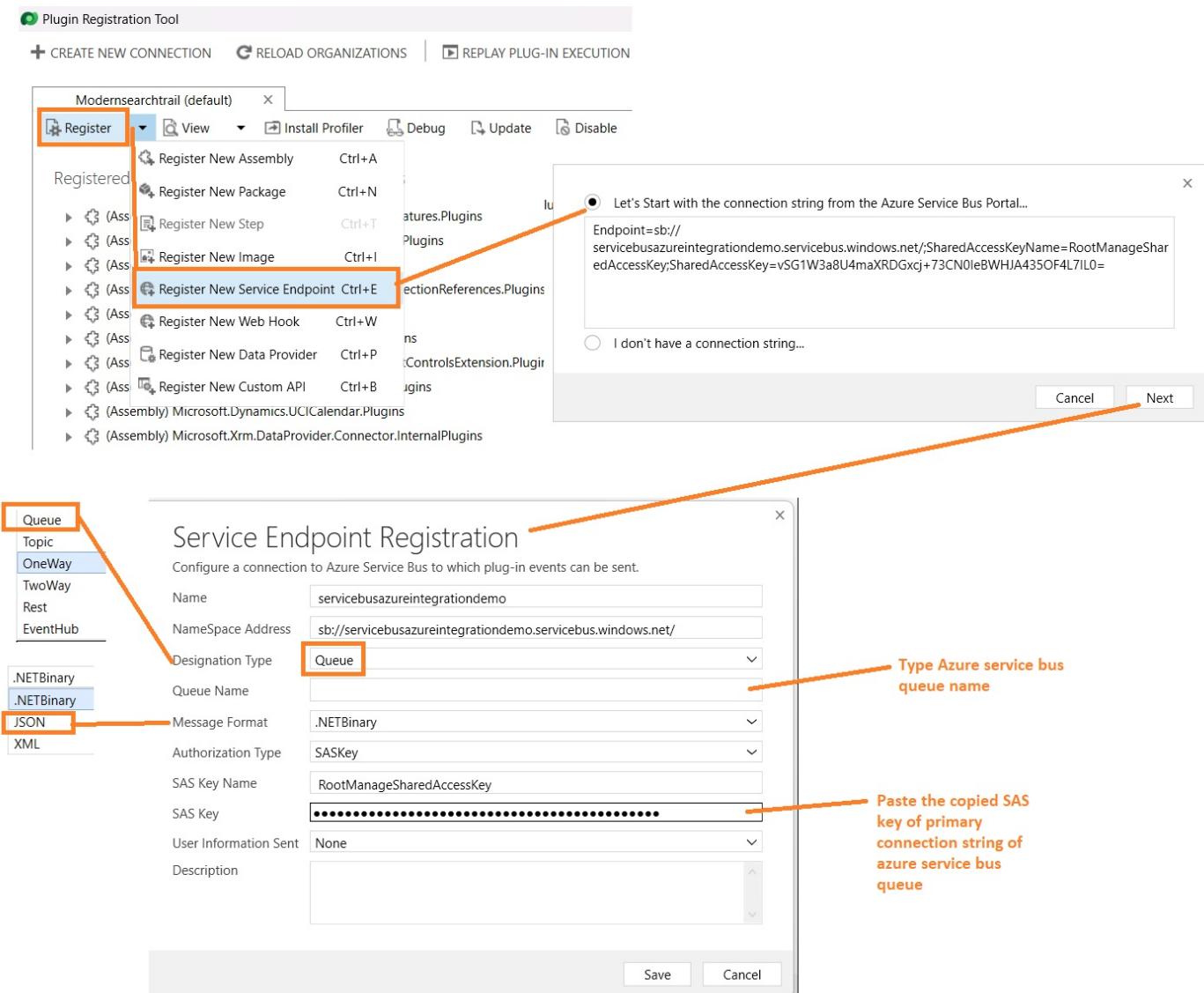
In the Service Endpoint PopUp --> copy the connection string of azure service bus copied previously and click on Next.

This will open up the Service Endpoint Registration.

In the Service Endpoint Registration popup, PRT will pick most of the details and update it to respective section based on the connection string. we need to select the destination type as "Queue" & Message format as "JSON" & verify the other fields. Once verified click on Save.

This will create the Azure service bus service endpoint registration for this environment.

Next we need Create new step against this service endpoint.



Register New Step for Azure Service Bus service endpoint

Steps for creating new step for azure service bus service endpoint in PRT

For creating the Azure aware step, we need to register the New step against the previously created Service Endpoint. Click on the newly created service endpoint, right click and select Register New Step.

In the Register New Step popup , fillin the Message as "Create", Primary entity as "account" and leave the rest as default and click on Register New Step

Register New Step

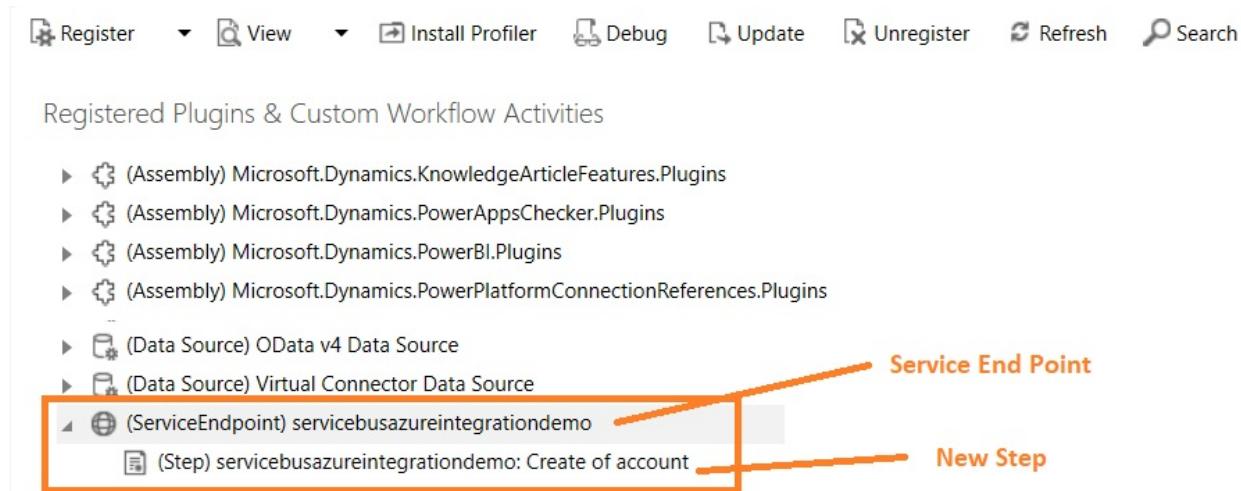
select Message & Primary Entity - leave the rest to default

Type create to configure this new step to get triggered during the creation of new item in the selected entity

Select the entity. We have selected account entity

General Configuration Information	
Message *	Create
Primary Entity	account
Secondary Entity	
Filtering Attributes	Message does not support Filtered Attributes
Event Handler	(ServiceEndpoint) servicebusazureintegrationdemo
Step Name *	servicebusazureintegrationdemo: Create of account
Run in User's Context	Calling User
Execution Order *	1
Description	servicebusazureintegrationdemo: Create of account
Event Pipeline Stage of Execution	
PostOperation	Execution Mode
	<input checked="" type="radio"/> Asynchronous
	<input type="radio"/> Synchronous
<input checked="" type="checkbox"/> Delete AsyncOperation if StatusCode = Successful	
<input type="button" value="Register New Step"/> <input type="button" value="Cancel"/>	

Once the Step is created, we can see that both service end point and step are updated in the environment. Now the configuration part is done, we can go ahead and create new items in account entity in model driven app so that the azure service bus queue will get new json queue message which will contains the details of the newly created account item.

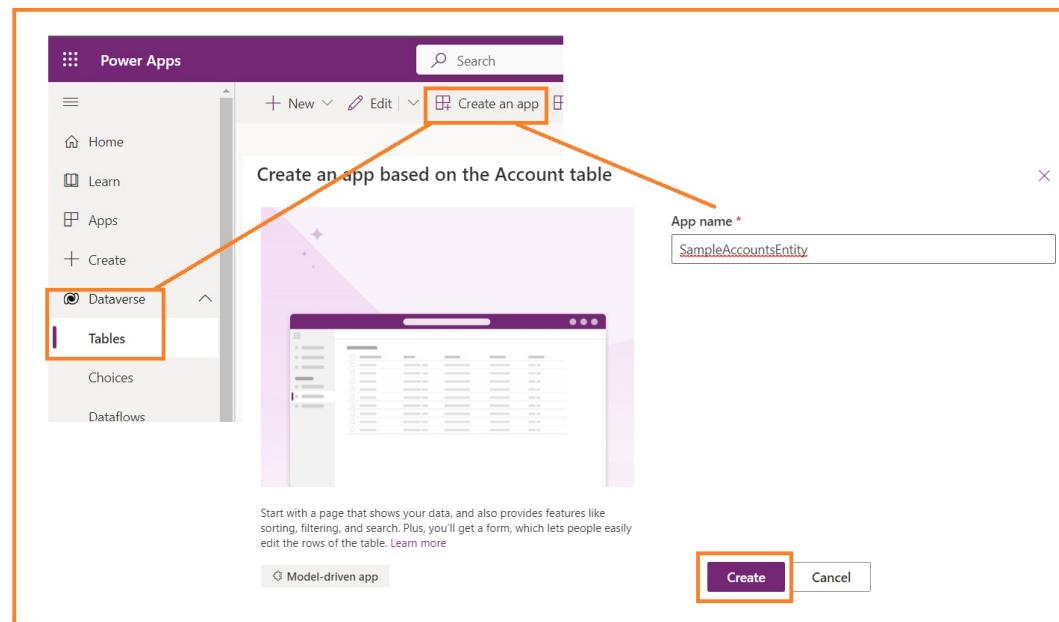


Create a test model driven app based on Account table and test the service endpoint

Steps to trigger the action to azure service bus endpoint for new item

Create a test model drive app linked with account entity or use any existing model driven app linked with account entity. Open the sample Model Driven app, create a new item for account entity and save the item.

Once the item is saved, the azure service bus queue service endpoint will get triggered and the new item will be sent as a JSON queue message.



The screenshot shows the 'Power Apps' interface. The left sidebar has 'Apps' selected. In the center, there's a list of apps including 'Deployment Pipeline Configuration', 'SampleAccounts' (which is highlighted with an orange box), and 'Solution Health Hub'. On the right, a detailed view of the 'SampleAccounts' app is shown. It displays a list titled 'My Active Accounts' with items like 'accounttest03', 'accountsazureintegrationtest1', and 'accountwebhookdemo'. Below this is a 'New Account' form with tabs for 'Summary' and 'Details'. The 'ACCOUNT INFORMATION' section includes fields for 'Account Name', 'Phone', 'Fax', 'Website', 'Parent Account', and 'Ticker Symbol'. An orange box highlights the 'Save' button at the top right of the form. To the right of the form, a numbered list of steps is displayed:

1. Go to Power Apps
2. Open Sample Model Driven App created from account entity.
3. Click on New item
4. Update the form of account entity and click on save.

Verify the action results of new item details in Azure Service Bus Queue

Steps to verify the Json response in azure service bus explorer

Once the new account item is created, go to service bus queue inside the azure portal.

Select Service Bus Explorer --> We can see that 1 queue message is available for us to peek.

Peek the message and we can able to see the JSON Queue response message of the newly created account item.

[Go to Azure Service Bus queue --> under Service Bus explorer](#)

The screenshot shows the Azure Service Bus Explorer interface. On the left, there's a sidebar with various options like Overview, Access control (IAM), Diagnose and solve problems, Service Bus Explorer (which is selected and highlighted with an orange border), Settings, Properties, Locks, Automation, Support + troubleshooting, and New Support Request. The main area has tabs for Service Bus Queue and Power Apps Integration Queue. The Power Apps Integration Queue tab is active and shows a table with one message. The table columns are Sequence Number, Message ID, Enqueued Time, Delivery Count, Label/Subject, and Message Text. The first message has a Sequence Number of 1 and a Message ID of 52ee08c4711e4d4bb01c6415ad... The Message Text column shows a JSON object. A large orange box highlights the entire JSON object in the Message Text column. Below the table, there's a detailed view of the Message Body with tabs for Message Body and Message Properties. The Message Body tab shows the JSON content, and the Message Properties tab shows various properties like CorrelationId, Depth, InitiatingUser, and InputParameters. A 'Fit message body' toggle switch is at the bottom right of this panel.

