**Cosmos DB CDC to EventHub and Storage Account**

**Solution Overview:**

The scope of this document is to explain how to use Azure Functions to split Cosmos DB change feed (metadata and payload) across Azure EventHub and Storage Account and overcome EventHub’s record limit (1 MB).

Diagram

Description automatically generated

**Creating Azure Functions with Cosmos DB trigger:**

1. Create an azure Functions project (C#) using Visual Studio.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

1. Select Cosmos DB as the trigger and create the function.

Graphical user interface, application

Description automatically generated

1. Update “host.json” with your Cosmos DB, EventHub and storage account connect strings.

**Note**: this just for testing purpose, we can later move the connection strings/secrets to a key vault.

{

"IsEncrypted": false,

"Values": {

"AzureWebJobsStorage": "DefaultEndpointsProtocol=https;AccountName=….",

"FUNCTIONS\_WORKER\_RUNTIME": "dotnet",

"cdbtoeh": "AccountEndpoint=https://….;",

"EventHubConnection": "Endpoint=sb://…"

}}

1. Install the following NuGet packages to your Azure function project.

* Azure.Storage.Blobs
* Azure.Storage.Common
* Microsoft.Azure.Management.EventHub
* Microsoft.Azure.WebJobs.Extensions.CosmosDB
* Microsoft.Azure.WebJobs.Extensions.EventHubs
* Microsoft.Azure.WebJobs.Extensions.Storage

1. Open Functions1.cs file and replace it with code given below.

using System;

using System.Collections.Generic;

using System.IO;

using Microsoft.Azure.Documents;

using Microsoft.Azure.Storage.Blob;

using Microsoft.Azure.WebJobs;

using Microsoft.Azure.WebJobs.Host;

using Microsoft.Extensions.Logging;

namespace cosmoscdc

{

public static class Function1

{

[FunctionName("Function1")]

public static void Run(

//Input binding to receive the change feed from CosmosDB

[CosmosDBTrigger(

databaseName: "mydb",

collectionName: "mycoll",

ConnectionStringSetting = "cdbtoeh",

LeaseCollectionName = "leases")]IReadOnlyList<Document> input,

//Output binding to the Storage Account

[Blob("cdbpayload", Connection = "AzureWebJobsStorage")] CloudBlobContainer outputsa,

//Output binding to the EventHub

[EventHub("cdbpointer", Connection = "EventHubConnection")] IAsyncCollector<string> outputeh,

ILogger log)

{

if (input != null && input.Count > 0)

{

log.LogInformation("Documents modified " + input.Count);

//log.LogInformation("document Id " + input[0].Id);

//storing the change data received from input binding to a string variable

String payload = input[0].ToString();

//Generate GUID to name document received from CosmoDB

var blobName = Guid.NewGuid().ToString();

//Send the GUID to EventHub ( aks pointer to the payload)

outputeh.AddAsync(blobName);

**//if required add your business logic to perform additional transformation to the payload before sending it to the Storage Account.**

//use the same GUID to name the datafile (aka payload received from CosmosDB) and send it to the Storage account

var cloudBlockBlob = outputsa.GetBlockBlobReference(blobName + ".json");

cloudBlockBlob.UploadTextAsync(payload);

log.LogInformation("document pointer Id " + blobName + "sent to event hub and document payload " + blobName + ".json" + " sent to storage account");

}

}

}

**Publish the function to Azure:**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

**Test results:**

Add a new item to a Cosmos DB collection.

Graphical user interface, text, application

Description automatically generated

This screenshot shows Azure Function capturing the change feed for the transaction and splitting it across EventHub(pointer to payload) and Storage Account(payload)Text

Description automatically generated

Screenshot showing the pointer record received by EventHub.

Graphical user interface

Description automatically generated

Screenshot showing the payload stored on Storage Account.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated