Microsoft

AZ-204T00A

Learning Path 10: Develop message-based solutions

Agenda

- Discover Azure message queues

Module 1: Discover Azure message queues

Learning objectives

- Choose the appropriate queue mechanism for your solution.
- Explain how the messaging entities that form the core capabilities of Service Bus operate.
- Send and receive message from a Service Bus queue by using .NET.
- Identify the key components of Azure Queue Storage
- Create queues and manage messages in Azure Queue Storage by using .NET.

Introduction

- Azure supports two types of queue mechanisms: *Service Bus queues* and *Storage queues*.
- Service Bus queues are part of a broader Azure messaging infrastructure that supports queuing, publish/subscribe, and more advanced integration patterns.
- Storage queues are part of the Azure Storage infrastructure, and they allow you to store large numbers of messages.

Choose a message queue solution

Consider using Service Bus queues when:

- Your solution needs to receive messages without having to poll the queue
- Your solution requires the queue to provide a guaranteed first-in-first-out (FIFO) ordered delivery.
- Your solution needs to support automatic duplicate detection.
- You want your application to process messages as parallel long-running streams
- Your solution requires transactional behavior and atomicity when sending or receiving multiple messages from a queue.
- Your application handles messages that can exceed 64 KB but won't likely approach the 256-KB limit.

Consider using Storage queues when:

- Your application must store over 80 gigabytes of messages in a queue.
- Your application wants to track progress for processing a message in the queue. It's useful if the worker processing a message crashes. Another worker can then use that information to continue from where the prior worker left off.
- You require server-side logs of all the transactions executed against your queues.

Explore Azure Service Bus

Some common scenarios are:

- Messaging - Transfer business data, such as sales or purchase orders, journals, or inventory movements.
- Decouple applications - Improve reliability and scalability of applications
- Topics and subscriptions - Enable 1:n relationships between publishers and subscribers
- Message sessions - Implement workflows that require message ordering or message deferral.

Premium	Standard
High throughput	Variable throughput
Predictable performance	Variable latency
Fixed pricing	Pay as you go variable pricing
Ability to scale workload up and down	N/A
Message size up to 100 MB.	Message size up to 256 KB

Discover Service Bus queues, topics, and subscriptions

Queues

- Queues offer First In, First Out (FIFO) message delivery to one or more competing consumers.

Receive modes

- You can specify two different modes in which Service Bus receives messages: Receive and delete or Peek lock.

Topics and subscriptions

- Provides a one-to-many form of communication in a publish and subscribe pattern.

Rules and actions

- You can configure subscriptions to find messages that have desired properties and then perform certain modifications to those properties.
- You can use filter actions to copy a subset of those messages to the virtual subscription queue.

Explore Service Bus message payloads and serialization

Message routing and correlation patterns

- *Simple request/reply* - A publisher sends a message into a queue and expects a reply from the message consumer.
- *Multicast request/reply* - As a variation of the prior pattern, a publisher sends the message into a topic and multiple subscribers become eligible to consume the message
- *Multiplexing* - This session feature enables multiplexing of streams of related messages through a single queue or subscription
- *Multiplexed request/reply* - This session feature enables multiplexing of streams of related messages through a single queue or subscription

Payload serialization

- The `ContentType` property enables applications to describe the payload
- The .NET Framework version of the Service Bus API supports creating `BrokeredMessage` instances by passing arbitrary .NET objects into the constructor.
- When using the legacy SBMP protocol, those objects are then serialized with the default binary serializer, or with a serializer that is externally supplied.
- When using the AMQP protocol, the object is serialized into an AMQP Bytes

Exercise: Send and receive message from a Service Bus queue by using .NET.

In this exercise you learn how to use the Azure CLI to create a Service Bus namespace and queue. You also create a .NET console app to send and receive messages from the queue.