Once the message is sent to Azure Service bus Queue,

Third Party services listens to the Azure Service Bus queue will peek the message and perform the business scenario.

For this demo, we will create and show Azure Function Service Bus Queue and Power Automate as listeners. Whenever the message gets in to azure service bus queue these listeners will peek these messages and perform the business scenarios.

The listner part is just to show case the full scenario of azure integration with power apps, we will show the listener demo only for azure service bus queue.

For Event hub and Webhook we will create and verify the service endpoint & step.

Create Azure Function Service Bus Queue listener

Steps to create Azure function service bus queue listner to listen / peek new messages from azure service bus queue

Go to Azure Portal.

Search and Create Function app.

Once the function app is created --> click on functions --> create

In create functions --> select Azure service bus queue azure function

FII the details of the function name and Service bus queue

Click create.

(Refer the below screenshots)

Create Function App

Subscription * ⓘ

Resource Group * ⓘ [Create new](#)

Instance Details

Function App name *

.azurewebsites.net

Publish * Code Docker Container

Runtime stack *

Version *

Region *

Operating system

The Operating System has been recommended for you based on your selection of runtime stack.

Operating System * Linux Windows

Plan

The plan you choose dictates how your app scales, what features are enabled, and how it is priced. [Learn more](#)

Plan type * ⓘ

Home > Function App > Servicebuswebhookfunctionsdemo

Servicebuswebhookfunctionsdemo | Functions

Create function

Select development environment
Instructions will vary based on your development environment. [Learn more](#)

Development environ...

Select a template
Use a template to create a function. Triggers describe the type of events that invoke your functions. [Learn more](#)

Template	Description
HTTP trigger	A function that will be run whenever it receives an HTTP request, responding based on data in the body or query string
Timer trigger	A function that will be run on a specified schedule
Azure Queue Storage trigger	A function that will be run whenever a message is added to a specified Azure Storage queue
Azure Service Bus Queue trigger	A function that will be run whenever a message is added to a specified Service Bus queue
Azure Service Bus Topic trigger	A function that will be run whenever a message is added to the specified Service Bus topic
Azure Blob Storage trigger	A function that will be run whenever a blob is added to a specified container
Azure Event Hub trigger	A function that will be run whenever an event hub receives a new event
Azure Cosmos DB trigger	A function that will be run whenever documents change in a document collection

Create **Cancel**

Template details

We need more information to create the Azure Service Bus Queue trigger function. [Learn more](#)

New Function *

Service Bus connection * ⓘ

Queue name * ⓘ

Provide the function name

Select the azure service bus queue connection

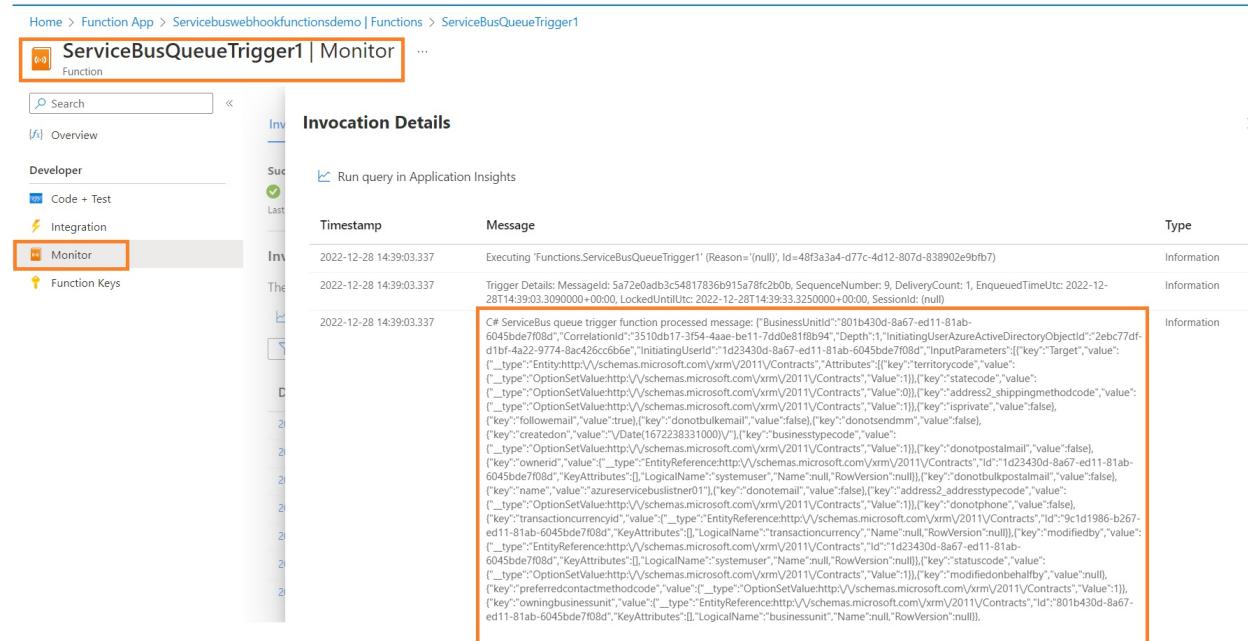
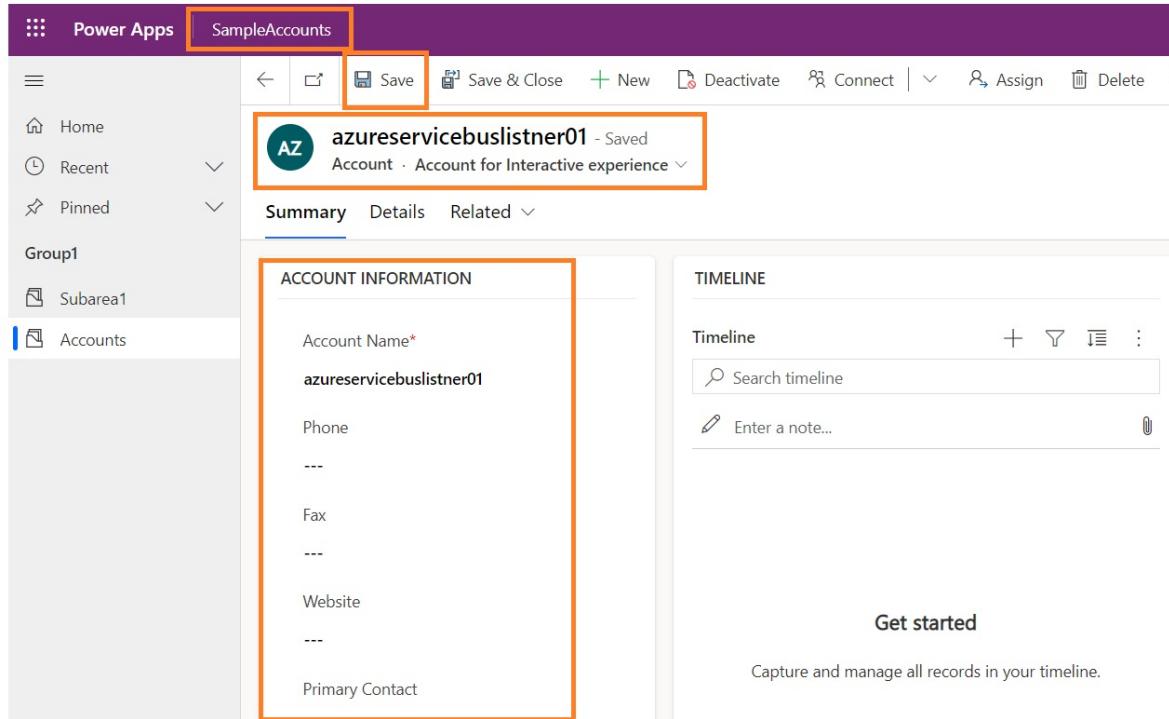
New

