Sender

using System;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using Microsoft.Azure.ServiceBus;

namespace TopicSender

{

        class Program

    {

        const string ServiceBusConnectionString =  "Endpoint=sb://demonamaepace.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=FEGL6iBRIqsaPdwHgCREJQIWO8JgZDgpaQWjNIoqMfQ=";

const string TopicName = "hitorytopic";

static ITopicClient topicClient;

       public static async Task Main(string[] args)

{

    const int numberOfMessages = 10;

    topicClient = new TopicClient(ServiceBusConnectionString, TopicName);

    Console.WriteLine("======================================================");

    Console.WriteLine("Press ENTER key to exit after sending all the messages.");

    Console.WriteLine("======================================================");

    // Send messages.

    await SendMessagesAsync(numberOfMessages);

    Console.ReadKey();

    await topicClient.CloseAsync();

}

static async Task SendMessagesAsync(int numberOfMessagesToSend)

{

    try

    {

        for (var i = 0; i < numberOfMessagesToSend; i++)

        {

            // Create a new message to send to the topic.

            string messageBody = $"Message {i}";

            var message = new Message(Encoding.UTF8.GetBytes(messageBody));

            // Write the body of the message to the console.

            Console.WriteLine($"Sending message: {messageBody}");

            // Send the message to the topic.

            await topicClient.SendAsync(message);

        }

    }

    catch (Exception exception)

    {

        Console.WriteLine($"{DateTime.Now} :: Exception: {exception.Message}");

    }

}

    }

}

Receiver

namespace CoreReceiverApp

{

    using System;

    using System.Text;

    using System.Threading;

    using System.Threading.Tasks;

    using Microsoft.Azure.ServiceBus;

    class Program

    {

        const string ServiceBusConnectionString = "Endpoint=sb://demonamaepace.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=FEGL6iBRIqsaPdwHgCREJQIWO8JgZDgpaQWjNIoqMfQ=";

        const string TopicName = "hitorytopic";

        const string SubscriptionName = "itsmysub";

        static ISubscriptionClient subscriptionClient;

        public static async Task Main(string[] args)

        {

            subscriptionClient = new SubscriptionClient(ServiceBusConnectionString, TopicName, SubscriptionName);

            Console.WriteLine("======================================================");

            Console.WriteLine("Press ENTER key to exit after receiving all the messages.");

            Console.WriteLine("======================================================");

            // Register subscription message handler and receive messages in a loop

            RegisterOnMessageHandlerAndReceiveMessages();

            Console.ReadKey();

            await subscriptionClient.CloseAsync();

        }

        static void RegisterOnMessageHandlerAndReceiveMessages()

        {

            // Configure the message handler options in terms of exception handling, number of concurrent messages to deliver, etc.

            var messageHandlerOptions = new MessageHandlerOptions(ExceptionReceivedHandler)

            {

                // Maximum number of concurrent calls to the callback ProcessMessagesAsync(), set to 1 for simplicity.

                // Set it according to how many messages the application wants to process in parallel.

                MaxConcurrentCalls = 1,

                // Indicates whether MessagePump should automatically complete the messages after returning from User Callback.

                // False below indicates the Complete will be handled by the User Callback as in `ProcessMessagesAsync` below.

                AutoComplete = false

            };

            // Register the function that processes messages.

            subscriptionClient.RegisterMessageHandler(ProcessMessagesAsync, messageHandlerOptions);

        }

        static async Task ProcessMessagesAsync(Message message, CancellationToken token)

        {

            // Process the message.

            Console.WriteLine($"Received message: SequenceNumber:{message.SystemProperties.SequenceNumber} Body:{Encoding.UTF8.GetString(message.Body)}");

            // Complete the message so that it is not received again.

            // This can be done only if the subscriptionClient is created in ReceiveMode.PeekLock mode (which is the default).

            await subscriptionClient.CompleteAsync(message.SystemProperties.LockToken);

            // Note: Use the cancellationToken passed as necessary to determine if the subscriptionClient has already been closed.

            // If subscriptionClient has already been closed, you can choose to not call CompleteAsync() or AbandonAsync() etc.

            // to avoid unnecessary exceptions.

        }

        static Task ExceptionReceivedHandler(ExceptionReceivedEventArgs exceptionReceivedEventArgs)

        {

            Console.WriteLine($"Message handler encountered an exception {exceptionReceivedEventArgs.Exception}.");

            var context = exceptionReceivedEventArgs.ExceptionReceivedContext;

            Console.WriteLine("Exception context for troubleshooting:");

            Console.WriteLine($"- Endpoint: {context.Endpoint}");

            Console.WriteLine($"- Entity Path: {context.EntityPath}");

            Console.WriteLine($"- Executing Action: {context.Action}");

            return Task.CompletedTask;

        }

    }

}