Objectives

- Create Azure resources
- Create console app to send messages to the queue
- Review results
- Update project to receive messages to the queue
- Clean up resources

Explore Azure Queue Storage

- Azure Queue Storage is a service for storing large numbers of messages.
- A queue message can be up to 64 KB in size
- The Queue service contains the following components:
  - URL format
  - Storage
  - Queue
  - Message

The Queue service contains the following components:

- *URL format* - Queues are addressable using the URL format `https://<account>.queue.core.windows.net/<queue>`.
- *Storage account* - All access to Azure Storage is done through a storage account.
- *Queue* - A queue contains a set of messages. All messages must be in a queue.
- *Message* - A message, in any format, of up to 64 KB. For version 2017-07-29 or later, the maximum time-to-live can be any positive number, or -1 indicating that the message doesn't expire. If this parameter is omitted, the default time-to-live is seven days.

Create and manage Azure Queue Storage queues and messages by using .NET (1 of 3)

The following code examples rely on the following NuGet packages:

- Azure.Core library for .NET: This package provides shared primitives, abstractions, and helpers for modern .NET Azure SDK client libraries.
- Azure.Storage.Common client library for .NET: This package provides infrastructure shared by the other Azure Storage client libraries.
- Azure.Storage.Queues client library for .NET: This package enables working with Azure Queue Storage for storing messages that may be accessed by a client.
- System.Configuration.ConfigurationManager library for .NET: This package provides access to configuration files for client applications.

Create and manage Azure Queue Storage queues and messages by using .NET (2 of 3)

```
// The QueueClient class enables you to retrieve queues stored in Queue storage.

QueueClient queueClient = new QueueClient(connectionString, queueName);
```

```
// This example shows how to create a queue if it does not already exist

// Get the connection string from app settings
string connectionString = ConfigurationManager.AppSettings["StorageConnectionString"];

// Instantiate a QueueClient which will be used to create and manipulate the queue
QueueClient queueClient = new QueueClient(connectionString, queueName);

// Create the queue
queueClient.CreateIfNotExists();
```

Create and manage Azure Queue Storage queues and messages by using .NET (3 of 3)

```
// To insert a message into an existing queue, call the SendMessage method.

// Get the connection string from app settings
string connectionString = ConfigurationManager.AppSettings["StorageConnectionString"];

// Instantiate a QueueClient which will be used to create and manipulate the queue
QueueClient queueClient = new QueueClient(connectionString, queueName);

// Create the queue if it doesn't already exist
queueClient.CreateIfNotExists();

if (queueClient.Exists())
{
    // Send a message to the queue
    queueClient.SendMessage(message);
}
```