Provide the azure service bus queue name created in previous sections

Open the sample Model Driven app, create a new item for account entity and save the item.

Once the item is saved, the azure service bus queue will get triggered and the new item will be sent as a JSON queue message. once the item reaches azure service bus queue, the azure function listener service bus queue will peek the new message.

We can verify this by going to Azure portal --> go to Azure Service Bus Queue Listener Function --> under Monitor we can see the new JSON details



Create Power Automate as Service Bus Queue listener

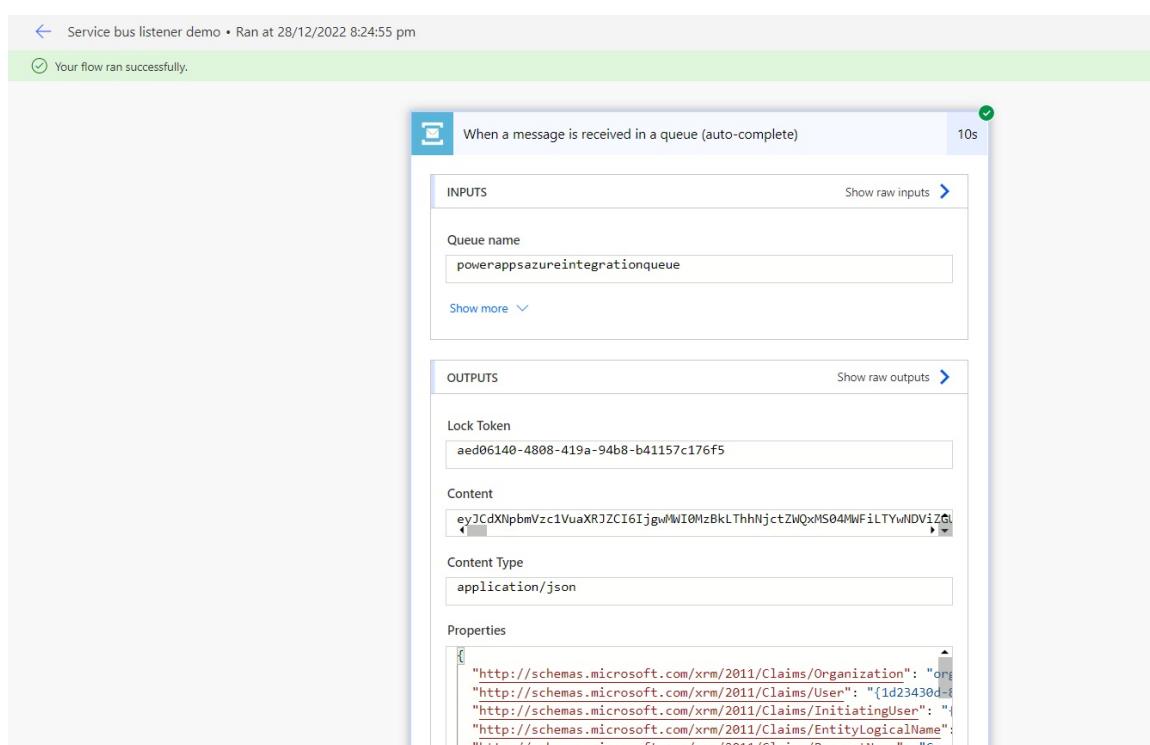
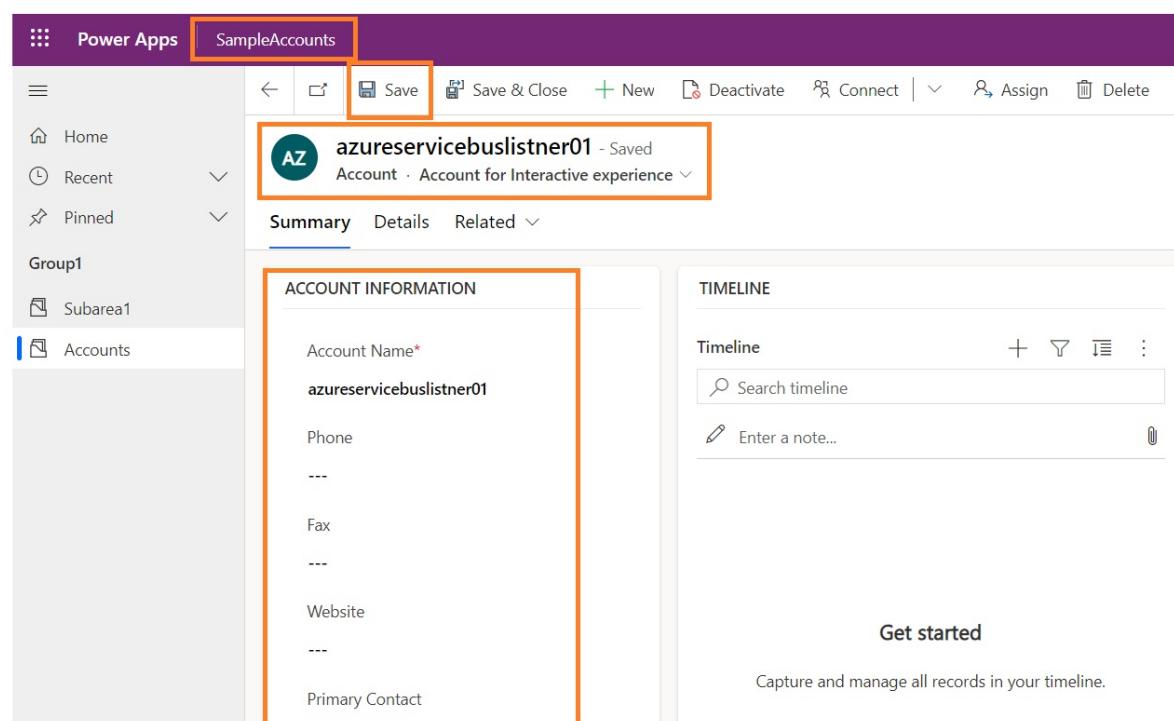
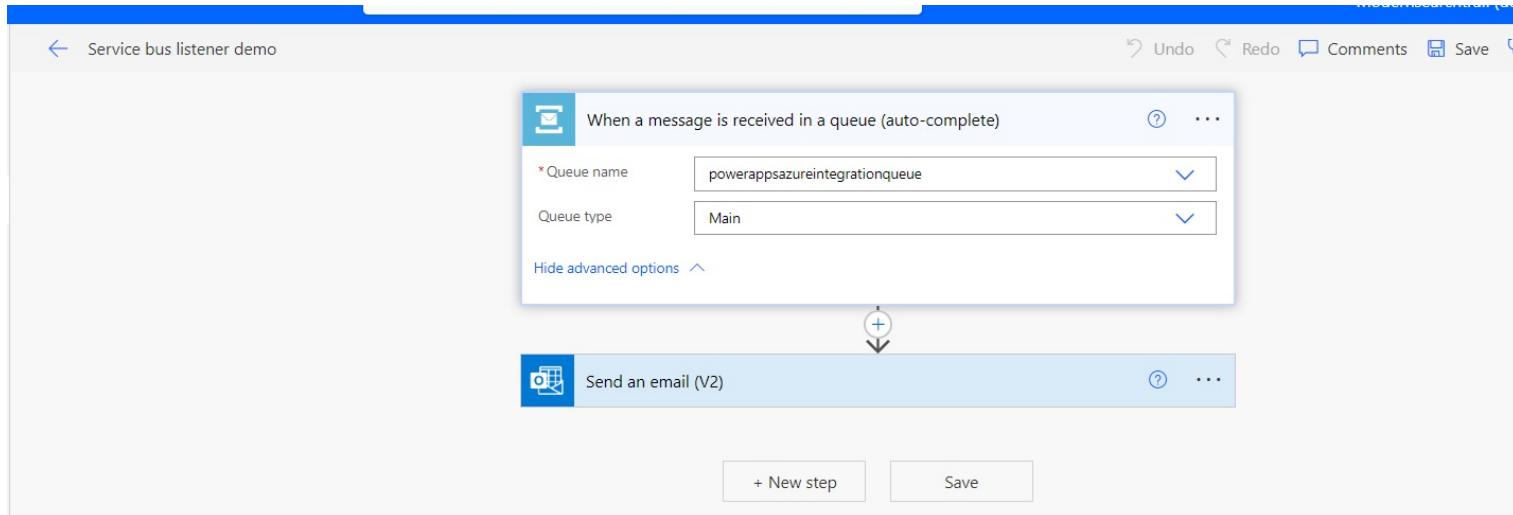
Steps to create Power Automate as Service Bus Queue listener to listen / peek new messages from Azure Service Bus Queue

Go to Power Automate --> Create new cloud flow and select when a message is received in a queue as a trigger. Add the business scenario or some steps and save the flow.

Go to Power APPS --> Open the sample Model Driven app, create a new item for account entity and save the item.

Once the item is saved, the Azure Service Bus Queue will get triggered and the new item will be sent as a JSON queue message. Once the item reaches Azure Service Bus Queue, the Power Automate Listener Service Bus Queue will peek the new message.

We can verify this by going to Power Automate --> runhistory --> we can see the new JSON details (Refer the below screenshots)



This ends the demo of azure service bus. In the next section we will see how to integrate with Azure Event hubs.

Integrate with Azure Event Hub

Sections:

- Creation of Azure Event Hub in Azure Portal
- New Registration of Service Endpoint for Event Hub
- New Step Registration against Azure Event Hub Service Endpoint
- Trigger the action using the model-driven app
- Verify the action message in Azure Event Hub Stream Analytics

Azure Event Hub Creation in Azure Portal

Steps for Creating Azure Event Hub

Go to Azure portal and search for Azure Event Hub..

Select Event Hub --> Click on Create NameSpace.

Provide the subscription , resource group, Namespace name, Location, Pricing tier details and click on create.

Once AzureEvent Hub Name space is created, go to Event Hub NameSpace.

For the demo, we will be creating the Event Hub . So whenever some one creates a new item in the account entity this Event Hub will receive the json message about the newly created account item details via service endpoint and step.

Under Entities --> Click on Event Hub

Provide the name, and leave the other settings to default and create.

Home > Event Hubs >

Create Namespace ...

Event Hubs

Basics Advanced Networking Tags Review + create

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Free Trial

Resource group * Create new

Instance Details

Enter required settings for this namespace, including a price tier and configuring the number of units (capacity).

Namespace name * .servicebus.windows.net

Location * Central US

The region selected supports Availability zones. Your namespace will have Availability Zones enabled. Learn more.

Pricing tier * Browse the available plans and their features

Throughput Units * 1

Review + create < Previous Next: Advanced >

Home > Event Hubs > sAzureintegrationEventHubDemo1 >

Create Event Hub ...

Event Hubs

Basics Capture Review + create

Event Hub Details

Enter required settings for this event hub, including partition count and message retention.

Name *

Partition count 2

Retention

Configure retention settings for this Event Hub. Learn more

Cleanup policy Delete

Retention time (hrs) 1 min. 1 hour, max. 24 hours (1day)

Review + create < Previous Next: Capture >

Under Settings --> Click on Shared access policies and copy the connection strings and paste it in notepad. These connection string will be used while creating the service end point in Plugin Registration tool.

The screenshot shows the Azure portal interface for an Event Hub namespace named 'sAzureintegrationEventHubDemo1'. In the left sidebar, under 'Settings', the 'Shared access policies' option is selected. On the right, a table lists a single policy named 'RootManageSharedAccessKey'. The 'Claims' column contains 'Manage, Send, Listen'. Below the table, there are fields for 'Primary key' (containing a long hex string), 'Secondary key' (containing another hex string), and two 'Connection string' fields (one for primary and one for secondary, both containing 'sb://' followed by the namespace name). There are also buttons for 'Save', 'Discard', 'Delete', 'Regenerate Primary Key', and more options.

Open PRT tool

Click create new connection and login .

Once the New Connection is registered against the environment.

Select Register --> Click on Register New Service Endpoint

In the Service Endpoint PopUp --> copy the connection string of azure Event Hub copied previously and click on Next.

This will open up the Service Endpoint Registration.

In the Service Endpoint Registration popup, PRT will pick most of the details and update it to respective section based on the connection string. we need to select the destination type as "EventHub" & Message format as "JSON" & verify the other fields. Once verified click on Save.

This will create the Azure Event Hub service endpoint registration for this environment.

Next we need Create new step against this service endpoint.

The New Step will be same as the one we configured earlier for service bus. (Refer the below screenshot and create the step again the event hub)

The screenshot shows the 'Plugin Registration Tool' interface with a 'Service Endpoint Registration' dialog box. The dialog has the following fields:

- Name:** azureintegrationdemoeventhub01 sazureintegrationeventhubdemo1
- NameSpace Address:** sb://sazureintegrationeventhubdemo1.servicebus.windows.net/
- Designation Type:** EventHub (highlighted with an orange box)
- EventHub Name:** azureintegrationdemoeventhub01
- Message Format:** JSON
- Authorization Type:** SASKey
- SAS Key Name:** RootManageSharedAccessKey
- SAS Key:** (redacted)
- User Information Sent:** UserId
- Description:** (empty)

At the bottom right of the dialog are 'Save' and 'Cancel' buttons. On the left side of the main window, there is a toolbar with various icons and a list of registered items including 'Register New Assembly', 'Register New Package', 'Register New Step', 'Register New Image', 'Register New Service Endpoint' (which is highlighted with a red arrow from the previous screenshot), 'Register New Web Hook', 'Register New Data Provider', 'Register New Custom API', and '(Assembly) Microsoft.Xrm.DataProvider.Connector.Plugins'.

Click on Azure Event Hub Service Endpoint --> right click and select register new step and create the step.

Register New Step

General Configuration Information

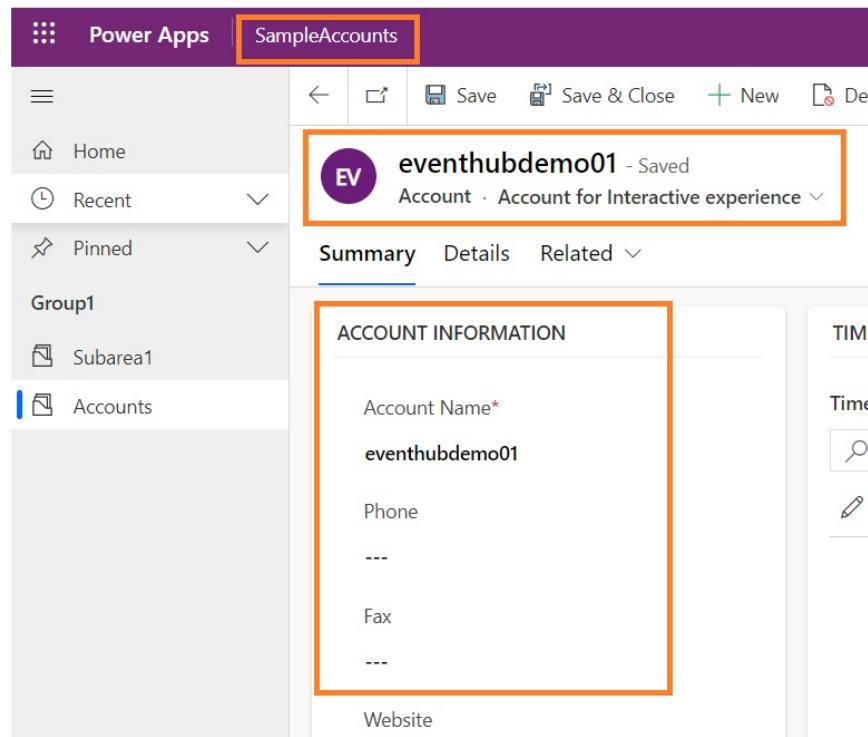
Message *	Create
Primary Entity	account
Secondary Entity	none
Filtering Attributes	Message does not support Filtered Attributes
Event Handler	(ServiceEndpoint) azureintegrationdemoeventhub01 sazureintegrationeventhubdemo1
Step Name *	azureintegrationdemoeventhub01 sazureintegrationeventhubdemo1: Create of account
Run in User's Context	Calling User
Execution Order *	1
Description	azureintegrationdemoeventhub01 sazureintegrationeventhubdemo1: Create of account