Summary and knowledge check

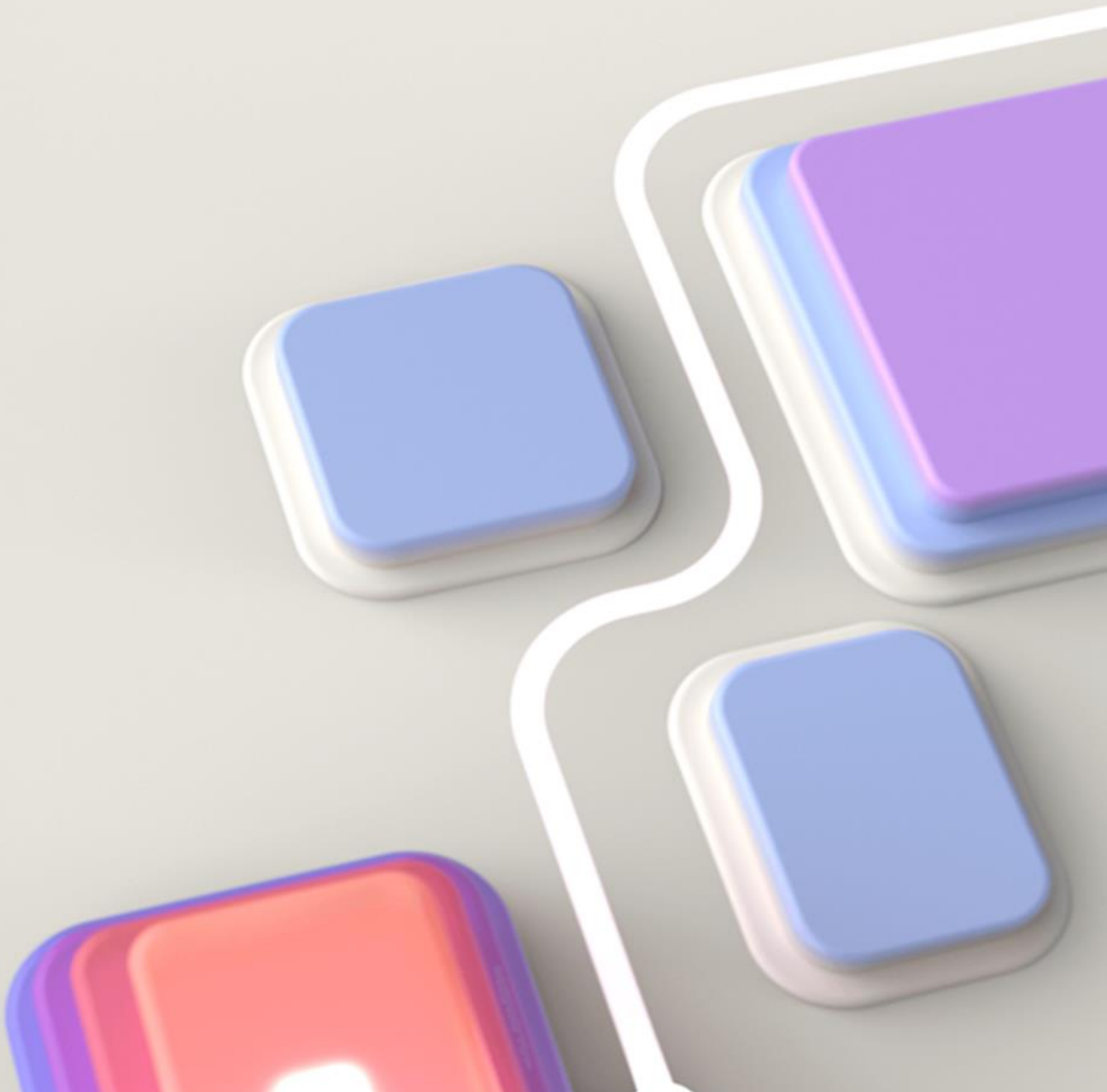
In this module, you learned how to:

- Choose the appropriate queue mechanism for your solution.
- Explain how the messaging entities that form the core capabilities of Service Bus operate.
- Send and receive message from a Service Bus queue by using .NET.
- Identify the key components of Azure Queue Storage
- Create queues and manage messages in Azure Queue Storage by using .NET.

1. What advanced feature of Azure Service Bus creates a first-in, first-out (FIFO) guarantee?

2. In Azure Service Bus messages are durably stored which enables a load-leveling benefit. What is the benefit of load-leveling relative to a consuming application's performance?

Discussion and lab



Group discussion questions

- Under what conditions would it best better to use Azure Service Bus for messages? Azure Queue Storage queues?
- Describe the four components of Azure Queue Storage.
- Describe Service Bus queues, topics, and subscriptions. What are the receive modes?

Lab 10: Asynchronously process messages by using Azure Service Bus Queues

In this lab, you will create a .NET Core project that will publish messages to the system, and a second .NET Core application that will read messages from the queue. The first app will simulate data coming from a sensor, while the second app will simulate the system that will read the messages from the queue for processing.

http://aka.ms/az204labs

- Exercise 1: Create Azure resources
- Exercise 2: Create a .NET Core project to publish messages to a Service Bus queue
- Exercise 3: Create a .NET Core project to read messages from a Service Bus queue

End of presentation