Event Pipeline Stage of Execution

PostOperation	Execution Mode	Deployment *
	<input checked="" type="radio"/> Asynchronous	<input checked="" type="checkbox"/> Server
	<input type="radio"/> Synchronous	<input type="checkbox"/> Offline
<input checked="" type="checkbox"/> Delete AsyncOperation if StatusCode = Successful		

Open the sample Model Driven app, create a new item for account entity and save the item.

Once the item is saved, the azure event hub service endpoint will get triggered and the new item will be sent as a JSON message.



We can verify the new json message response via Azure event hub stream analytics.

Go to azure portal --> Search and select the event hub

Under the event hub --> click on Process data under features.

Click on start under Enable real time insights from events (Stream Analytics Query language)

The screenshot shows the Azure Event Hub Stream Analytics interface. The top navigation bar includes 'Home > azureintegrationdemoeventhub01 (sAzureIntegrationEventHubDemo1/azureintegrationdemoeventhub01) | Process data'. On the left, a sidebar lists various features like Overview, Access control (IAM), Diagnose and solve problems, Settings, Entities, Consumer groups, Features, Capture, and Process data (which is highlighted with an orange box). The main area contains three cards: 'Materialize data in Cosmos DB', 'Filter and ingest to ADLS Gen2', and 'Start with a blank canvas'. Below these is a large box titled 'Process your Event Hub data using Stream Analytics Query Language.' It contains a sub-section titled 'Enable real time insights from events' with options to preview Event Hub data, analyze it using SQL-like query, and deploy the query by creating a new Azure Stream Analytics job. A 'Start' button is present in this section. The entire 'Process your Event Hub data...' box is also highlighted with an orange box.

This will open stream analytics query instance.

Select the query and click test query.

we can see the new json response under the preview.

The screenshot shows the Azure Stream Analytics Query interface. The top navigation bar includes 'Home > azureintegrationdemoeventhub01 (sAzureIntegrationEventHubDemo1/azureintegrationdemoeventhub01) | Process data > Query ...'. The main area has a sidebar with 'Inputs (1)', 'Outputs (1)', and 'Functions (0)'. The main workspace is titled 'Test query' and contains the following SQL-like query:

```
1 SELECT
2 *
3 INTO
4 [OutputAlias]
5 FROM
6 [azureintegrationdemoeventhub01]
```

Below the query, there are tabs for 'Input preview' and 'Test results'. The 'Input preview' tab is selected, showing sample events from 'azureintegrationdemoeventhub01'. The preview table includes columns for line numbers (78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88) and event data. One row is highlighted with a yellow background, showing the key-value pair 'name': 'eventhubdemo01'. The entire 'Test query' and 'Input preview' sections are highlighted with an orange box.

This ends the demo of integration with Azure Event Hub. In the next section we will see how to integrate with webhook.

Integrate with Webhook

Sections:

- Creation of Azure Function as http triggered webhook in Azure Portal
- New Registration of webhook Endpoint for azure function
- New Step Registration against webhook Endpoint
- Trigger the action using the model-driven app
- Verify the action message in Azure function http triggered

Azure Function App Creation in Azure Portal

Steps for Creating Http Triggered Azure Function as Webhook

Go to Azure Portal --> search azure function

Create Function App

Create Http Triggered Azure Function and copy the function url and copy it in notepad

Home > Function App >

Create Function App

Subscription * ⓘ Free Trial

Resource Group * ⓘ PowerPlatformDemo Create new

Instance Details

Function App name * Servicebuswebhookfunctionsdemo .azurewebsites.net

Publish * Code

Runtime stack * .NET

Version * 6

Region * Central US

Operating system

The Operating System has been recommended for you based on your selection of runtime stack.

Operating System * Linux Windows

Plan

The plan you choose dictates how your app scales, what features are enabled, and how it is priced. [Learn more](#)

Plan type * ⓘ Consumption (Serverless)

[Review + create](#) [< Previous](#) [Next : Networking >](#)

Webhook Endpoint and Step Creation for webhook

Steps for Webhook Endpoint

Open PRT tool

Click create new connection and login.

Once the New Connection is registered against the environment.

Select Register --> Click on Register New WebHook Registration

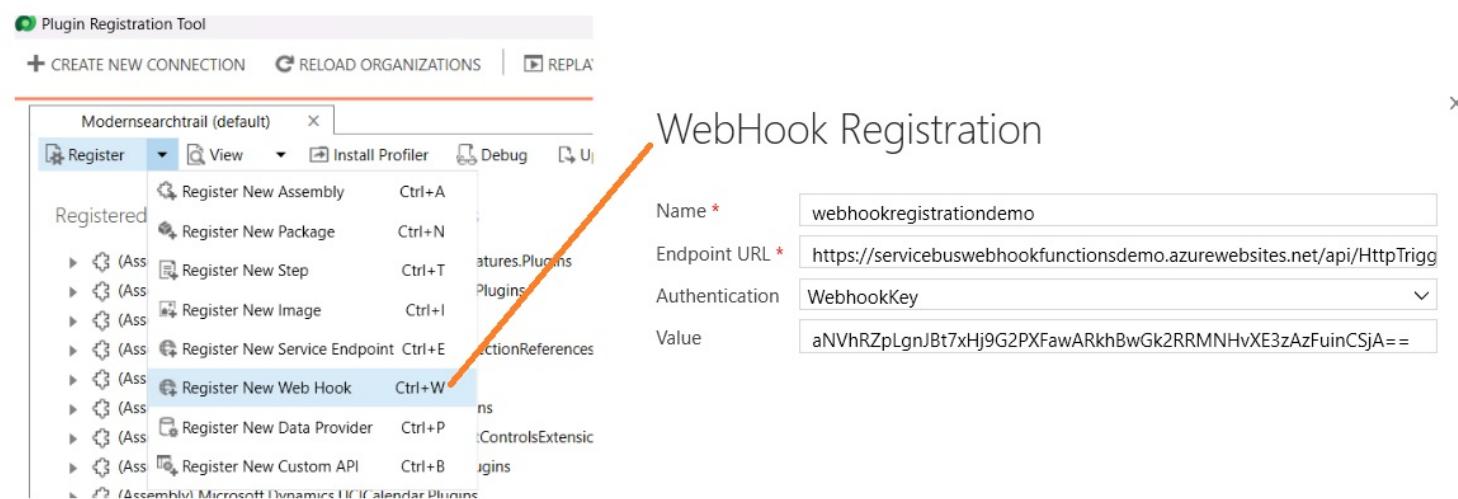
In the Webhook registration PopUp --> Paste the endpoint url as azure function url and choose authentication as WebhookKey and paste the value of azure function code value copied previously and click on Next.

Once verified click on Save.

This will create the WebHook endpoint registration for this environment.

Next we need Create new step against this webhook endpoint.

The New Step will be same as the one we configured earlier for service bus. (Refer the below screenshot and create the step again the webhook)



Register New Step

General Configuration Information

Message *	Create
Primary Entity	account
Secondary Entity	
Filtering Attributes	Message does not support Filtered Attributes
Event Handler	(WebHook) webhookregistrationdemo
Step Name *	webhookregistrationdemo: Create of account
Run in User's Context	Calling User
Execution Order *	1
Description	webhookregistrationdemo: Create of account

Event Pipeline Stage of Execution

PostOperation

Execution Mode

Asynchronous

Deployment *

Server

Synchronous

Offline

Delete AsyncOperation if StatusCode = Successful

▲ (WebHook) webhookregistrationdemo

■ (Step) webhookregistrationdemo: Create of account

Go to Power APPS

Open the sample model driven app

Add new item for account entity and click on save.

Once saved, the webhook endpoint will get triggered and the newly added item details will be sent as json to Http Trigerred azure function.

Go to Azure portal

Open Function App

Under monitor --> we can see the new item details.

The screenshot shows two main windows. The top window is a Power App titled 'SampleAccounts' showing a new account record for 'accountwebhookdemo02'. The 'Save' button is highlighted with an orange box. The bottom window is the 'HttpTrigger1 | Monitor' function in the Azure portal. The 'Logs' tab is selected, showing a log entry with the ID '4cfb-926c-1e4887224468'. The log details a request processed by the HTTP trigger function, including the JSON payload sent from the Power App.

```
4cfb-926c-1e4887224468
2022-12-28T13:34:52Z [Information] C# HTTP trigger function processed a request.
2022-12-28T13:34:52Z [Information] {"BusinessUnitId": "801b430d-8a67-ed11-81ab-6045bde7f08d", "CorrelationId": "09e8b525-a45e-c8346b4cced8", "Depth": 1, "InitiatingUserAzureActiveDirectoryObjectId": "2ebc7df-d1bf-4a22-9774-8ac426cc6b6e", "InitiatingUserId": "6045bde7f08d", "InputParameters": [{"key": "Target", "value": {"__type": "Entity:http://schemas.microsoft.com/xrm/2011/Contracts", "Value": 1}}], {"__type": "OptionSetValue:http://schemas.microsoft.com/xrm/2011/Contracts", "Value": 0}, {"key": "address2_shippingmethodcode", "value": 1}, {"__type": "OptionSetValue:http://schemas.microsoft.com/xrm/2011/Contracts", "Value": 1}, {"key": "isprivate", "value": false}, {"key": "donotbulkemail", "value": false}, {"key": "donotsendmm", "value": false}, {"key": "createdon", "value": "/Date(1672234491000)/"}, {"__type": "OptionSetValue:http://schemas.microsoft.com/xrm/2011/Contracts", "Value": 1}, {"key": "donotpostalmail", "value": false}, {"__type": "EntityReference:http://schemas.microsoft.com/xrm/2011/Contracts", "Id": "1d23430d-8a67-ed11-81ab-6045bde7f08d", "LogicalName": "systemuser", "Name": null, "RowVersion": null}, {"key": "donotbulkpostalmail", "value": false}, {"key": "name", "value": ""}, {"key": "donotemail", "value": false}, {"key": "address2_addressstypecode", "value": ""}, {"__type": "OptionSetValue:http://schemas.microsoft.com/xrm/2011/Contracts", "Value": 1}, {"key": "donotphone", "value": false}, {"__type": "EntityReference:http://schemas.microsoft.com/xrm/2011/Contracts", "Id": "9c1d1986-b267-ed11-81ab-6045bde7f08d", "LogicalName": "transactioncurrency", "Name": null, "RowVersion": null}, {"key": "modifiedby", "value": ""}, {"__type": "EntityReference:http://schemas.microsoft.com/xrm/2011/Contracts", "Id": "1d23430d-8a67-ed11-81ab-6045bde7f08d", "LogicalName": "systemuser", "Name": null, "RowVersion": null}, {"key": "statuscode", "value": ""}, {"__type": "OptionSetValue:http://schemas.microsoft.com/xrm/2011/Contracts", "Value": 1}, {"key": "modifiedonbehalfby", "value": ""}, {"key": "preferredcontactmethodcode", "value": {"__type": "OptionSetValue:http://schemas.microsoft.com/xrm/2011/Contracts", "Value": 1}}, {"key": "owningbusinessunit", "value": {"__type": "EntityReference:http://schemas.microsoft.com/xrm/2011/Contracts", "Id": "801b430d-8a67-ed11-81ab-6045bde7f08d"}}
```

This is a high level reference document to get started with Azure integration with Power Apps.

Please refer to the below video for the more details regarding the same.

<https://youtu.be/-FklzLHQFFo>

<https://youtu.be/IwYf9sQ59c8>